Bilevel Policy Optimization with Nyström Hypergradients

Arjun Prakash*
Brown University
Providence, RI, USA
arjun_prakash@brown.edu

Naicheng He*
Brown University
Providence, RI, USA
naicheng_he@brown.edu

Denizalp Goktas
Simulacrum
New York City, NY, USA
deni@smlcrm.com

Amy Greenwald
Brown University
Providence, RI, USA
amy_greenwald@brown.edu

ABSTRACT

The dependency of the actor on the critic in actor-critic (AC) reinforcement learning means that AC can be characterized as a bilevel optimization (BLO) problem, also called a Stackelberg game. This characterization motivates two modifications to vanilla AC algorithms. First, the critic's update should be nested to learn a best response to the actor's policy. Second, the actor should update according to a hypergradient that accounts for changes in the critic. Computing this hypergradient involves finding an inverse Hessian vector product, a process that can be numerically unstable. We thus propose a new algorithm, Bilevel Policy Optimization with Nyström Hypergradients (BLPO), which uses nesting to account for the nested structure of BLO, and leverages the Nyström method to compute the hypergradient. Theoretically, we prove BLPO converges to (a point that satisfies the necessary conditions for) a local strong Stackelberg equilibrium in polynomial time with high probability, assuming a linear parametrization of the critic's objective. Empirically, we demonstrate that BLPO performs on par with or better than PPO on a variety of discrete and continuous control tasks.

KEYWORDS

Reinforcement Learning; Bilevel Optimization; Actor-Critic

ACM Reference Format:
Arjun Prakash, Naicheng He, Denizalp Goktas, and Amy Greenwald. 2026. Bilevel Policy Optimization with Nyström Hypergradients. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 9 pages. https://doi.org/10.65109/NETG3616

1 INTRODUCTION

Bilevel optimization (BLO) is a class of hierarchical optimization problems with two objectives, an *outer* one and an *inner* one. Crucially, the objective and the variables of the outer problem depend on the solution to the inner problem. An active area of research, bilevel optimization has applications in myriad areas of machine learning and beyond. Examples include reinforcement learning [7, 45], hyperparameter optimization [27], meta-learning [33], energy markets [1], agriculture [2], and security [40], to name a few. Given functions $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ and $g : \mathbb{R}^m \to \mathbb{R}$, a(n unconstrained) bilevel opmization problem can be formulated as follows:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \Phi(\boldsymbol{x}) \doteq f(\boldsymbol{x}, \boldsymbol{y}^*(\boldsymbol{x})) \quad \text{s.t.} \quad \boldsymbol{y}^*(\boldsymbol{x}) \in \mathcal{Y}^*_{\boldsymbol{x}} \doteq \arg\min_{\boldsymbol{y}\in\mathbb{R}^m} g_{\boldsymbol{x}}(\boldsymbol{y}). \quad (1)$$

A solution to a BLO comprises a pair $(\boldsymbol{x}^*, \boldsymbol{y}^*) \in (\mathbb{R}^n, \mathbb{R}^m)$ s.t. $\boldsymbol{x}$ optimizes $\Phi(\boldsymbol{x})$ subject to the constraint that $\boldsymbol{y}^*$ optimizes $g_{\boldsymbol{x}}(\boldsymbol{y})$. The notation $g_{\boldsymbol{x}}$ allows the optimal value of g (and the corresponding solution set $\mathcal{Y}^*_{\boldsymbol{x}}$) to vary with $\boldsymbol{x}$. In this paper, we assume f and g are differentiable functions, so they can be represented by neural networks, with $\boldsymbol{x}$ and $\boldsymbol{y}$ as the network parameters.

Bi-level optimization problems are sometimes referred to as two-player general-sum Stackelberg games [9], with the outer (respectively, inner) player as the Stackelberg leader (respectively, follower), in which case solutions are called Stackelberg equilibria (SE). When the inner objective is strongly convex, solutions are guaranteed to exist under fairly general conditions [9, Theorem 5.1] and are called strong Stackelberg equilibria (SSE). In such cases, a local solution to a BLO (i.e., a local SSE) is a point $\boldsymbol{x}^*$ s.t. $f(\boldsymbol{x}^*)$ is a local minimum and $\boldsymbol{y}^*(\boldsymbol{x}^*)$ is a global minimum of $g_{\boldsymbol{x}^*}(\boldsymbol{y}^*)$.

Just as many optimization problems can be solved via gradient descent, BLOs, which comprise an outer optimization and an inner one, are naturally solved via *nested* gradient descent. For each outer value $\boldsymbol{x}$, the inner optimization is solved (e.g., by gradient descent) to find $\boldsymbol{y}^*(\boldsymbol{x})$, which is then used to update $\boldsymbol{x}$ by following the *hypergradient* of f at $\boldsymbol{y}^*(\boldsymbol{x})$. The hypergradient comprises two components: the direct gradient, which captures the direct dependence of f on $\boldsymbol{x}$, and the implicit gradient, also called the best-response Jacobian, which accounts for how changes in $\boldsymbol{x}$ affect $\boldsymbol{y}^*$.

It is straightforward to calculate the direct gradient via auto-differentiation. The implicit gradient, however, is more difficult to compute. There are two common approaches [27]. The first, called unrolling gradients, makes use of a modern auto-differentiation library like PyTorch [31] or Jax [5] to differentiate through the inner optimization algorithm, rather than through an optimal solution. This method, however, has been shown to be empirically unstable [39]. Moreover, it has large memory requirements, as it requires storing a copy of $\boldsymbol{y}$ at each gradient descent step en route to computing $\boldsymbol{y}^*$.

*Equal contribution.

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/NETG3616

The other popular approach is to leverage the implicit function theorem (IFT) to compute the implicit gradient [27]. At face value, this method requires inverting the Hessian of $g_{\boldsymbol{x}}$, which can be computationally intractable for large neural networks and numerically unstable for ill-conditioned Hessians. We tackle precisely this challenge in this paper, leveraging the Nyström method [11] to find a low-rank approximation of the inverse Hessian vector product (IHVP), which avoids materializing the full Hessian and thus can be empirically more stable than other methods [19].

Our contribution is: we posit actor-critic, a classic reinforcement learning architecture [3], as a BLO, for which we propose a nested gradient desent approach that computes hypergradients using Nyström's method. We call our reinforcement learning algorithm *Bilevel Policy Optimization (BLPO)*. We show empirically that BLPO outperforms PPO [38] on a variety of standard discrete and continuous control tasks, without imposing significant additional computational burden. We also prove that BLPO converges in polynomial time to a point satisfying the necessary conditions of a local SSE assuming a linear parameterization of the critic.[1]

1.1 Actor-Critic Methods

A reinforcement learning (RL) agent learns to make decisions by interacting with its environment sequentially (i.e., choosing actions at each state it encounters), receiving rewards along the way, with the goal of maximizing its expected return, or long-term cumulative rewards [41]. RL algorithms generally fall into two categories: policy-based methods, which directly optimize the parameters $\boldsymbol{\theta}$ of a parameterized policy $\boldsymbol{\pi_\theta}$ that maps states to actions; and value-based methods, which first estimate the expected return of a state by a parameterized value function $V_{\boldsymbol{\omega}}$, and then infer an optimal policy by selecting reward-maximizing actions at all states. Actor-critic (AC) algorithms combine these approaches by learning both a parameterized policy (the actor) and value function (the critic). Many state-of-the-art RL algorithms are built on AC-like structures, such as Trust Region Policy Optimization [36], Proximal Policy Optimization [38] and Deep Deterministic Policy Gradient [26].

Formally, a discrete-time Markov decision process (MDP) is defined by the tuple $\mathcal{M} \doteq \langle \mathcal{S}, \mathcal{A}, P, r, \gamma, \nu \rangle$. The letter $\mathcal{S}$ denotes a (possibly continuous) state space, and $\mathcal{A}$, a (possibly continuous) action space. The initial state $\boldsymbol{s}_0$ is drawn from initial state distribution ι, a probability density over $\mathcal{S}$. The transition dynamics $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ are specified by a conditional probability density $P\left[\boldsymbol{S}_{t+1} \mid \boldsymbol{s}_t, \boldsymbol{a}_t\right]$, which describes transitions to the next state $\boldsymbol{S}_{t+1}$ from the current state $\boldsymbol{s}_t$ after taking action $\boldsymbol{a}_t$. The function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ specifies the reward for taking action $\boldsymbol{a}_t$ in state $\boldsymbol{s}_t$. The return $R_\tau \doteq \sum_{t=0}^{\infty} \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t)$ along trajectory $\tau = (\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1, \boldsymbol{a}_1, \ldots)$ is defined as the discounted sum of the cumulative rewards, where $\gamma \in [0, 1]$ is the discount factor.

Given an MDP $\mathcal{M}$, a policy $\boldsymbol{\pi} : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ is a mapping from states to probability distributions over actions. The initial state distribution ι, the transition dynamics, and a policy induce a discounted history distribution $\rho_\iota^{\boldsymbol{\pi}}$ over trajectories and a discounted occupancy distribution $\nu_\iota^{\boldsymbol{\pi}}$ over states. When ι is a Dirac delta function with its impulse defined at $\boldsymbol{s}$, then abusing notation, we write $\rho_{\boldsymbol{s}}^{\boldsymbol{\pi}}$ and $\nu_{\boldsymbol{s}}^{\boldsymbol{\pi}}$.

Given a policy $\boldsymbol{\pi_\theta}$, the value function $V_{\boldsymbol{\omega}}^{\boldsymbol{\pi}} : \mathcal{S} \to \mathbb{R}$ at a state $\boldsymbol{s}$ is defined as the expected return over trajectories originating at $\boldsymbol{s}$ under policy $\boldsymbol{\pi}$: i.e., $V_{\boldsymbol{\omega}}^{\boldsymbol{\pi}}(\boldsymbol{s}) = \mathbb{E}_{\tau \sim \rho_{\boldsymbol{s}}^{\boldsymbol{\pi}}}\left[R_\tau\right]$. Given a value function $V_{\boldsymbol{\omega}}$, the actor aims to maximize the expected return $J(\boldsymbol{\theta}, \boldsymbol{\omega})$ of the policy $\boldsymbol{\pi_\theta}$, i.e., $J(\boldsymbol{\theta}, \boldsymbol{\omega}) \doteq \mathbb{E}_{\boldsymbol{s} \sim \nu_\iota^{\boldsymbol{\pi_\theta}}}\left[V_{\boldsymbol{\omega}}^{\boldsymbol{\pi_\theta}}(\boldsymbol{s})\right]$, while the critic chooses parameters $\boldsymbol{\omega}$ so as to minimize the (typically, squared) error in its value function representation, given the actor's policy $\boldsymbol{\pi_\theta}$, i.e.,

$$L(\boldsymbol{\theta}, \boldsymbol{\omega}) \doteq \frac{1}{2} \mathop{\mathbb{E}}_{\boldsymbol{s} \sim \nu_\iota^{\boldsymbol{\pi_\theta}}}\left[\left(V^{\boldsymbol{\pi_\theta}}(\boldsymbol{s}) - V_{\boldsymbol{\omega}}(\boldsymbol{s})\right)^2\right]. \tag{2}$$

The actor's objective thus depends on the critic's value function, while the critic's objective depends on the actor's policy.

Celebrated AC algorithms like PPO [38] and SAC [18] update the actor and critic simultaneously, meaning each updates the parameters of its network during iteration $t + 1$, given the other's parameters at iteration t. Simultaneous updating corresponds to a mutual better-response dynamic, which, in the event of convergence, would ideally find a solution to the following simultaneous-move game:

$$\arg\min_{\boldsymbol{\theta} \in \mathbb{R}^n} -J(\boldsymbol{\theta}, \boldsymbol{\omega}) \qquad \arg\min_{\boldsymbol{\omega} \in \mathbb{R}^m} L(\boldsymbol{\omega}, \boldsymbol{\theta}). \tag{3}$$

We make a different modeling choice, partially inspired by [45], which is to define the critic's loss function as a *parameterized* function of the actor's policy. That is, we take $L_{\boldsymbol{\theta}}(\boldsymbol{\omega}) \doteq L(\boldsymbol{\theta}, \boldsymbol{\omega})$, and model the problem as the following BLO:

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^n} \Phi(\boldsymbol{\theta}) \doteq -J(\boldsymbol{\theta}, \boldsymbol{\omega}^*(\boldsymbol{\theta})) \quad \text{s.t.} \quad \boldsymbol{\omega}^*(\boldsymbol{\theta}) \in \arg\min_{\boldsymbol{\omega} \in \mathbb{R}^m} L_{\boldsymbol{\theta}}(\boldsymbol{\omega}). \tag{4}$$

This characterization (AC-BLO) of the AC framework highlights its asymmetric nature, emphasizing the actor's policy search as the primary objective, with the critic's value function representation as a secondary objective, whose *raison d'être* is merely to aid the actor in its search. Indeed, poor estimates of the value function are known to produce less-than-ideal policies [15].

Since the critic's objective is a squared loss, a linear representation of the value function renders this objective strongly convex, ensuring a unique solution to the critic's optimization problem. The actor's optimization problem, however, is in general non-convex. Thus, assuming a linear parameterization for the critic, the solution to our AC-BLO is a local SSE, the necessary conditions of we prove can be approximated in polynomial time, with high probability.

1.2 Contributions

We propose a new policy-gradient based RL algorithm, Bilevel Policy Optimization with Nyström Hypergradient (BLPO), inspired by our AC-BLO formulation, in which the actor (definitively[2]) plays the role of the Stackelberg leader, while the critic plays the role of the follower. The nested structure of AC-BLO motivates a nested approach to solving it; specifically, it motivates iterating between taking one step along the hypergradient of the actor's objective, followed by many steps along the gradient of the critic's.

[1] Supplementary material for this paper be found here.

[2] In past work, the actor has also been nested [7], at least some of the time [45].

The aforementioned intuitions are not entirely novel, as others before us have nested the critic's computation. In practice, however, past AC-BLO formulations have not born fruit [45]. Our empirical studies reveal that the culprit is the instability of the requisite IHVP computations. In particular, iterative methods, such as conjugate gradient (CG) [20], are unstable because even regularized Hessians are poorly conditioned, and thus difficult to invert. As a result, we compute a low-rank approximation of the IHVP via Nyström's method. This modification characterizes our new algorithm, BLPO.

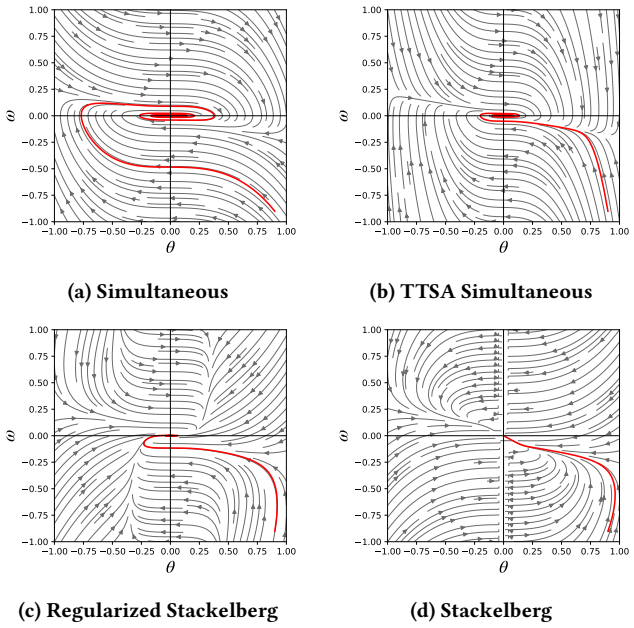
Empirically, we show that BLPO outperforms PPO in a variety of discrete and continuous control tasks. We also present a series of ablations to confirm that both nesting (with the actor as leader, and the critic as follower) and the Nyström method are essential for these performance gains. Related, we show that using CG to approximate the IHVP can lead to severe performance degradation.

Under a linear parameterization of the critic's value function, the inner optimization of our proposed AC-BLO formulation is strongly convex. Under this assumption, we prove that BLPO converges in polynomial time to a point that satisfies the necessary conditions of local SSE, with high probability. This latter caveat is necessary, as Nyström's method is randomized [11]. The rate we derive suggests a faster learning rate for the critic ($O\left(1/\kappa\right)$), and a much slower one for the actor ($O\left(1/\kappa^3\right)$), where κ is the condition number of the inner objective function. Moreover, since Nyström's method is a constant-time operation, dependent only on the number of columns sampled to build a low-rank approximation, Nyström's method eliminates a factor of $O(\sqrt{\kappa})$ from the IHVP computation as compared to CG [23].

Related Work on Game Theoretic AC. Actor-critic as a BLO (i.e., a Stackelberg game) has been considered previously by the RL community. Wen et al. [43], Zheng et al. [45] attempted to solve various MDPs using variants of AC that incorporate a hypergradient. Their methods are largely unstable, however, because they compute the hypergradient using CG, which can perform arbitrarily badly when the Hessian is ill-conditioned. Chakraborty et al. [7] developed PARL, a hypergradient-based method for RLHF, which also uses CG and also suffers from arbitrarily bad estimates of the hypergradient [10]. Hong et al. [22] assume a linear parametrization for an inner critic, and conclude that the critic should learn at a faster rate than the actor, a result corroborated here and by Zhang et al. [44].

Regarding AC as a Nash (i.e., simultaneous-move) game, Castro and Meir [6] analyze two-timescale stochastic approximate (TTSA) AC algorithms with linear function approximation, and find that simultaneous TTSA AC algorithms converge to a neighborhood around a local Nash equilibrium. The size of this neighborhood can be decreased by speeding up the learning rate of the critic at the risk of increased instability. Heusel et al. [21] sharpens this analysis to show that TTSA AC with a faster critic converges to a local Nash equilibrium almost surely. Note that these results concern Nash not Stackelberg equilibria, as these AC variants rely only on gradients, not on hypergradients. The behavior of simultaneous vs. Stackelberg training dynamics in a simple single-step MDP is shown in Figure 1.

Assuming a linear parameterization for the critic, our formulation of AC is as an unconstrained non-convex (outer) strongly-convex (inner) BLO. Ghadimi and Wang [16] analyze an iterative approach to solving non-convex strongly-convex BLOs using hypergradients, the implicit function theorem (IFT), and the conjugate gradient method. Ji et al. [23] sharpen this analysis using warm starts in the inner optimization. Hataya and Yamada [19] introduce the Nyström method to estimate the hypergradient with a low-rank approximation, overcoming challenges associated with iterative methods [27]. We combine ideas from all of the above to prove polynomial-time convergence, with high probability, to (a point that satisfies the necessary conditions of) a local SSE using hypergradients, the IFT, and Nyström's method, and we validate our approach experimentally.



(a) Simultaneous (b) TTSA Simultaneous

(c) Regularized Stackelberg (d) Stackelberg

Figure 1: We extend the one-step MDP problem from Zheng et al. [45] to show the cycling behavior of simultaneous training dynamics (Figure 1a), even with TTSA (Figure 1b). Stackelberg dynamics (Figure 1d) converge to the equilibrium, $(0, 0)$, even when regularized (Figure 1c). Further details can be found here.

2 MATHEMATICAL PRELIMINARIES

Notation. We use calligraphic uppercase letters to denote sets (e.g., $\mathcal{X}$), uppercase letters to denote matrices (e.g., H), bold lowercase letters to denote vectors (e.g., $\boldsymbol{v}$), and lowercase letters to denote scalar quantities (e.g., x). We denote an element of a matrix H by the corresponding lowercase letter using the index as a subscript, e.g., h_{ij} denotes the element at the ith row of the jth column. Similarly, we denote the jth entry of a vector by the same lowercase letter with subscript j, e.g., v_j. We use $\mathbb{R}$ to denote the set of real numbers. We use $\|\cdot\|_2$ to denote the Euclidean norm and $\|\cdot\|_{\text{op}}$ to denote the operator norm. We denote a function's parameters by a subscript (e.g., $f_{\boldsymbol{x}}$). We denote an iteration step by k using superscripted brackets (e.g., $x^{(k)}$). We denote a total gradient by ∇ and a partial derivative by ∇ with a subscript, e.g., the partial derivative of $f(\boldsymbol{x}, \boldsymbol{y})$ w.r.t. $\boldsymbol{x}$ is written $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}, \boldsymbol{y})$.

Concepts. A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be L-Lipschtiz continuous w.r.t. $\|\cdot\|$ iff $\forall \boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^n$, $\|f(\boldsymbol{x}) - f(\boldsymbol{x}')\| \leq L\|\boldsymbol{x} - \boldsymbol{x}'\|$. A function f is μ-convex if $f(\boldsymbol{x}) \geq f(\boldsymbol{x}') + \langle \nabla_{\boldsymbol{x}} f(\boldsymbol{x}'), \boldsymbol{x} - \boldsymbol{x}' \rangle + \mu/2 \|\boldsymbol{x} - \boldsymbol{x}'\|^2$. For a symmetric matrix A, we write $0 \preccurlyeq A$ if all eigenvalues of A are nonnegative, which implies A is positive semidefinite. If all eigenvalues of are strictly positive, then A is positive definite, and we write $0 \prec A$. Additionally, if $0 \preccurlyeq A - B$, then we write $B \preccurlyeq A$. If a function f has L-Lipschitz continuous gradients, and is twice differentiable, its Hessian satisfies $\nabla^2 f(\boldsymbol{x}) \preccurlyeq L\boldsymbol{I}$. If the function f is also μ-convex, its condition number is $\kappa = L/\mu$ and $\mu \boldsymbol{I} \preccurlyeq \nabla^2 f(\boldsymbol{x}) \preccurlyeq L\boldsymbol{I}$. For $\epsilon > 0$, the Euclidean ϵ-ball around x is defined as $\mathcal{B}_\epsilon(x) = \{y \in \mathbb{R}^d \mid \|y - x\|_2 \leq \epsilon\}$.

Definition 2.1. Let $h : \mathbb{R}^d \to \mathbb{R}$ be differentiable function. A point $\boldsymbol{x}^* \in \mathbb{R}^d$ is called an ϵ-**first-order stationary point** of h iff the gradient of h at $\boldsymbol{x}^*$ does not exceed ϵ, i.e., $\|\nabla h(\boldsymbol{x}^*)\|^2 \leq \epsilon$, for $\epsilon > 0$.

Definition 2.2. Let $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ and $g : \mathbb{R}^m \to \mathbb{R}$ be continuously differentiable everywhere, and assume g is μ-strongly convex. For $\epsilon > 0$, we say that $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is an ϵ-**bilevel stationary point** iff it satisfies lower-level $O(\epsilon/\mu)$-optimality, i.e., $\|\boldsymbol{y}^* - \arg\min_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}^*)} g_{\boldsymbol{x}^*}(\boldsymbol{y})\| \leq O(\epsilon/\mu)$, and $\boldsymbol{x}^*$ is an ϵ-**first-order stationary point** of f, i.e., $\|\nabla \Phi(\boldsymbol{x}^*)\|^2 \leq \epsilon$.

We seek an efficient algorithm to compute a solution to Equation (1) in the form of an ϵ-bilevel stationary point, which we can then apply to solve Equation (4).

3 BLO WITH NYSTRÖM HYPERGRADIENTS

Implicit Function Theorem. Recall the definition of a BLO presented in Equation (1). Given its hierarchical nature, the outer gradient must take into account how changes in the outer variable $\boldsymbol{x}$ affect the inner variable $\boldsymbol{y}$. This *hypergradient* is given by

$$\nabla f(\boldsymbol{x}, \boldsymbol{y}^*(\boldsymbol{x})) = \nabla_{\boldsymbol{x}} f(\boldsymbol{x}, \boldsymbol{y}) + \nabla \boldsymbol{y}^*(\boldsymbol{x}) \nabla_{\boldsymbol{y}} f(\boldsymbol{x}, \boldsymbol{y}). \tag{5}$$

The main challenge in computing the hypergradient lies in computing the implicit gradient $\nabla \boldsymbol{y}^*(\boldsymbol{x})$. Assuming ∇f is continuously differentiable and $\nabla^2_{\boldsymbol{y}\boldsymbol{y}} g_{\boldsymbol{x}}$ is well-behaved, if an optimizer of $g_{\boldsymbol{x}}$ exists (which we can ensure, for example, by assuming strong convexity), then we can invoke the implicit function theorem (IFT) to evaluate $\nabla \boldsymbol{y}^*(\boldsymbol{x})$ without explicilty solving for $\boldsymbol{y}^*(\boldsymbol{x})$:

Theorem 3.1. *Assuming $\nabla f(\boldsymbol{x}, \boldsymbol{y})$ is continuously differentiable and $\nabla^2_{\boldsymbol{y}\boldsymbol{y}} g_{\boldsymbol{x}}(\boldsymbol{y})$ is invertible, if $\tilde{\boldsymbol{y}}(\tilde{\boldsymbol{x}})$ is a first-order stationary point of $g_{\boldsymbol{x}}$ in a neighborhood around $\tilde{\boldsymbol{x}}$, i.e., $\nabla_{\boldsymbol{y}} g_{\boldsymbol{x}}(\boldsymbol{y})|_{\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}(\tilde{\boldsymbol{x}})} = 0$, for all $\boldsymbol{x} \in \mathcal{B}(\tilde{\boldsymbol{x}})$, then there exists a unique continuous function $\boldsymbol{x} \mapsto \boldsymbol{y}^*(\boldsymbol{x})$ s.t. $\nabla_{\boldsymbol{y}} g_{\boldsymbol{x}}(\boldsymbol{y})|_{\boldsymbol{x}, \boldsymbol{y}^*(\boldsymbol{x})} = 0$, for all $\boldsymbol{x} \in \mathcal{B}(\tilde{\boldsymbol{x}})$. Moreover,*

$$\nabla \boldsymbol{y}^*(\boldsymbol{x})\Big|_{\tilde{\boldsymbol{x}}} = -\nabla^2_{\boldsymbol{x}\boldsymbol{y}} g_{\boldsymbol{x}}(\boldsymbol{y}) (\nabla^2_{\boldsymbol{y}\boldsymbol{y}} g_{\boldsymbol{x}}(\boldsymbol{y}))^{-1} \Big|_{\tilde{\boldsymbol{x}}, \boldsymbol{y}^*(\tilde{\boldsymbol{x}})}. \tag{6}$$

By applying the IFT, we can re-express the second term of the hypergradient as follows:

$$\nabla \boldsymbol{y}^*(\boldsymbol{x}) \nabla_{\boldsymbol{y}} f(\boldsymbol{x}, \boldsymbol{y}) = \underbrace{-\nabla^2_{\boldsymbol{x}\boldsymbol{y}} g_{\boldsymbol{x}}(\boldsymbol{y}) \underbrace{(\nabla^2_{\boldsymbol{y}\boldsymbol{y}} g_{\boldsymbol{x}}(\boldsymbol{y}))^{-1} \nabla_{\boldsymbol{y}} f(\boldsymbol{x}, \boldsymbol{y})}_{\text{inverse Hessian vector product}}}_{\text{Jacobian vector product}}. \tag{7}$$

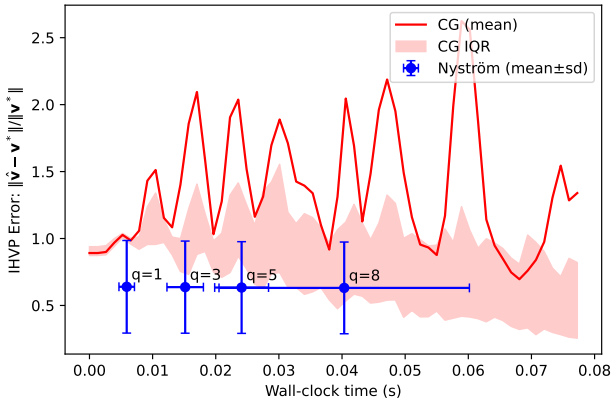


Figure 2: We calculate the IHVP error averaged over 100 small neural networks using both the Nystöm method and CG. The CG mean is consistently outside the interquartile range (IQR), which suggests at least occasional occurrence of massive errors. Further experimental details can be found here.

We then define the **inverse Hessian vector product (IHVP)** $\boldsymbol{v}^* \doteq (\nabla^2_{\boldsymbol{y}\boldsymbol{y}} g_{\boldsymbol{x}}(\boldsymbol{y}))^{-1} \nabla_{\boldsymbol{y}} f(\boldsymbol{x}, \boldsymbol{y})$.

Our goal thus reduces to computing the inverse Hessian of the inner objective. Since modern neural networks can be millions of parameters in size, storing and directly inverting their Hessians is impractical: an $m \times m$ Hessian would require $O(m^3)$ operations [25]. Nonetheless, it is possible to compute the IHVP via iterative methods, like conjugate gradient [8].

Approximating the hypergradient. Given the computational intractability of directly computing the IHVP, many approaches have been proposed to instead approximate $\boldsymbol{v}$. One of the most popular is the conjugate gradient (CG) method [20], which iteratively refines the solution to the linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ by updating in conjugate directions until convergence to $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$. In our application, $\boldsymbol{A} = \nabla^2_{\boldsymbol{y}\boldsymbol{y}} g_{\boldsymbol{x}}$ and $\boldsymbol{b} = \nabla_{\boldsymbol{y}} f$.

A standard way to improve the performance of CG is to use a preconditioner (PCG): choose a symmetric positive definite matrix $\boldsymbol{M}$, and then solve $A'\hat{\boldsymbol{x}} = b'$, where $A' = \boldsymbol{M}^{-1/2}\boldsymbol{A}\boldsymbol{M}^{-1/2}$, $\hat{\boldsymbol{x}} = \boldsymbol{M}^{-1/2}\boldsymbol{x}$, and $b' = \boldsymbol{M}^{-1/2}b$. If the preconditioner is chosen well, PCG reduces the effective condition number—for example, by rescaling the diagonal—yielding faster convergence. Choosing a preconditioner can be an art, however [29].

While CG overcomes the memory issue of storing the entire Hessian, it can take an arbitrary number of iterations to converge [13]. Furthermore, and key to our main contribution, is the fact that CG requires a well-conditioned matrix to converge. If the matrix is not well-conditioned, numerical errors can accumulate, and the approximation can become arbitrarily bad. As shown in Figure 2, ill conditioning can arise from the inherent non-convexity of deep neural networks and from overparameterization [30, 34], which can cause the IHVP error to spike when using CG.

The Nyström Method. Proposed by Hataya and Yamada [19], the Nyström method produces a low-rank approximation of the IHVP.

Assume a p-dimensional positive (semi)definite Hessian denoted by H and a low-rank approximation of H denoted by H_q, for $q \ll p$. By selecting a set of Q indices of size q at random, we obtain the Nyström approximation $H_q = H_{[:,Q]} H_{[Q,Q]}^{\dagger} H_{[:,Q]}^{\top}$, where $H_{[:,Q]} \in \mathbb{R}^{p \times q}$ is a matrix of select columns of H at indices Q; $H_{[Q,Q]} \in \mathbb{R}^{q \times q}$ is a matrix of select rows of $H_{[:,Q]}$ at indices Q; and $H_{[Q,Q]}^{\dagger} = \boldsymbol{U}\boldsymbol{\Lambda}^{-1}\boldsymbol{U}^{\top}$ is the Moore-Penrose pseudoinverse of $H_{[Q,Q]}$, where $\boldsymbol{U}$ are the eigenvectors and $\boldsymbol{\Lambda}$ are the eigenvalues of $H_{[Q,Q]}$. For some small regularization constant $\alpha > 0$, we define the α-regularized IHVP as $(H_q + \alpha \boldsymbol{I})^{-1}$, where $\boldsymbol{I}$ is the p-dimensional identity matrix.[3] This α-regularized IHVP decomposes as follows: $(H_q + \alpha \boldsymbol{I})^{-1} = (H_{[:,Q]} H_{[Q,Q]}^{\dagger} H_{[:,Q]}^{\top} + \alpha \boldsymbol{I})^{-1}$. Next, applying the Woodbury matrix identity allows us to obtain:

$$(H_q + \alpha \boldsymbol{I})^{-1} = \frac{1}{\alpha}\boldsymbol{I} - \frac{1}{\alpha^2} H_{[:,Q]} \left(H_{[Q:Q]} + \frac{1}{\alpha} H_{[:,Q]}^{\top} H_{[:,Q]} \right)^{-1} H_{[:,Q]}^{\top}.$$

This final expression allows us to approximate the IHVP $\boldsymbol{v}$ by $\widehat{\boldsymbol{v}} = (H_q + \alpha \boldsymbol{I})^{-1} \nabla_{\boldsymbol{y}} f(\boldsymbol{x}, \hat{\boldsymbol{y}})$. The regularization parameter α plays a crucial role in balancing numerical stability and the fidelity of curvature information. As discussed by Vicol et al. [42], for an eigenvalue λ of the Hessian H, the effect of regularization is to modify the inverse eigenvalue as $(\lambda + \alpha)^{-1}$. When $\lambda \gg \alpha$, this term behaves as λ^{-1}, preserving curvature information. Conversely, when $\alpha \gg \lambda$, the approximation becomes insensitive to low-curvature directions, effectively ignoring them. In practice, α is selected empirically. One practical advantage of the Nyström method is its ability to operate effectively with smaller α. For example, we use $\alpha = 50$, compared to $\alpha = 500$ in Zheng et al. [45] for continuous control RL, and $\alpha = 10,000$ in Fiez et al. [12] for a GAN-based bilevel problem, both of which use CG. Notably, Lorraine et al. [27] observed that using the identity matrix, which is equivalent to choosing $\alpha \gg L$ and then rescaling the IHVP, matched the performance of approximate IHVPs computed via CG. We find that the Nyström method offers an attractive balance, where regularization remains modest without sacrificing stability. Algorithm 1 presents our bilevel optimization algorithm with Nyström hypergradients.

Algorithm 1 BLO with Nyström Hypergradients

input $K_{\boldsymbol{x}}, K_{\boldsymbol{y}}, \boldsymbol{x}^{(0)}, \boldsymbol{y}^{(0)}$, and learning rates $\eta_{\boldsymbol{x}}, \eta_{\boldsymbol{y}}$
output $\left(\{\boldsymbol{x}^{(k)}\}_{k=0}^{K_{\boldsymbol{x}}-1}, \{\boldsymbol{y}^{(k)}\}_{k=0}^{K_{\boldsymbol{y}}-1} \right)$
  for $k = 0, \ldots, K_{\boldsymbol{x}} - 1$ **do**
    $\boldsymbol{z}^{(0)} \leftarrow \boldsymbol{y}^{(k-1)}$ if $k > 0$ else $\boldsymbol{y}^{(0)}$ {Warm starts}
    for $d = 0, \ldots, K_{\boldsymbol{y}} - 1$ **do**
      $\boldsymbol{z}^{(d+1)} \leftarrow \boldsymbol{z}^{(d)} - \eta_{\boldsymbol{y}} \nabla_{\boldsymbol{y}} g_{\boldsymbol{x}^{(k)}}(\boldsymbol{z}^{(d)})$
    end for
    $\boldsymbol{y}^{(k)} \leftarrow \boldsymbol{z}^{(K_{\boldsymbol{y}})}$
    calculate $\widehat{\boldsymbol{v}}$ using the Nyström method
    $\nabla \hat{\Phi}(\boldsymbol{x}^{(k)}) \leftarrow \nabla_{\boldsymbol{x}} f(\boldsymbol{x}^{(k)}, \boldsymbol{y}^{(k)}) - \nabla^2_{\boldsymbol{x}\boldsymbol{y}} g_{\boldsymbol{x}^{(k)}}(\boldsymbol{y}^{(k)}) \widehat{\boldsymbol{v}}$
    $\boldsymbol{x}^{(k+1)} \leftarrow \boldsymbol{x}^{(k)} - \eta_{\boldsymbol{x}} \nabla \hat{\Phi}(\boldsymbol{x}^{(k)})$
  end for

[3] Incorporating the small perturbation $\alpha \boldsymbol{I}$ into this IHVP is equivalent to optimizing a proximal regularization of the inner objective: i.e., $g_{\boldsymbol{x}}(\boldsymbol{y}) + \alpha/2 \|\boldsymbol{y} - \arg\min_{\boldsymbol{y}} g_{\boldsymbol{x}}(\boldsymbol{y})\|$ [27].

4 CONVERGENCE OF BLO-NYSTRÖM

We make several convexity and smoothness assumptions, which are standard in the BLO literature [16, 23]. Our first assumption ensures that the inner optimization admits a unique solution $\boldsymbol{y}^*$ for a given $\boldsymbol{x}$, and thus implies invertibility of the Hessian.

Assumption 4.1 (Strong convexity)**.** The inner objective is μ-strongly convex in the inner variable, i.e., $\boldsymbol{y} \mapsto g_{\boldsymbol{x}}(\boldsymbol{y})$ is μ-strongly-convex. This assumption ensures the Hessian $\nabla^2_{\boldsymbol{y}\boldsymbol{y}} g_{\boldsymbol{x}}$ is full rank.

The next three assumptions pertain to the smoothness of both the inner and outer objective functions, meaning they ensure that the first and second-order gradients are well behaved. In particular, they imply the condition number $\kappa = L/\mu$ for the inner objective $g_{\boldsymbol{x}}$. Moreover, Assumptions 4.1 and 4.2 are sufficient for the inverse function theorem to apply, because together they imply $\boldsymbol{y}^*(\boldsymbol{x})$ is a first-order stationary point.

Assumption 4.2 (Lipschitz continuity)**.** Both the inner and outer objectives, f and $g_{\boldsymbol{x}}$, have L-Lipschitz bounded gradients: i.e., for all $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^n$, and $\boldsymbol{y}, \boldsymbol{y}' \in \mathbb{R}^m$,

$$\|\nabla f(\boldsymbol{x}, \boldsymbol{y}) - \nabla f(\boldsymbol{x}', \boldsymbol{y}')\| \leq L \|(\boldsymbol{x}, \boldsymbol{y}) - (\boldsymbol{x}', \boldsymbol{y}')\|,$$
$$\|\nabla g_{\boldsymbol{x}}(\boldsymbol{y}) - \nabla g_{\boldsymbol{x}'}(\boldsymbol{y}')\| \leq L \|(\boldsymbol{x}, \boldsymbol{y}) - (\boldsymbol{x}', \boldsymbol{y}')\|.$$

Assumption 4.3 (Lipschitz smoothness)**.** The inner objective $g_{\boldsymbol{x}}$ is also ρ-Lipschitz smooth, i.e., its second derivatives are bounded: for all $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^n$, and $\boldsymbol{y}, \boldsymbol{y}' \in \mathbb{R}^m$,

$$\|\nabla_{\boldsymbol{x}\boldsymbol{y}} g_{\boldsymbol{x}}(\boldsymbol{y}) - \nabla_{\boldsymbol{x}\boldsymbol{y}} g_{\boldsymbol{x}'}(\boldsymbol{y}')\| \leq \rho \|(\boldsymbol{x}, \boldsymbol{y}) - (\boldsymbol{x}', \boldsymbol{y}')\|,$$
$$\|\nabla^2_{\boldsymbol{y}\boldsymbol{y}} g_{\boldsymbol{x}}(\boldsymbol{y}) - \nabla^2_{\boldsymbol{y}\boldsymbol{y}} g_{\boldsymbol{x}'}(\boldsymbol{y}')\| \leq \rho \|(\boldsymbol{x}, \boldsymbol{y}) - (\boldsymbol{x}', \boldsymbol{y}')\|.$$

Assumption 4.4. There exists a constant $M \in \mathbb{R}^{++}$ s.t. for all $\boldsymbol{x} \in \mathbb{R}^n$, $\|\nabla_{\boldsymbol{y}} f(\boldsymbol{x}, \boldsymbol{y})\| \leq M$.

Ghadimi and Wang [16] were the first to provide convergence results for non-convex Φ and strongly-convex $g_{\boldsymbol{x}}$. Their analysis was sharpened by Ji et al. [23], whose analysis also relied on conjugate gradient to compute the IHVP. Our analysis extends the literature by considering the Nyström method, rather than CG, in this non-convex strongly-convex case. Specifically, we characterize the irreducible error incurred by the random sampling of columns.

Our first result[4] justifies the use of warm-starts in Algorithm 1: if the value of the outer variable changes slightly, then the corresponding inner solutions do not stray too far from one another either:

Lemma 4.5 (Warm Starts)**.** *Under Assumption 4.1 and 4.2, for all $\boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathbb{R}^n$, $\boldsymbol{y} \in \arg\min_{\boldsymbol{z}} g_{\boldsymbol{x}}(\boldsymbol{z})$, and $\tilde{\boldsymbol{y}} \in \arg\min_{\boldsymbol{z}} g_{\tilde{\boldsymbol{x}}}(\boldsymbol{z})$, $\|\boldsymbol{y} - \tilde{\boldsymbol{y}}\| \leq L/\mu \|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|$.*

Theorem 4.6 (Convergence)**.** *Given μ, L, ρ, M as defined in Assumptions 4.2 through 4.4, a Nyström regularization constant α, and a hypergradient smoothness parameter $L_\Phi \in O(\kappa^3)$. Choose $\eta_{\boldsymbol{y}} = 1/L$, $\eta_{\boldsymbol{x}} = 1/8L_\Phi$, $\delta \in (0, 1]$, $K_{\boldsymbol{y}} \in O(\kappa)$ inner iterations, and $K \doteq K_{\boldsymbol{x}} \in \mathbb{N}$ outer iterations. Then, with probability at least $1 - O(\delta)$, the outputs of Algorithm 1 satisfy $\frac{1}{K} \sum_{k=0}^{K-1} \left\| \Phi(\boldsymbol{x}^{(k)}) \right\|^2$*

$$\leq \frac{1}{K} \left(64 L_\Phi (\Phi(\boldsymbol{x}^{(0)}) - \inf_{\boldsymbol{x}} \Phi(\boldsymbol{x})) + 5 \left\| \boldsymbol{y}^{0(0)} - \boldsymbol{y}^0 \right\|^2 \right) + 10 L^2 \Psi^2,$$

[4] The proof of all claims in this paper can be found in here.

where Ψ is the irreducible error incurred by the Nyström approximation. In order to obtain an ϵ-stationary point, for each IHVP calculation, sample $q \geq O\left(\frac{p\log(1/\delta)}{(\epsilon/2)^4}\right)$ columns of $\nabla^2_{\boldsymbol{yy}} g_{\boldsymbol{x}}$, where the indices chosen are sampled proportionally to the square of the values of the diagonal elements of $\nabla^2_{\boldsymbol{yy}} g_{\boldsymbol{x}}$. Then, the requisite number of gradient computations are $O(\frac{\kappa^3}{\epsilon/2})$ for $\nabla_{\boldsymbol{x}} f$; $O(\frac{\kappa^4}{\epsilon/2})$ for $\nabla_{\boldsymbol{y}} g_{\boldsymbol{x}}$; $O(\frac{\kappa^3}{\epsilon/2})$ for the IHVP $\widehat{\boldsymbol{v}}$; and $O(\frac{\kappa^3}{\epsilon/2})$ for the Jacobian vector product $\nabla^2_{\boldsymbol{xy}} g_{\boldsymbol{x}} \widehat{\boldsymbol{v}}$.

Theorem 4.6 states that Algorithm 1 can be used to solve a non-convex strongly-convex BLO. By carefully selecting the learning rates $(\eta_{\boldsymbol{x}}, \eta_{\boldsymbol{y}})$ and the number of inner and outer iterations (D, K), and by column sampling according to eigenvalue magnitude, we derive an explicit convergence rate to an ϵ-bilevel stationary point of Equation (1).

5 BILEVEL POLICY OPTIMIZATION (BLPO)

Vanilla actor-critic algorithms run gradient descent for the actor and the critic simultaneously, updating $\boldsymbol{\theta}$ and $\boldsymbol{\omega}$ much like a better-response dynamic:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}} \hat{J}(\boldsymbol{\theta}, \boldsymbol{\omega}) \qquad \boldsymbol{\omega} \leftarrow \boldsymbol{\omega} - \eta_{\boldsymbol{\omega}} \nabla_{\boldsymbol{\omega}} \hat{L}(\boldsymbol{\theta}, \boldsymbol{\omega}).$$

(Here, $\eta_{\boldsymbol{\theta}}$ and $\eta_{\boldsymbol{\omega}}$ are the actor's and critic's learning rates, respectively.) Our approach, as already described, is *not* to update these two sets of parameters simultaneously, but rather to solve the following BLO by descending on the hypergradient for the actor, and the gradient for critic:

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^n} -J(\boldsymbol{\theta}, \boldsymbol{\omega}^*(\boldsymbol{\theta})) \quad \text{s.t.} \quad \boldsymbol{\omega}^*(\boldsymbol{\theta}) \in \arg\min_{\boldsymbol{\omega} \in \mathbb{R}^m} L_{\boldsymbol{\theta}}(\boldsymbol{\omega}).$$

The main challenge that arises with our approach is computing the actor's hypergradient, namely:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, \boldsymbol{\omega}^*(\boldsymbol{\theta})) = \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, \boldsymbol{\omega}) + (\nabla_{\boldsymbol{\theta}} (\boldsymbol{\omega}^*(\boldsymbol{\theta})) \nabla_{\boldsymbol{\omega}} J(\boldsymbol{\theta}, \boldsymbol{\omega}), \tag{8}$$

$$= \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, \boldsymbol{\omega}) - \nabla^2_{\boldsymbol{\theta\omega}} L_{\boldsymbol{\theta}}(\boldsymbol{\omega}) (\nabla^2_{\boldsymbol{\omega}} L_{\boldsymbol{\theta}}(\boldsymbol{\omega}))^{-1} \nabla_{\boldsymbol{\omega}} J(\boldsymbol{\theta}, \boldsymbol{\omega}). \tag{9}$$

There are four gradients involved in this formula. Three of them, $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, \boldsymbol{\omega})$, $\nabla_{\boldsymbol{\omega}} J(\boldsymbol{\theta}, \boldsymbol{\omega})$, and $\nabla_{\boldsymbol{\omega}} L_{\boldsymbol{\theta}}(\boldsymbol{\omega})$, can be easily obtained through autodifferentiation. The only potential difficulty lies in computing $\nabla_{\boldsymbol{\theta}} L_{\boldsymbol{\theta}}(\boldsymbol{\omega})$, since L is parameterized by $\boldsymbol{\theta}$, and as such, is not directly a function of $\boldsymbol{\theta}$. In the next theorem, we invoke the policy gradient theorem to derive an analytical expression for $\nabla_{\boldsymbol{\theta}} L_{\boldsymbol{\theta}}(\boldsymbol{\omega})$, which we can use along with $\nabla_{\boldsymbol{\omega}} L_{\boldsymbol{\theta}}(\boldsymbol{\omega})$ to compute the mixed partial $\nabla^2_{\boldsymbol{\theta\omega}} L_{\boldsymbol{\theta}}(\boldsymbol{\omega})$.

Theorem 5.1. *The gradient of the critic's objective function with respect to the actor's parameters, $\nabla_{\boldsymbol{\theta}} L_{\boldsymbol{\theta}}(\boldsymbol{\omega})$, is:*

$$\mathbb{E}_{\boldsymbol{s} \sim \nu_l^{\boldsymbol{\pi_\theta}}} \left[\left(V^{\boldsymbol{\pi_\theta}}(\boldsymbol{s}) - V_{\boldsymbol{\omega}}(\boldsymbol{s}) \right) \left(\mathbb{E}_{\substack{\boldsymbol{s}' \sim \nu_{\boldsymbol{s}}^{\boldsymbol{\pi_\theta}} \\ \boldsymbol{a} \sim \boldsymbol{\pi_\theta}(\boldsymbol{s}')}} \left[\nabla_{\boldsymbol{\theta}} \log \boldsymbol{\pi_\theta}(\boldsymbol{a} \mid \boldsymbol{s}') \, Q^{\boldsymbol{\pi_\theta}}(\boldsymbol{s}', \boldsymbol{a}) \right] \right) \right], \tag{10}$$

where $\boldsymbol{\pi_\theta}(\boldsymbol{a} \mid \boldsymbol{s})$ denotes the probability of taking action $\boldsymbol{a}$ under policy $\boldsymbol{\pi}$ at state $\boldsymbol{s}$.

Algorithm 2 Bilevel Policy Optimization (BLPO)

input number of outer (respectively, inner) iterations $K_{\boldsymbol{\theta}}$ (respectively, $K_{\boldsymbol{\omega}}$), length of rollouts T, number of epochs n, number of mini-batches m, learning rates $\eta_{\boldsymbol{\theta}}$ and $\eta_{\boldsymbol{\omega}}$
output the actor and critic networks, $\boldsymbol{\pi_\theta}$ and $V_{\boldsymbol{\omega}}$

```
Initialize θ^(0), ω^(0)
for k = 0, 1, ..., K_θ − 1 do
    Collect rollouts of length T using policy π_{θ^(k)}
    Use the value function V_{ω^(k)} to compute A^(k)
    for all n epochs do
        for all m mini-batches do
            z^(0) ← ω^(k−1) if k > 0 else ω^(0)
            for l = 0, 1, ..., K_ω − 1 do
                ω^(l+1) ← ω^(l) − η_ω (1/(T/m)) ∇_ω Σ_{t=0}^{T/m} (1/2) ( V_{ω^(k)}(s_t) + A^(k)(s_t) − V_{ω^(l)}(s_t) )^2
            end for
            ω^(k) ← z^(K_ω)
            Estimate the IHVP via the Nyström method: v̂_AC ← (∇^2_ω L̂_θ(ω^(k)))^{-1} ∇_ω ĵ(θ^(k), ω^(k)))
            Calculate the hypergradient using Equation 12 for the direct gradient and Corollary 5.4 for the implicit gradient: ∇_θ ĵ^(k) ← ∇_θ ĵ(θ^(k), ω^(k)) − ∇_{θω} L̂(θ^(k), ω^(k)) v̂_AC
            Update the actor by following the hypergradient: θ^(k+1) ← θ^(k) + η_θ ∇_θ ĵ^(k)
        end for
    end for
end for
```

Notably, we use the Nyström method to approximate the IHVP $\widehat{\boldsymbol{v}}_{AC} = (\nabla^2_{\boldsymbol{\omega}} L_{\boldsymbol{\theta}}(\boldsymbol{\omega}))^{-1} \nabla_{\boldsymbol{\omega}} J(\boldsymbol{\theta}, \boldsymbol{\omega})$. We then compute the Jacobian vector product as $\nabla_{\boldsymbol{\theta\omega}} L(\boldsymbol{\theta}, \boldsymbol{\omega}) \widehat{\boldsymbol{v}}_{AC}$, which we subtract from $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, \boldsymbol{\omega})$ to obtain the hypergradient. BLPO is thus a special case of Algorithm 1 (BLO with Nyström gradients) in which $f = J$ and $g = L$.

Next, we we present assumptions under which the critic's objective is strongly convex.

Theorem 5.2. *Given a policy $\boldsymbol{\pi_\theta}$, assume $\gamma < 1$, and suppose the value function $V_{\boldsymbol{\omega}}$ is linearly parameterized, i.e., $V_{\boldsymbol{\omega}}(\boldsymbol{s}) = \boldsymbol{\omega}^\top \boldsymbol{\zeta}(\boldsymbol{s})$, for all $\boldsymbol{s} \in \mathcal{S}$, where $\boldsymbol{\zeta} : \mathcal{S} \to \mathbb{R}^m$ is a feature map. Define $\Delta\boldsymbol{\zeta\gamma}(\boldsymbol{s}, \boldsymbol{s}') = \boldsymbol{\zeta}(\boldsymbol{s}) - \gamma \boldsymbol{\zeta}(\boldsymbol{s}')$, for all $\boldsymbol{s}, \boldsymbol{s}' \in \mathcal{S}$ where $\boldsymbol{s}$ is the current state and $\boldsymbol{s}'$ is the successor state under $\boldsymbol{\pi_\theta}$. Then, assuming the covariance matrix $\mathbb{E}_{\substack{\boldsymbol{s} \sim \nu_l^{\boldsymbol{\pi_\theta}} \\ \boldsymbol{s}' \sim P[\cdot \mid \boldsymbol{s}, \boldsymbol{\pi_\theta}(\boldsymbol{s})]}} \left[\Delta\boldsymbol{\zeta\gamma}(\boldsymbol{s}, \boldsymbol{s}') \Delta\boldsymbol{\zeta\gamma}(\boldsymbol{s}, \boldsymbol{s}')^\top \right]$ is positive definite, $L_{\boldsymbol{\theta}}(\boldsymbol{\omega})$ is μ-strongly-convex in $\boldsymbol{\omega}$, where the strong-convexity constant μ is the smallest eigenvalue of the covariance matrix.*

We have thus established conditions under which our actor-critic formulation, Equation (4), is indeed a non-convex strongly-convex BLO so that Theorem 4.6 applies. As a result, we can design a reinforcement learning algorithm in the style of Algorithm 1, which converges in polynomial time to an ϵ-bilevel stationary point of Equation (4), with high probability.

Advantage Functions and Estimators. In practice, the actor and the critic's objective functions must be estimated. To reduce the

variance in these estimates, they are often computed using advantage functions. At iteration k, when the critic's learned value function is $V_{\omega^{(k)}}$, we define the advantage function as $A^{(k)}(s,a) = Q^{\pi_\theta}(s,a) - V_{\omega^{(k)}}(s)$. Now, since $V^{\pi_\theta}(s) = Q^{\pi_\theta}(s,\pi_\theta(s))$,

$$L_\theta^{(k)}(\omega) = \mathop{\mathbb{E}}_{\substack{s\sim\nu_t^{\pi_\theta}\\ a\sim\pi_\theta(s)}}\left[\frac{1}{2}\left(\underbrace{V_{\omega^{(k)}}(s) + A^{(k)}(s,a)}_{\text{target}} - V_\omega(s)\right)^2\right]. \tag{11}$$

Using this advantage function, we obtain the following corollary of Theorem 5.1.

Corollary 5.3. *At iteration k, the gradient of the critic's objective function with respect to the actor's parameters is given by:*

$$\nabla_\theta L_\theta^{(k)}(\omega^{(k)}) = \mathop{\mathbb{E}}_{s\sim\nu_t^{\pi_\theta}}\left[\left(V_{\omega^{(k)}}(s) + A^{(k)}(s,a) - V_\omega(s)\right)\times \left(\mathop{\mathbb{E}}_{\substack{s'\sim\nu_s^{\pi_\theta}\\ a\sim\pi_\theta(s')}}\left[\nabla_\theta \log \pi_\theta(a\mid s')\, A^{(k)}(s,a)\right]\right)\right]. \tag{12}$$

Using a straightforward estimator of this gradient, we obtain a further corollary:

Corollary 5.4. *Given M sampled trajectories of length T $\{(s_0^1, a_0^1, r_0^1, s_1^1, \ldots, r_T^0), \ldots, (s_0^M, a_0^M, r_0^M, s_1^M, \ldots, r_T^M)\}$, at iteration k, the gradient of the critic's objective function can be estimated by:* $\nabla_\theta \hat{L}_\theta^{(k)}(\omega) = \frac{1}{MT}\sum_{m=1}^{M}\sum_{t=0}^{T-1}\left(V_{\omega^{(k)}}(s_t^m) + A^{(k)}(s_t^m, a_t^m) - V_\omega(s_t^m)\right)\times \left(\frac{1}{T-t}\sum_{k=0}^{T-1-t}\nabla_\theta \log \pi_\theta\left(a_{t+k}^m \mid s_{t+k}^m\right) A^{(k)}\left(s_{t+k}^m, a_{t+k}^m\right)\right)$.

We can likewise restate the (direct) gradient of the actor's objective using an advantage function and an estimator. By the policy gradient theorem [41], $\nabla_\theta J(\theta,\omega) = \mathbb{E}_{\substack{s\sim\nu_t^{\pi_\theta}\\ a\sim\pi_\theta(s)}}\left[\nabla_\theta \log \pi_\theta(a\mid s)\, Q^{\pi_\theta}(s,a)\right]$. This gradient can be stated equivalently in terms of $Q^{\pi_\theta}(s,a)$ the TD-Residual $r_t + V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t)$, or an advantage function $A(s,a)$ [37]. At iteration k, we employ the advantage function $A^{(k)}(s,a)$ so that $\nabla_\theta J^{(k)}(\theta,\omega) = \mathbb{E}_{\substack{s\sim\nu_t^{\pi_\theta}\\ a\sim\pi_\theta(s)}}\left[\nabla_\theta \log \pi_\theta(a\mid s)\, A^{(k)}(s,a)\right]$. We thus arrive at the following gradient estimator:

$$\nabla_\theta \hat{J}^{(k)}(\theta,\omega) \doteq \frac{1}{MT}\sum_{m=1}^{M}\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t^m \mid s_t^m)\, A^{(k)}\left(s_t^m, a_t^m\right). \tag{13}$$

The estimator $\nabla_\theta \hat{L}_\theta^{(k)}(\omega)$ defined in Corollary 5.4 is only unbiased if we take care to sample the inner $A^{(k)}$ from a new trajectory originating at each $s_{t+k} \sim \nu_{s_t}^{\pi_\theta}$. For reasons of data efficiency, our estimator is biased; that is, we use the same trajectories to compute the inner and the outer $A^{(k)}$s. Our empirical results suggest that this biased estimator is sufficient for strong performance. We leave to future work a more thorough investigation of methods to mitigate this bias while preserving data efficiency.

Pseudocode for BLPO with an advantage function and these estimators appears in Algorithm 2.

6 EXPERIMENTS

We use as our baseline the PPO algorithm from PureJaxRL [28], with separate policy and value networks of fully-connected MLPs, each with two hidden layers of 64 units. Our implementation of BLPO—BLPO Nyström hereafter, specified in Algorithm 2—extends this implementation of PPO with a Nyström-based hypergradient computation and nested critic updates. We also test another variant, BLPO-CG, which uses CG to estimate the hypergradient instead of Nyström's method. In all cases, we calculate advantages using the Generalized Advantage Estimator (GAE) [37].

We conduct experiments on discrete control tasks from Gymnax [24] and continuous control tasks[5] from Brax [14]. All experiments were run using the default PureJaxRL PPO hyperparameters on a single RTX 3090 GPU.[6]

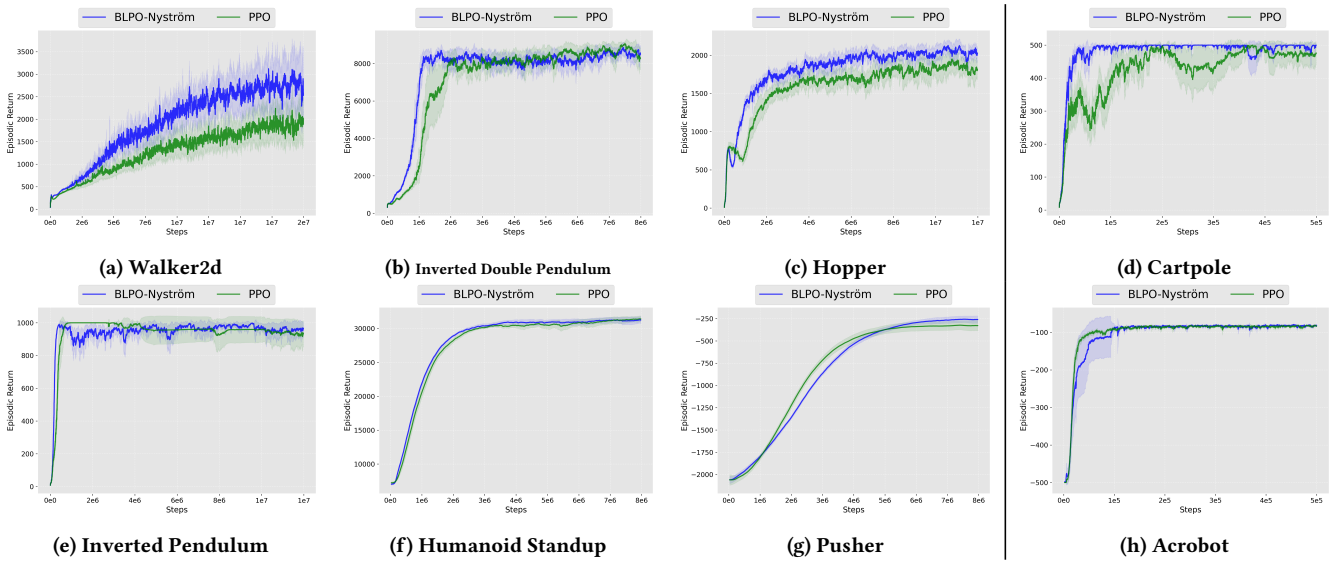
Heuristic. The eigenspectrum of the Hessian has been found to have low empirical rank, with the bulk of the eigenspectrum clustered around zero with several large outliers [17, 35]. As a result, we clip the IHVP to ensure that the inversion does not lead to an arbitrarily large gradient step, which would destabilize training.

Performance. In Figure 3 we report 95% confidence intervals around the average episodic returns across 15 seeds. We find that BLPO outperforms PPO in most of our experiments and matches PPO in the rest. In Walker2D and Hopper, BLPO learns a significantly better policy than PPO. Moreover, BLPO converges with fewer samples than PPO in both Pendulum tasks. We also compare BLPO, hereafter BLPO-Nyström, to a variant that uses CG instead of Nyström to estimate the hypergradient. We find that BLPO-Nyström either outperforms or matches BLPO-CG in terms of episodic returns across all tasks. We present these ablations, as well as nesting without any hypergradients here.

Performance parity on simpler tasks (e.g., Inverted Pendulum, Acrobot) reflects a ceiling effect: PPO already finds near-optimal solutions rapidly, rendering hypergradient information unnecessary. Similarly, while Humanoid standup involves 376 parameters, its optimal configuration is a static hold with relatively non-chaotic dynamics. BLPO's advantage emerges in complex, chaotic (Inverted Double Pendulum), and open-ended environments without a performance ceiling (Walker2d and Hopper), where actor-critic coupling is strong. In these settings, actor updates drastically shift the value landscape, causing PPO's performance, which mimics simultaneous updates, to degrade. BLPO's formulation, which allows the actor to anticipate the critic's response via the hypergradient, yields better policy updates. In summary, BLPO should be the preferred choice when simulations are expensive, so that the reduced number of environment steps outweighs the per-step gradient overhead.

Runtime Analysis. Algorithms that rely on second-order information can run significantly slower than their first-order counterparts. On the simple control tasks, we find that BLPO-Nyström is indeed slower than PPO. In the more complex environments, however, the gap between all algorithms is smaller, as most of the computation time is spent on the simulator. With regards to BLPO-CG, although

[5] In continuous control, we use a multivariate Gaussian to represent the policy network.
[6] See the arXiv version of this work for additional experimental details. Our code is available online at https://github.com/Arnie-He/BLPO.



(a) Walker2d (b) Inverted Double Pendulum (c) Hopper (d) Cartpole

(e) Inverted Pendulum (f) Humanoid Standup (g) Pusher (h) Acrobot

Figure 3: In control tasks, BLPO either outperforms PPO or performs comparably. The left three columns show continuous control experiments; the right column shows discrete control experiments.

the calculation of an IHVP takes approximately 1.5 times a normal gradient step [8, 32], BLPO-CG is slower than BLPO-Nyström on all tasks, as poor conditioning necessitates more CG iterations.

We also experimented with BLPO variants using pre-conditioned conjugate gradient.[7] Interestingly, the Nyström method can itself be used as a preconditioner for conjugate gradient [13]. We find that doing so improves the convergence time over unconditioned CG. We also use the Gauss-Newton matrix, a positive-semi-definite approximation of the Hessian; however, the additional overhead required to compute the Gauss-Newton matrix makes this method impractical. Our results, shown in Figure 4, demonstrate that the Nyström method achieves the best balance of speed and performance.

7 CONCLUSION

In this paper, we developed BLO–Nyström, an algorithm that solves a BLO using Nyström's method to compute the hypergradient, and we established convergence rates for our algorithm to a local stationary point of a non-convex strongly-convex BLO. Our main contribution, BLPO–Nyström applies these ideas to a BLO formulation of actor critic, which nests the critic's optimization inside the actor's, and follows the hypergradient of the actor. A BLO formulation of AC is akin to a Stackelberg game, where it can be advantageous to be the leader: a Stackelberg leader (in our application, the actor) often attains greater rewards as the first-mover than she would in a simultaneous-move version of the same game [4].

[7]Empirically, we found 50 iterations maximized the performance of CG on in RL environments in our study, and we thus set the maximum number of CG iterations to 50.

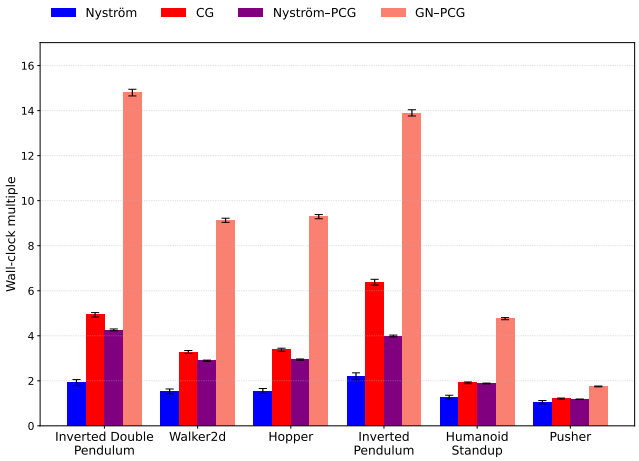


Figure 4: Wall-clock runtime of BLPO variants as a multiple of PPO's runtime (i.e., PPO corresponds to 1×).

While the AC-BLO formulation is not new, BLPO leverages the Nyström method to compute a low-rank approximation of the inverse Hessian-vector product, stabilizing the hypergradient computation. Experimentally, we found that BLPO consistently achieved stronger performance on a suite of discrete and continuous control tasks than PPO, the standard baseline. Future research should evaluate BLPO on a wider variety of tasks, such as Atari. Future research is also needed to better evaluate its scalability to larger environments. Successful scaling would render it applicable to other larger BLO, such as RLHF, meta-learning, and hyperparameter optimization. This work takes a positive step towards paving the way for more efficient and scalable second-order optimization in deep learning.

ACKNOWLEDGMENTS

This work was supported by the Office of Naval Research (ONR) grants N00014-22-1-2592 and N00014-24-1-2657 (institutional IDs GR5291318 and GR5250124, respectively). We would also like to thank Jacob Makar-Limanov for investigating the low-rank structure of the Hessian in our experimental setup, and for his invaluable feedback on its implications for stabilizing the IHVP.

REFERENCES

[1] Sezin Afcsar, Luce Brotcorne, Patrice Marcotte, and Gilles Savard. 2016. Achieving an optimal trade-off between revenue and energy peak within a smart grid environment. *Renewable Energy* 91 (2016), 293–301.

[2] Jonathan F Bard, John Plummer, and Jean Claude Sourie. 1998. Determining tax credits for converting nonfood crops to biofuels: an application of bilevel programming. *Multilevel Optimization: Algorithms and Applications* (1998), 23–50.

[3] Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. 1983. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics* SMC-13, 5 (1983), 834–846. https://doi.org/10.1109/TSMC.1983.6313077

[4] Avrim Blum, Nika Haghtalab, MohammadTaghi Hajiaghayi, and Saeed Seddighin. 2019. Computing stackelberg equilibria of large general-sum games. In *Algorithmic Game Theory: 12th International Symposium, SAGT 2019, Athens, Greece, September 30–October 3, 2019, Proceedings 12*. Springer, 168–182.

[5] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. 2018. *JAX: composable transformations of Python+NumPy programs*. http://github.com/jax-ml/jax

[6] Dotan Di Castro and Ron Meir. 2010. A convergent online single time scale actor critic algorithm. *The Journal of Machine Learning Research* 11 (2010), 367–410.

[7] Souradip Chakraborty, Amrit S. Bedi, Alec Koppel, Huazheng Wang, Dinesh Manocha, Mengdi Wang, and Furong Huang. 2024. PARL: A Unified Framework for Policy Alignment in Reinforcement Learning from Human Feedback. In *ICLR*. https://openreview.net/forum?id=ByR3NdDSZB

[8] Mathieu Dagréou, Pierre Ablin, Samuel Vaiter, and Thomas Moreau. 2024. How to compute Hessian-vector products?. In *ICLR Blogposts 2024* (May 7, 2024). https://iclr-blogposts.github.io/2024/blog/bench-hvp/ https://iclr-blogposts.github.io/2024/blog/bench-hvp/.

[9] Stephan Dempe. 2018. *Bilevel optimization: theory, algorithms and applications*. Vol. 3. TU Bergakademie Freiberg, Fakultät für Mathematik und Informatik Freiberg ....

[10] Mucong Ding, Souradip Chakraborty, Vibhu Agrawal, Zora Che, Alec Koppel, Mengdi Wang, Amrit Bedi, and Furong Huang. 2024. Sail: Self-improving efficient online alignment of large language models. *arXiv preprint arXiv:2406.15567* (2024).

[11] Petros Drineas, Michael W Mahoney, and Nello Cristianini. 2005. On the Nyström Method for Approximating a Gram Matrix for Improved Kernel-Based Learning. *journal of machine learning research* 6, 12 (2005).

[12] Tanner Fiez, Benjamin Chasnov, and Lillian Ratliff. 2020. Implicit Learning Dynamics in Stackelberg Games: Equilibria Characterization, Convergence Analysis, and Empirical Study. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 3133–3144. https://proceedings.mlr.press/v119/fiez20a.html ISSN: 2640-3498.

[13] Zachary Frangella, Joel A Tropp, and Madeleine Udell. 2023. Randomized nyström preconditioning. *SIAM J. Matrix Anal. Appl.* 44, 2 (2023), 718–752.

[14] C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. 2021. *Brax - A Differentiable Physics Engine for Large Scale Rigid Body Simulation*. http://github.com/google/brax

[15] Scott Fujimoto, Herke Hoof, and David Meger. 2018. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*. PMLR, 1587–1596.

[16] Saeed Ghadimi and Mengdi Wang. 2018. Approximation methods for bilevel programming. *arXiv preprint arXiv:1802.02246* (2018).

[17] Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. 2019. An investigation into neural net optimization via hessian eigenvalue density. In *International Conference on Machine Learning*. PMLR, 2232–2241.

[18] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*. PMLR, 1861–1870.

[19] Ryuichiro Hataya and Makoto Yamada. 2023. Nyström method for accurate and scalable implicit differentiation. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 4643–4654.

[20] Magnus Rudolph Hestenes, Eduard Stiefel, et al. 1952. *Methods of conjugate gradients for solving linear systems*. Vol. 49. NBS Washington, DC.

[21] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems* 30 (2017).

[22] Mingyi Hong, Hoi-To Wai, Zhaoran Wang, and Zhuoran Yang. 2023. A Two-Timescale Stochastic Algorithm Framework for Bilevel Optimization: Complexity Analysis and Application to Actor-Critic. *SIAM Journal on Optimization* 33, 1 (2023), 147–180. https://doi.org/10.1137/20M1387341 arXiv:https://doi.org/10.1137/20M1387341

[23] Kaiyi Ji, Junjie Yang, and Yingbin Liang. 2021. Bilevel optimization: Convergence analysis and enhanced design. In *International conference on machine learning*. PMLR, 4882–4892.

[24] Robert Tjarko Lange. 2022. *gymnax: A JAX-based Reinforcement Learning Environment Library*. http://github.com/RobertTLange/gymnax

[25] William Layton and Myron Sussman. 2020. *Numerical linear algebra*. World Scientific.

[26] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2019. Continuous control with deep reinforcement learning. arXiv:1509.02971 [cs.LG] https://arxiv.org/abs/1509.02971

[27] Jonathan Lorraine, Paul Vicol, and David Duvenaud. 2020. Optimizing millions of hyperparameters by implicit differentiation. In *International conference on artificial intelligence and statistics*. PMLR, 1540–1552.

[28] Chris Lu, Jakub Kuba, Alistair Letcher, Luke Metz, Christian Schroeder de Witt, and Jakob Foerster. 2022. Discovered policy optimisation. *Advances in Neural Information Processing Systems* 35 (2022), 16455–16468.

[29] James Martens et al. 2010. Deep learning via hessian-free optimization.. In *Icml*, Vol. 27. 735–742.

[30] Christopher C Paige and Michael A Saunders. 1975. Solution of sparse indefinite systems of linear equations. *SIAM journal on numerical analysis* 12, 4 (1975), 617–629.

[31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[32] Barak A Pearlmutter. 1994. Fast exact multiplication by the Hessian. *Neural computation* 6, 1 (1994), 147–160.

[33] Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. 2019. Meta-learning with implicit gradients. *Advances in neural information processing systems* 32 (2019).

[34] Sirpa Saarinen, Randall Bramley, and George Cybenko. 1993. Ill-conditioning in neural network training problems. *SIAM Journal on Scientific Computing* 14, 3 (1993), 693–714.

[35] Levent Sagun, Leon Bottou, and Yann LeCun. 2016. Eigenvalues of the hessian in deep learning: Singularity and beyond. *arXiv preprint arXiv:1611.07476* (2016).

[36] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. 2015. Trust Region Policy Optimization. *CoRR* abs/1502.05477 (2015). arXiv:1502.05477 http://arxiv.org/abs/1502.05477

[37] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2018. High-Dimensional Continuous Control Using Generalized Advantage Estimation. arXiv:1506.02438 [cs.LG] https://arxiv.org/abs/1506.02438

[38] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *CoRR* abs/1707.06347 (2017). arXiv:1707.06347 http://arxiv.org/abs/1707.06347

[39] Damien Scieur, Gauthier Gidel, Quentin Bertrand, and Fabian Pedregosa. 2022. The curse of unrolling: Rate of differentiating through optimization. *Advances in Neural Information Processing Systems* 35 (2022), 17133–17145.

[40] Arunesh Sinha, Fei Fang, Bo An, and Christopher Kiekintveld. 2018. Stackelberg Security Games: Looking Beyond a Decade of Success. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. 5704–5710. https://www.ijcai.org/proceedings/2018/775

[41] Richard S Sutton and Andrew G. Barto. 2018. Reinforcement learning: An introduction. *A Bradford Book* (2018).

[42] Paul Vicol, Jonathan P. Lorraine, Fabian Pedregosa, David Duvenaud, and Roger B. Grosse. 2022. On Implicit Bias in Overparameterized Bilevel Optimization. In *Proceedings of the 39th International Conference on Machine Learning*. PMLR, 22234–22259. ISSN: 2640-3498.

[43] Junfeng Wen, Saurabh Kumar, Ramki Gummadi, and Dale Schuurmans. 2021. Characterizing the gap between actor-critic and policy gradient. In *International Conference on Machine Learning*. PMLR, 11101–11111.

[44] Shangtong Zhang, Bo Liu, Hengshuai Yao, and Shimon Whiteson. 2020. Provably convergent two-timescale off-policy actor-critic with function approximation. In *International Conference on Machine Learning*. PMLR, 11204–11213.

[45] Liyuan Zheng, Tanner Fiez, Zane Alumbaugh, Benjamin Chasnov, and Lillian J Ratliff. 2022. Stackelberg actor-critic: Game-theoretic reinforcement learning algorithms. In *Proceedings of the AAAI conference on Artificial Intelligence*, Vol. 36. 9217–9224.