Journal of Machine Learning Research 26 (2025) 1-56 Submitted 8/23; Revised 2/25; Published 5/25

Near-Optimal Nonconvex-Strongly-Convex Bilevel Optimization with Fully First-Order Oracles

Lesi Chen* CHENLC23@MAILS.TSINGHUA.EDU.CN
IIIS, Tsinghua University
Shanghai Qi Zhi Institute
Beijing, China

Yaohua Ma* MA-YH21@MAILS.TSINGHUA.EDU.CN
IIIS, Tsinghua University
Beijing, China

Jingzhao Zhang† JINGZHAOZ@MAIL.TSINGHUA.EDU.CN
IIIS, Tsinghua University
Shanghai AI Lab
Shanghai Qi Zhi Institute
Beijing, China

Editor: Dan Alistarh

Abstract

In this work, we consider bilevel optimization when the lower-level problem is strongly convex. Recent works show that with a Hessian-vector product (HVP) oracle, one can provably find an ϵ-stationary point within $\mathcal{O}(\epsilon^{-2})$ oracle calls. However, the HVP oracle may be inaccessible or expensive in practice. Kwon et al. (ICML 2023) addressed this issue by proposing a first-order method that can achieve the same goal at a slower rate of $\tilde{\mathcal{O}}(\epsilon^{-3})$. In this paper, we incorporate a two-time-scale update to improve their method to achieve the near-optimal $\tilde{\mathcal{O}}(\epsilon^{-2})$ first-order oracle complexity. Our analysis is highly extensible. In the stochastic setting, our algorithm can achieve the stochastic first-order oracle complexity of $\tilde{\mathcal{O}}(\epsilon^{-4})$ and $\tilde{\mathcal{O}}(\epsilon^{-6})$ when the stochastic noises are only in the upper-level objective and in both level objectives, respectively. When the objectives have higher-order smoothness conditions, our deterministic method can escape saddle points by injecting noise, and can be accelerated to achieve a faster rate of $\tilde{\mathcal{O}}(\epsilon^{-1.75})$ using Nesterov's momentum.

Keywords: bilevel optimization, stationary points, first-order methods

1. Introduction

In this paper, we consider the following bilevel optimization problem:

$$\min_{x\in\mathbb{R}^{d_x}} \varphi(x) := f(x, y^*(x)), \quad \text{where } y^*(x) = \arg\min_{y\in\mathbb{R}^{d_y}} g(x, y). \tag{1}$$

We call f the upper-level problem, g the lower-level problem, and φ the hyper-objective. This formulation covers many applications, including but not limited to hyperparameter

*. Equal contributions.
†. The corresponding author.

tuning (Franceschi et al., 2018; Pedregosa, 2016; Bae and Grosse, 2020; Mackay et al., 2018), neural architecture search (Liu et al., 2019; Wang et al., 2022b; Zoph and Le, 2016; Zhang et al., 2021), meta-learning (Finn et al., 2017; Fallah et al., 2020; Andrychowicz et al., 2016; Mishchenko et al., 2023; Rajeswaran et al., 2019; Chandra et al., 2022; Ji et al., 2022; Zhou et al., 2019), adversarial training (Zhang et al., 2022; Brückner and Scheffer, 2011; Wang et al., 2021; Bishop et al., 2020; Wang et al., 2022a; Robey et al., 2023) and data hyper-cleaning (Gao et al., 2022; Zhou et al., 2022; Ren et al., 2018; Yong et al., 2022; Shu et al., 2019; Li et al., 2022).

This work focuses on the complexity of solving Problem 1. Since the hyper-objective $\varphi(x)$ is usually a nonconvex function, a common goal for non-asymptotic analysis is to find an approximate stationary point (Carmon et al., 2020, 2021). When $g(x, \cdot)$ is convex but not strongly convex, this goal is intractable as there exists a hard instance such that any first-order algorithm would always get stuck and have no progress in x (Chen et al., 2024a, Theorem 3.2). Therefore, it is common to impose the lower-level strong convexity assumption in the literature (Dagréou et al., 2022; Ji et al., 2021; Chen et al., 2021; Khanduri et al., 2021; Hong et al., 2023): In this case, the hyper-gradient $\nabla\varphi(x)$ can be expressed by:

$$\begin{aligned}\nabla\varphi(x) &= \nabla_x f(x, y^*(x)) + \nabla y^*(x)\nabla_y f(x, y^*(x)) \\ &= \nabla_x f(x, y^*(x)) - \nabla^2_{xy} g(x, y^*(x))[\nabla^2_{yy} g(x, y^*(x))]^{-1}\nabla_y f(x, y^*(x)).\end{aligned} \tag{2}$$

This equation enables one to implement Hessian-vector-product(HVP)-based methods (Ji et al., 2021; Ghadimi and Wang, 2018) for nonconvex-strongly-convex bilevel optimization. These methods query HVP oracles to estimate $\nabla\varphi(x)$ via Equation 2, and then perform the so-called hyper-gradient descent on $\varphi(x)$. By using known convergence results of gradient descent for smooth nonconvex objectives, these methods can find an ϵ-first-order stationary point of $\varphi(x)$ with $\tilde{\mathcal{O}}(\epsilon^{-2})$ HVP oracle calls. However, in many applications (Sow et al., 2022; Song et al., 2019; Finn et al., 2017; Nichol et al., 2018), calling the HVP oracle may be very costly and become the computational bottleneck in bilevel optimization.

Very recently, Kwon et al. (2023) proposed a novel gradient-based algorithm that can find an ϵ-first-order stationary point of $\varphi(x)$ without resorting to second-order information. The key idea is to exploit the following value-function reformulation (Outrata, 1990; Ye and Zhu, 1995, 2010; Dempe, 2002; Lin et al., 2014) for Equation 1:

$$\min_{x\in\mathbb{R}^{d_x}, y\in\mathbb{R}^{d_y}} f(x, y), \quad \text{subject to} \;\; g(x, y) - g^*(x) \le 0. \tag{3}$$

Here $g^*(x) = g(x, y^*(x))$ is usually called the value-function. The Lagrangian with a multiplier $\lambda > 0$ for this inequality-constrained problem takes the form of:

$$\mathcal{L}_\lambda(x, y) := f(x, y) + \lambda(g(x, y) - g^*(x)). \tag{4}$$

Kwon et al. (2023) further defines the following auxiliary function:

$$\mathcal{L}^*_\lambda(x) := \mathcal{L}_\lambda(x, y^*_\lambda(x)), \;\text{ where } y^*_\lambda(x) = \arg\min_{y\in\mathbb{R}^{d_y}} \mathcal{L}_\lambda(x, y), \tag{5}$$

and showed that $\mathcal{L}^*_\lambda(x)$ is a good proxy of $\varphi(x)$ with

$$\|\nabla\mathcal{L}^*_\lambda(x) - \nabla\varphi(x)\| = \mathcal{O}\left(\kappa^3/\lambda\right).$$

Table 1: We present the oracle complexities of different **deterministic** methods for finding an ϵ-first-order stationary point of the hyper-objective under Assumption 3.1.

Oracle	Method	Oracle Calls
HVP	BA (Ghadimi and Wang, 2018)	$\mathcal{O}(\epsilon^{-2.5})$
	AID (Ji et al., 2021)	$\mathcal{O}(\epsilon^{-2})$
	ITD (Ji et al., 2021)	$\tilde{\mathcal{O}}(\epsilon^{-2})$
Gradient	PZOBO (Sow et al., 2022)	$\tilde{\mathcal{O}}(d_x^2\epsilon^{-4})$
	BOME (Liu et al., 2022) [(a)]	$\tilde{\mathcal{O}}(\epsilon^{-6})$
	PBGD (Shen and Chen, 2023) [(a)]	$\tilde{\mathcal{O}}(\epsilon^{-3})$
	F^{2}SA (Kwon et al., 2023) [(b)]	$\tilde{\mathcal{O}}(\epsilon^{-3})$
	F^{2}BA (Algorithm 1)	$\tilde{\mathcal{O}}(\epsilon^{-2})$

(a) Shen and Chen (2023), Liu et al. (2022) use a weaker notation of stationary, they proved the convergence to an ϵ-stationary point of $\mathcal{L}_\lambda(x,y)$. We present the complexity of Shen and Chen (2023), Liu et al. (2022) for $\lambda \asymp \epsilon^{-1}$, which also implies a $\mathcal{O}(\kappa\epsilon)$-stationary point of $\varphi(x)$ as we prove in Appendix H.1.

(b) Kwon et al. (2023) additionally assume $\nabla^2 f$ is Lipschitz and both $\|\nabla_x f(x,y)\|$, $\|\nabla_x g(x,y)\|$ are upper bounded. But it is unnecessary if we use a fixed penalty λ as discussed in Appendix H.2.

It indicates that when we set $\lambda \asymp \epsilon^{-1}$, an ϵ-first-order stationary point of $\mathcal{L}_\lambda^*(x)$ is also an $\mathcal{O}(\epsilon)$-first-order stationary point of $\varphi(x)$. Therefore, we can reduce the problem of finding an ϵ-first-order stationary point of $\varphi(x)$ to finding that of $\mathcal{L}_\lambda^*(x)$. The advantage of this reduction is that $\nabla\mathcal{L}_\lambda^*(x)$ can be evaluated with only first-order information of f and g:

$$\begin{aligned}\nabla\mathcal{L}_\lambda^*(x) &= \nabla_x\mathcal{L}_\lambda^*(x, y_\lambda^*(x)) + \nabla y_\lambda^*(x)\underbrace{\nabla_y\mathcal{L}_\lambda^*(x, y_\lambda^*(x))}_{=0} \\ &= \nabla_x f(x, y_\lambda^*(x)) + \lambda(\nabla_x g(x, y_\lambda^*(x)) - \nabla_x g(x, y^*(x))).\end{aligned} \tag{6}$$

Since $\mathcal{L}_\lambda(x,y)$ in Equation 4 is $\mathcal{O}(\lambda)$-gradient Lipschitz, the single-time-scale ($\eta_x = \eta_y$) approach for penalty methods (Kwon et al., 2023; Shen and Chen, 2023) requires a complexity of $\tilde{\mathcal{O}}(\lambda\epsilon^{-2}) = \tilde{\mathcal{O}}(\epsilon^{-3})$ to find an ϵ-first-order stationary point, which is slower than the $\tilde{\mathcal{O}}(\epsilon^{-2})$ complexity of HVP-based methods. Kwon et al. (2023) then conjectured that a fundamental gap may exist between gradient-based and HVP-based methods.

However, this paper refutes this conjecture by showing that gradient-based methods can also achieve the near-optimal $\tilde{\mathcal{O}}(\epsilon^{-2})$ rate as HVP-based methods. We prove that for a sufficiently large λ, the proxy $\mathcal{L}_\lambda^*(x)$ satisfies:

$$\|\nabla^2\mathcal{L}_\lambda^*(x)\| \asymp \|\nabla^2\varphi(x)\| \asymp \kappa^3,$$

where κ is the condition number that will be formally defined later. It indicates that although λ is large, the gradient Lipschitz coefficient of $\mathcal{L}_\lambda^*(x)$ would remain constant and not depend on λ. Therefore, although the largest possible step size in y is still bounded

Table 2: We present the oracle complexities of different **stochastic** methods for finding an ϵ-first-order stationary point of the hyper-objective under Assumption 3.1, 4.1.

Oracle	Method	Partially Stochastic	Fully Stochastic
HVP*	BSA (Ghadimi and Wang, 2018)	$\mathcal{O}(\epsilon^{-6})$	$\mathcal{O}(\epsilon^{-6})$
	TTSA (Hong et al., 2023)	$\mathcal{O}(\epsilon^{-5})$	$\mathcal{O}(\epsilon^{-5})$
	StocBiO (Ji et al., 2021)	$\tilde{\mathcal{O}}(\epsilon^{-4})$	$\mathcal{O}(\epsilon^{-4})$
Gradient	PZOBO-S (Sow et al., 2022)	$\tilde{\mathcal{O}}(d_x^2\epsilon^{-8})$	$\tilde{\mathcal{O}}(d_x^2\epsilon^{-8})$
	F^{2}SA (Kwon et al., 2023)	$\tilde{\mathcal{O}}(\epsilon^{-5})$	$\tilde{\mathcal{O}}(\epsilon^{-7})$
	F^{2}BSA (Algorithm 2)	$\tilde{\mathcal{O}}(\epsilon^{-4})$	$\tilde{\mathcal{O}}(\epsilon^{-6})$

We remark in "*" that HVP-based methods **additionally** assume the stochastic estimators of second-order derivatives $\nabla^2_{xy}g/\nabla^2_{yy}g$ are unbiased and have bounded variance. Therefore, it is reasonable that gradient-based methods has worse complexities than HVP-based method in the fully stochastic case. Such a gap also exists in nonconvex single-level optimization (Arjevani et al., 2020).

by $\mathcal{O}(1/\lambda)$, *i.e.* $\eta_y = \mathcal{O}(\epsilon)$, we can use a larger step size in x because the landscape in x is much smoother, *i.e.* $\eta_x = \mathcal{O}(1)$. It motivates us to propose the Fully First-order Bilevel Approximation (F^{2}BA) which uses the two-time-scale step size ($\eta_x \neq \eta_y$) in the method by Kwon et al. (2023). Our proposed method can find an ϵ-stationary point with $\tilde{\mathcal{O}}(\epsilon^{-2})$ complexity. As nonconvex-strongly-concave bilevel optimization problems subsume standard nonconvex optimization problems, the $\tilde{\mathcal{O}}(\epsilon^{-2})$ upper bound is optimal up to logarithmic factors according to the lower bound provided by Carmon et al. (2020). We compare our complexity result with prior works in Table 1. In the stochastic setting, we prove that our algorithm enjoys a complexity of $\tilde{\mathcal{O}}(\epsilon^{-4})$ and $\tilde{\mathcal{O}}(\epsilon^{-6})$ for partially and fully stochastic cases, respectively, which also improves the best-known results Kwon et al. (2023). We compare our result in the stochastic case with prior works in Table 2.

We also study the problem under additional smoothness conditions. If we additionally assume the Hessian Lipschitz continuity of f and the third-order derivative Lipschitz continuity of g, we prove that

$$\|\nabla^2\mathcal{L}_\lambda^*(x) - \nabla^2\varphi(x)\| = \mathcal{O}(\kappa^6/\lambda) \quad \text{and} \quad \|\nabla^3\mathcal{L}_\lambda^*(x)\| \asymp \|\nabla^3\varphi(x)\| \asymp \kappa^5.$$

Based on this observation, we propose the Perturbed F^{2}BA (or PF^{2}BA for short, see Algorithm 3), which can provably find an ϵ-second-order stationary point of $\varphi(x)$ within $\tilde{\mathcal{O}}(\epsilon^{-2})$ gradient oracle calls. Our result shows that gradient-based methods can also escape saddle points in bilevel optimization like HVP-based methods (Huang et al., 2025). Additionally, we further exploit the Hessian Lipschitz continuity of $\mathcal{L}_\lambda^*(x)$ to achieve a better complexity of $\tilde{\mathcal{O}}(\epsilon^{-1.75})$ by incorporating Nesterov's acceleration. We name this new method Accelerated F^{2}BA (or AccF^{2}BA for short, see Algorithm 4). We compare the results for finding second-order stationary points in Table 3.

Table 3: We present the oracle complexities of different **deterministic** methods for finding an ϵ-**second-order** stationary point of the hyper-objective under Asmp. 3.1, 3.2.

Oracle	Method	Oracle Calls
HVP	PAID (Huang et al., 2025)	$\tilde{\mathcal{O}}(\epsilon^{-2})$
	PRAHGD (Yang et al., 2023)	$\tilde{\mathcal{O}}(\epsilon^{-1.75})$
Gradient	PF^{2}BA (Algorithm 3)	$\tilde{\mathcal{O}}(\epsilon^{-2})$
	AccF^{2}BA (Algorithm 4)	$\tilde{\mathcal{O}}(\epsilon^{-1.75})$

Notations. We use notations $\mathcal{O}(\cdot)$, $\tilde{\mathcal{O}}(\cdot)$, $\Omega(\cdot)$, $\asymp$ as follows: given two functions $p : \mathbb{R}^+ \to \mathbb{R}^+$ and $q : \mathbb{R}^+ \to \mathbb{R}^+$, $p(x) = \mathcal{O}(q(x))$ means $\limsup_{x\to+\infty} p(x)/q(x) < +\infty$; $p(x) = \tilde{\mathcal{O}}(p(x))$ means there exists some positive integer $k \geq 0$ such that $p(x) = \mathcal{O}(q(x)\log^k(q(x)))$, $p(x) = \Omega(q(x))$ means $\limsup_{x\to+\infty} p(x)/q(x) > 0$, and $p(x) \asymp q(x)$ means we both have $p(x) = \mathcal{O}(q(x))$ and $p(x) = \Omega(q(x))$. We use $I_d \in \mathbb{R}^{d\times d}$ to denote a d=dimensional identity matrix. For two symmetric matrices A and B, we use $A \succeq B$ to indicate that $A - B$ is positive semidefinite. We use $\mathbb{B}(r)$ to denote the Euclidean ball centered at the origin and radius r. For a function $h : \mathbb{R}^d \to \mathbb{R}$, we use $\nabla h \in \mathbb{R}^d, \nabla^2 h \in \mathbb{R}^{d\times d}, \nabla^3 h \in \mathbb{R}^{d\times d\times d}$ to denote its gradient, Hessian, and third-order derivative, and use h^* to denote the global minimum of $h(\cdot)$. We denote $\|\cdot\|$ to be the operator norm of a tensor and more details about the notations of tensors can be found in Appendix A.

2. Related Works

We review the related works on HVP-based and gradient-based methods.

HVP-Based Methods for Bilevel Optimization. Most existing HVP-based methods for nonconvex-strongly-convex bilevel optimization can be categorized into the approximate implicit differentiation (AID) methods and the iterative differentiation (ITD) methods. The AID approach (Liao et al., 2018; Ji et al., 2021; Ghadimi and Wang, 2018; Lorraine et al., 2020) constructs hyper-gradients explicitly according to Equation 2 that $\nabla\varphi(x) = \nabla_x f(x, y^*(x)) - \nabla^2_{xy} g(x, y^*(x))v^*$, where v^* is the solution to the linear system $\nabla^2_{yy} g(x, y^*(x))v^* = \nabla_y f(x, y^*(x))$. Then one can use iterative algorithms such as fix point iteration or conjugate gradient method to solve this linear system, avoiding the computation of the Hessian inverse (Grazzi et al., 2020; Ji et al., 2021; Hong et al., 2023). The ITD approach (Maclaurin et al., 2015; Shaban et al., 2019; Domke, 2012; Franceschi et al., 2017, 2018) takes advantage of the fact that backpropagation can be efficiently implemented via modern automatic differentiation frameworks such as PyTorch (Paszke et al., 2019). These methods approximate hyper-gradients by $\partial f(x, y^K(x))/\partial x$, where $y^K(x)$ is the output from K-steps of gradient descent on $g(x, \cdot)$. Although the ITD approach does not query second-order information explicitly, the analytical form of $\partial f(x, y^K(x))/\partial x$ involves second-order derivatives (Ji et al., 2021), which also requires HVP oracle implicitly. Both AID and ITD

methods require $\tilde{\mathcal{O}}(\epsilon^{-2})$ HVP oracle calls to find an ϵ-first-order stationary point of $\varphi(x)$. Recently, Huang et al. (2025) proposed the perturbed AID to find an ϵ-second-order stationary point with $\tilde{\mathcal{O}}(\epsilon^{-2})$ complexity under additional smoothness conditions. Yang et al. (2023) proposed the Perturbed Restarted Accelerated HyperGradient Descent (PRAHGD) algorithm with an improved complexity of $\tilde{\mathcal{O}}(\epsilon^{-1.75})$. Wang et al. (2024, Section 4.2) also mentioned that the $\tilde{\mathcal{O}}(\epsilon^{-1.75})$ complexity can also be achieved by applying the Inexact APPA Until Nonconvexity (IAPUN) algorithm to bilevel problems.

Gradient-Based Methods for Bilevel Optimization. The basic idea of gradient-based methods for bilevel optimization is to approximate the hyper-gradient in Equation 2 using gradient information. Sow et al. (2022) proposed the Partial Zeroth-Order based Bilevel Optimizer (PZOBO) that applies a zeroth-order-like estimator to approximate the response Jacobian matrix $\nabla y^*(x)$ in Equation 2, and hence the complexity has a polynomial dependency on the dimension of the problem like the standard results in zeroth-order optimization (Duchi et al., 2015; Ghadimi and Lan, 2013; Kornowski and Shamir, 2023). Liu et al. (2022) first observed Equation 6 that $\nabla \mathcal{L}_\lambda^*(x)$ only involves first-order information and proposed the method named Bilevel Optimization Made Easy (BOME). Shen and Chen (2023) studied the relationship of global and local solutions of the penalty function and the original bilevel problem, and proposed the penalty-based bilevel gradient descent (PBGD) that can converge to the stationary point of $\mathcal{L}_\lambda(x, y)$ in Equation 4 with $\tilde{\mathcal{O}}(\lambda\epsilon^{-2})$ oracle calls. But Liu et al. (2022), Shen and Chen (2023) did not provide any convergence result of $\varphi(x)$. Remarkably, Kwon et al. (2023) established the relationship between $\mathcal{L}_\lambda^*(x)$ and $\varphi(x)$, and proposed the Fully First-order Stochastic Approximation (F^2SA) that can provably find an ϵ-first-order stationary point of $\varphi(x)$ within $\tilde{\mathcal{O}}(\epsilon^{-3})$ oracle complexity. As a by-product of their analysis, one can also show that BOME (Liu et al., 2022) and PBGD (Shen and Chen, 2023) converge to an ϵ-stationary point of $\varphi(x)$ at the rate of $\tilde{\mathcal{O}}(\epsilon^{-6})$ and $\tilde{\mathcal{O}}(\epsilon^{-3})$, respectively. However, before our work, we did not know whether gradient-based methods could have comparable theoretical guarantees to HVP-based methods.

3. Preliminaries

In this section, we introduce the different setups studied in this work. We focus on stating the assumptions and definitions used later, while delaying a more comprehensive description to future sections where we state and describe our main results.

First-Order Stationary Points We first discuss the assumptions for finding first-order stationary points of the hyper-objective $\varphi(x)$, detailed below.

Assumption 3.1 *Suppose that*

a. $g(x, y)$ *is* μ*-strongly convex in* y*;*

b. $g(x, y)$ *is* L_g*-gradient Lipschitz;*

c. $g(x, y)$ *is* ρ_g*-Hessian Lipschitz;*

d. $f(x, y)$ *is* C_f*-Lipschitz in* y*;*

e. $f(x, y)$ *is* L_f*-gradient Lipschitz;*

f. $f(x, y)$ *is two-times continuous differentiable;*

g. $\varphi(x)$ *is lower bounded,* i.e. $\inf_{x \in \mathbb{R}^{d_x}} \varphi(x) > -\infty$*;*

The above assumptions are common and necessary for non-asymptotic analyses. According to Theorem 3.2 in (Chen et al., 2024a), bilevel problems are intractable in general when $g(x, \cdot)$ is not strongly convex. For this reason, existing non-asymptotic analyses for bilevel optimization commonly make the lower-level strong convexity assumption (Assumption 3.1a). In this case, $\nabla\varphi(x)$ can be expressed jointly by $\nabla_x f(x, y)$, $\nabla_y f(x, y)$, $\nabla^2_{xy} g(x, y)$ and $\nabla^2_{yy} g(x, y)$ as Equation 2. This expression indicates that we need the smoothness conditions for f and g (Assumption 3.1b - 3.1e) to guarantee the gradient Lipschitz continuity of $\varphi(x)$. Besides these, we adopt Assumption 3.1f to ensure that $\mathcal{L}_\lambda(x, y)$ (Equation 5) is two-times continuous differentiable, and Assumption 3.1g to ensure the bilevel optimization problem (Equation 1) is well-defined.

Definition 3.1 *Under Assumption 3.1, we define the largest smoothness constant* $\ell := \max\{C_f, L_f, L_g, \rho_g\}$ *and the condition number* $\kappa := \ell/\mu$*.*

We can derive from the above assumptions that $\nabla\varphi(x)$ is uniquely defined and Lipschitz continuous, formally stated as follows.

Proposition 3.1 (Ghadimi and Wang (2018, Lemma 2.2)) *Under Assumption 3.1, the hyper-gradient* $\nabla\varphi(x)$ *is uniquely defined by Equation 2, and the hyper-objective* $\varphi(x)$ *is* L_φ*-gradient Lipschitz, where* $L_\varphi = \mathcal{O}(\ell\kappa^3)$*.*

As the above proposition ensures that the hyper-objective $\varphi(x)$ is differentiable, we can define the ϵ-first-order stationary points as follows.

Definition 3.2 *Given a differentiable function* $\varphi(x) : \mathbb{R}^d \to \mathbb{R}$*, we call* $\hat{x}$ *an* ϵ*-first-order stationary point of* $\varphi(x)$ *if* $\|\nabla\varphi(\hat{x})\| \le \epsilon$*.*

Second-Order Stationary Points Algorithms that pursue first-order stationary points may get stuck in saddle points and have poor performances (Dauphin et al., 2014). For single-level optimization problems, there have been many researchers studying how to escape saddle points (Jin et al., 2017; Ge et al., 2015; Fang et al., 2019; Tripuraneni et al., 2018; Agarwal et al., 2017; Lee et al., 2016; Allen-Zhu and Li, 2018; Carmon et al., 2017; Zhou et al., 2020; Allen-Zhu, 2018b; Xu et al., 2018; Zhang and Li, 2021). A common assumption in these works is to suppose that the objective is Hessian Lipschitz. When generalizing to bilevel optimization, we also expect $\varphi(x)$ to be Hessian Lipschitz, which can be proved if we further assume the following higher-order smoothness condition of f and g.

Assumption 3.2 *Suppose that*

a. $f(x, y)$ *is three-times continuous differentiable;*

b. $f(x, y)$ *is* ρ_f*-Hessian Lipschitz;*

Algorithm 1 F^{2}BA (x_0, y_0)

$z_0 = y_0$
for $t = 0, 1, \cdots, T-1$
  $y_t^0 = y_t,\ z_t^0 = z_t$
  for $k = 0, 1, \cdots, K-1$
    $z_t^{k+1} = z_t^k - \eta_z \lambda \nabla_y g(x_t, z_t^k)$
    $y_t^{k+1} = y_t^k - \eta_y \left(\nabla_y f(x_t, y_t^k) + \lambda \nabla_y g(x_t, y_t^k)\right)$
  end for
  $\hat{\nabla} \mathcal{L}_\lambda^*(x_t) = \nabla_x f(x_t, y_t^K) + \lambda(\nabla_x g(x_t, y_t^K) - \nabla_x g(x_t, z_t^K))$
  $x_{t+1} = x_t - \eta_x \hat{\nabla} \mathcal{L}_\lambda^*(x_t)$
end for

c. $g(x, y)$ is ν_g-third-order derivative Lipschitz.

Definition 3.3 *Under Assumption 3.1 and 3.2, we define the largest smoothness constant $\ell := \max\{C_f, L_f, L_g, \rho_g, \rho_f, \nu_g\}$ and the condition number $\kappa := \ell/\mu$.*

Proposition 3.2 (Huang et al. (2025, Lemma 3.4)) *Under Assumption 3.1 and 3.2, the hyper-objective $\varphi(x)$ is two-times continuously differentiable and ρ_φ-Hessian Lipschitz, where $\rho_\varphi = \mathcal{O}(\ell\kappa^5)$.*

We can then formally define the approximate second-order stationary point as follows.

Definition 3.4 (Nesterov and Polyak (2006)) *Given a two-times continuously differentiable function $\varphi(x) : \mathbb{R}^d \to \mathbb{R}$ with ρ-Lipschitz Hessian, we call $\hat{x}$ an ϵ-second-order stationary point of $\varphi(x)$ if*

$$\|\nabla \varphi(\hat{x})\| \le \epsilon, \quad \nabla^2 \varphi(\hat{x}) \succeq -\sqrt{\rho\epsilon} I_d.$$

We will discuss finding second-order stationary points for bilevel problems later in Section 5.

4. Finding First-Order Stationary Points

In bilevel optimization, the hyper-objective $\varphi(x)$ is usually a nonconvex function. Since finding the global minimum of a nonconvex function in the worst case requires an exponential number of queries, a common compromise is to find a local minimum (Ge et al., 2016). First-order stationary points (Definition 3.2) are the points that satisfy the first-order necessary condition of a local minimum, which turns out to be a valid optimality criterion for nonconvex optimization.

4.1 Near-Optimal Rate in the Deterministic Case

In this section, we propose our method, namely Fully First-order Bilevel Approximation (F^{2}BA). The detailed procedure of is presented in Algorithm 1. The algorithm introduces

an auxiliary variable $z \in \mathbb{R}^{d_y}$, and performs gradient descent jointly in x, y, z to solve the following optimization problem:

$$\min_{x\in\mathbb{R}^{d_x}, y\in\mathbb{R}^{d_y}} \left\{ f(x,y) + \lambda \left(g(x,y) - \min_{z\in\mathbb{R}^{d_y}} g(x,z) \right) \right\} \overset{\text{Eq.5}}{=} \min_{x\in\mathbb{R}^{d_x}} \mathcal{L}_\lambda^*(x). \tag{7}$$

The intuition behind the algorithm is that optimizing $\mathcal{L}_\lambda^*(x)$ is almost equivalent to optimizing $\varphi(x)$ when λ is large. Therefore, to analyze the convergence of the algorithm, we first characterize the relationship between $\mathcal{L}_\lambda^*(x)$ and $\varphi(x)$ in the following lemmas.

Lemma 4.1 *Suppose Assumption 3.1 holds. Define ℓ, κ according to Definition 3.1, and $\mathcal{L}_\lambda^*(x)$ according to Equation 5. Set $\lambda \geq 2L_f/\mu$, then it holds that*

a. $\|\nabla \mathcal{L}_\lambda^*(x) - \nabla\varphi(x)\| = \mathcal{O}(\ell\kappa^3/\lambda),\ \forall x \in \mathbb{R}^{d_x}$ (Lemma B.4).

b. $|\mathcal{L}_\lambda^*(x) - \varphi(x)| = \mathcal{O}(\ell\kappa^2/\lambda),\ \forall x \in \mathbb{R}^{d_x}$ (Lemma B.3).

c. $\mathcal{L}_\lambda^*(x)$ *is* $\mathcal{O}(\ell\kappa^3)$*-gradient Lipschitz* (Lemma B.7).

All the formal versions of these lemmas and the corresponding proofs can be found in Appendix B. Lemma 4.1a is a restatement of Lemma 3.1 by Kwon et al. (2023), which demonstrates that when $\lambda \asymp \epsilon^{-1}$, an ϵ-first-order stationary point of $\mathcal{L}_\lambda^*(x)$ is also an $\mathcal{O}(\epsilon)$-first-order stationary of $\varphi(x)$. Lemma 4.1c is a new result proved in this paper. It means although $\mathcal{L}_\lambda^*(x)$ depends on λ, when λ exceeds a certain threshold, the gradient Lipschitz coefficient of $\mathcal{L}_\lambda^*(x)$ only depends on that of $\varphi(x)$ and does not depend on λ. The high-level intuition is that since we know $\mathcal{L}_\lambda^*(x)$ would converge to a fixed objective $\nabla\varphi(x)$ when $\lambda \to \infty$, it is possible that the Lipschitz constant of $\mathcal{L}_\lambda^*(x)$ also converges to a fixed value. Below, we sketch the proof of Lemma 4.1c. The non-trivial part is the show that $\lambda(g(x, y_\lambda^*(x)) - g(x, y^*(x))$ is $\mathcal{O}(1)$-gradient Lipschitz independent of λ. We prove this by bounding the operator norm of its second-order derivative, which by calculation is

$$\begin{aligned}&\lambda(\nabla_{xx}^2 g(x, y_\lambda^*(x)) - \nabla_{xx}^2 g(x, y^*(x)))\\ &+ \lambda\left(\nabla y_\lambda^*(x)\nabla_{yx}^2 g(x, y_\lambda^*(x)) - \nabla y^*(x)\nabla_{yx}^2 g(x, y^*(x))\right),\end{aligned}$$

which should be $\mathcal{O}(1)$ since we can show that

$$\|y_\lambda^*(x) - y^*(x)\| = \mathcal{O}(1/\lambda), \quad \text{and} \quad \|\nabla y_\lambda^*(x) - \nabla y^*(x)\| = \mathcal{O}(1/\lambda).$$

Since the convergence rate of gradient descent depends on the gradient Lipschitz coefficient of the objective, Lemma 4.1c indicates that optimizing $\mathcal{L}_\lambda^*(x)$ is as easy as optimizing $\varphi(x)$. Note that $\nabla\mathcal{L}_\lambda^*(x)$ only involves first-order information (Equation 6), Lemma 4.1c then suggests that first-order methods can have the same convergence rate as HVP-based methods, as stated in the following theorem.

Theorem 4.1 *Suppose Assumption 3.1 holds. Define $\Delta := \varphi(x_0) - \inf_{x\in\mathbb{R}^{d_x}} \varphi(x)$ and $R := \|y_0 - y^*(x_0)\|^2$. Let $\eta_x \asymp \ell^{-1}\kappa^{-3}$, $\lambda \asymp \max\{\kappa/R,\ \ell\kappa^2/\Delta,\ \ell\kappa^3/\epsilon\}$ and set other parameters in Algorithm 1 as*

$$\eta_z = \eta_y = \frac{1}{2\lambda L_g},\ K = \mathcal{O}\left(\frac{L_g}{\mu}\log\left(\frac{\lambda L_g}{\mu}\right)\right),$$

Algorithm 2 F^{2}BSA (x_0, y_0)

$z_0 = y_0$
for $t = 0, 1, \cdots, T-1$
  $y_t^0 = y_t,\ z_t^0 = z_t$
  for $k = 0, 1, \cdots, K-1$
    $z_t^{k+1} = z_t^k - \eta_z \lambda \nabla_y g(x_t, z_t^k; B_{\text{in}})$
    $y_t^{k+1} = y_t^k - \eta_y \left(\nabla_y f(x_t, y_t^k; B_{\text{in}}) + \lambda \nabla_y g(x_t, y_t^k; B_{\text{in}})\right)$
  end for
  $G_t = \nabla_x f(x_t, y_t^K; B_{\text{out}}) + \lambda(\nabla_x g(x_t, y_t^K; B_{\text{out}}) - \nabla_x g(x_t, z_t^K; B_{\text{out}}))$
  $x_{t+1} = x_t - \eta_x G_t$
end for

then it can find an ϵ-first-order stationary point of $\varphi(x)$ within $\mathcal{O}(\ell\kappa^4\epsilon^{-2}\log(\ell\kappa/\epsilon))$ first-order oracle calls, where ℓ, κ are defined in Definition 3.1.

Remark 1 *When the upper-level function only depends on x, i.e. we have $f(x, y) \equiv h(x)$ for some function $h(\cdot)$, the bilevel problem reduces to a single-level problem, for which Carmon et al. (2020) proved a lower complexity bound of $\Omega(\epsilon^{-2})$. Therefore, we can conclude that the first-order oracle complexity of F^{2}BA we proved is near-optimal.*

We defer the proof of Theorem 4.1 to Appendix D. The complexity of F^{2}BA in Theorem 4.1 achieves the near-optimal rate in the dependency on ϵ, and matches the state-of-the-art second-order methods AID and ITD (Ji et al., 2021) in the dependency of κ. Our result, for the first time, closes the gap between gradient-based and HVP-based methods for nonconvex-strongly-convex bilevel optimization. In Appendix G, we also discuss the advantage of gradient-based methods compared to HVP-based methods in the distributed scenarios. The distributed F^{2}BA is much more easy to implement than HVP-based methods.

4.2 Extension to the Stochastic Case

In this section, we study the case when the algorithms only have access to stochastic gradient oracles that satisfy the following assumptions.

Assumption 4.1 *We access the gradients of objective functions via unbiased estimators $\nabla f(x, y; \phi_f)$ and $\nabla g(x, y; \phi_g)$ such that*

$$\mathbb{E}_{\phi_f}\left[\nabla f(x, y; \phi_f)\right] = \nabla f(x, y), \quad \mathbb{E}_{\phi_g}\left[\nabla g(x, y; \phi_g)\right] = \nabla g(x, y).$$

And the variance of stochastic gradients is bounded:

$$\mathbb{E}_{\phi_f}\left[\|\nabla f(x, y; \phi_f) - \nabla f(x, y)\|^2\right] \le \sigma_f^2, \quad \mathbb{E}_{\phi_g}\left[\|\nabla g(x, y; \phi_g) - \nabla g(x, y)\|^2\right] \le \sigma_g^2.$$

We propose the Fully First-order Stochastic Bilevel Approximation (F^{2}BSA) in Algorithm 2. At each iteration, our algorithm samples a mini-batch to estimate the true gradient:

$$\nabla f(x, y; B) = \frac{1}{B}\sum_{i=1}^{B} \nabla f(x, y; \phi_f^{(i)}), \quad \nabla g(x, y; B) = \frac{1}{B}\sum_{i=1}^{B} \nabla g(x, y; \phi_g^{(i)}),$$

where both $\phi_f^{(i)}$ and $\phi_g^{(i)}$ are sampled i.i.d. ate each iteration, and B denotes the batch size. We use B_{out} and B_{in} to denote the batch size of outer and inner loop, respectively. By properly setting the value of B_{out} and B_{in}, F^{2}BSA can track the deterministic method F^{2}BA up to $\mathcal{O}(\epsilon)$ error, and therefore also converges to an ϵ-stationary point of $\varphi(x)$.

Theorem 4.2 *Suppose Assumption 3.1 and 4.1 hold. Let ℓ, κ defined as Definition 3.1. Define $\Delta := \varphi(x_0) - \inf_{x\in\mathbb{R}^{d_x}} \varphi(x)$ and $R := \|y_0 - y^*(x_0)\|^2$. Let $\eta_x \asymp \ell^{-1}\kappa^{-3}$, $\lambda \asymp \max\left\{\kappa/R,\ \ell\kappa^2/\Delta,\ \ell\kappa^3/\epsilon\right\}$ and set other parameters in Algorithm 2 as*

$$\eta_z = \eta_y = \frac{1}{2\lambda L_g},\ K_t = \tilde{\mathcal{O}}\left(\frac{L_g \log \delta_t}{\mu}\right),$$
$$B_{\text{out}} \asymp \frac{\sigma_f^2 + \lambda^2\sigma_g^2}{\epsilon^2},\quad B_{\text{in}} \asymp \frac{L_f^2 + \lambda^2 L_g^2}{\lambda^2\mu^2}\cdot B_{\text{out}}.$$

where δ_t is defined via the recursion

$$\delta_{t+1} = \frac{1}{2}\delta_t + \frac{34L_g^2}{\mu^2}\|x_{t+1} - x_t\|^2 + \frac{\sigma_g^2}{2L_g B_{\text{in}}},\quad \delta_0 = \mathcal{O}(R),$$

then it can output a point such that $\mathbb{E}\|\nabla\varphi(x)\| \le \epsilon$ within $T = \mathcal{O}(\ell\kappa^3\epsilon^{-2})$ iterations. The total number of stochastic first-order oracle calls is bounded by

$$\begin{cases} \mathcal{O}(\ell\kappa^6\epsilon^{-4}\log(\ell\kappa/\epsilon)), & \sigma_f > 0, \sigma_g = 0; \\ \mathcal{O}(\ell^3\kappa^{12}\epsilon^{-6}\log(\ell\kappa/\epsilon)), & \sigma_f > 0, \sigma_g > 0. \end{cases}$$

Our results improve that of Kwon et al. (2023) by a factor of $\mathcal{O}(\epsilon^{-1})$ in both cases. In the partially stochastic case ($\sigma_g > 0, \sigma_g = 0$), the $\tilde{\mathcal{O}}(\epsilon^{-4})$ upper bound is near-optimal due to the lower bound by Arjevani et al. (2023). However, the $\tilde{\mathcal{O}}(\epsilon^{-6})$ complexity in the fully stochastic case ($\sigma_f > 0, \sigma_g > 0$) is worse than the $\tilde{\mathcal{O}}(\epsilon^{-4})$ upper bound by the current best stochastic HVP-based methods (Ji et al., 2021). It is reasonable as the stochastic HVP-based methods rely on stronger assumptions that $\nabla^2 g(x, y; \phi)$ is unbiased and has bounded variance. Similar separation between HVP-based and gradient-based methods in the stochastic setting also exists in single-level optimization (Arjevani et al., 2020).

Remark 2 *One drawback of Theorem 4.2 is that it requires a large batch size $B \asymp (\sigma_f^2 + \lambda^2\sigma_g^2)\epsilon^{-2}$ to track the deterministic algorithm. The large batch size ensures that the hypergradient estimator G_t is nearly unbiased, i.e., $\|\mathbb{E}G_t - \nabla\mathcal{L}_\lambda^*(x_t)\| = \mathcal{O}(\epsilon)$.*

In contrast, the algorithms in (Kwon et al., 2023) use single-batch SGD update in both inner and outer loops. To support such small-batch updates, we can apply the existing technique in the literature (Asi et al., 2021; Hu et al., 2021), known as the multi-level Monte Carlo (MLMC) technique (Blanchet and Glynn, 2015; Giles, 2008). This method converts the original gradient estimator to a nearly unbiased estimator at negligible extra costs. We specify Algorithm 5 in Appendix E, and provide the analysis in Theorem E.3. The convergence rate matches the large-batch algorithm. We provide more details in Appendix E.

Algorithm 3 Perturbed F^{2}BA (x_0, y_0)

1: $z_0 = y_0$
2: **for** $t = 0, 1, \cdots, T-1$
3: $\quad y_t^0 = y_t,\ z_t^0 = z_t$
4: $\quad$ **for** $k = 0, 1, \cdots, K_t - 1$
5: $\quad\quad z_t^{k+1} = z_t^k - \eta_z \lambda \nabla_y g(x_t, z_t^k)$
6: $\quad\quad y_t^{k+1} = y_t^k - \eta_y \left(\nabla_y f(x_t, y_t^k) + \lambda \nabla_y g(x_t, y_t^k)\right)$
7: $\quad$ **end for**
8: $\quad \hat{\nabla}\mathcal{L}_\lambda^*(x_t) = \nabla_x f(x_t, y_t^{K_t}) + \lambda(\nabla_x g(x_t, y_t^{K_t}) - \nabla_x g(x_t, z_t^{K_t}))$
9: $\quad$ **if** $\|\hat{\nabla}\mathcal{L}_\lambda^*(x_t)\| \le \frac{4}{5}\epsilon$ **and** no perturbation added in the last $\mathcal{T}$ steps
10: $\quad\quad x_t = x_t - \eta_x \xi_t$, where $\xi_t \sim \mathbb{B}(r)$
11: $\quad$ **end if**
12: $\quad x_{t+1} = x_t - \eta_x \hat{\nabla}\mathcal{L}_\lambda^*(x_t)$
13: **end for**

5. Finding Second-Order Stationary Points

We have shown in the previous section that the F^{2}BA is near-optimal for finding first-order stationary points. However, a first-order stationary point may be a saddle point or a local maximizer, which needs to be escaped from for an effective optimizer. For this reason, many works aim to find a second-order stationary point (Definition 3.4).

5.1 Perturbed F^{2}BA

In this section, we propose a simple variant of F^{2}BA (Algorithm 3) that can achieve this higher goal. The only difference to Algorithm 1 is the additional Line 9-10 in Algorithm 3, which is motivated by the perturbed strategy for escaping saddle points (Jin et al., 2017).

To prove the desired conclusion, we need to extend the analysis in Lemma 4.1 to higher-order derivatives. Below, we show that once λ is sufficiently large, $\mathcal{L}_\lambda^*(x)$ and $\varphi(x)$ have not only very close gradients (Lemma 4.1a) but also very close Hessian matrices.

Lemma 5.1 *Suppose both Assumption 3.1 and 3.2 hold. Define ℓ, κ according to Definition 3.3, and $\mathcal{L}_\lambda^*(x)$ according to Equation 5. Set $\lambda \ge 2L_f/\mu$, then it holds that*

a. $\|\nabla^2 \mathcal{L}_\lambda^*(x) - \nabla^2 \varphi(x)\| = \mathcal{O}(\ell\kappa^6/\lambda),\ \forall x \in \mathbb{R}^{d_x}$. (Lemma C.3)

b. $\mathcal{L}_\lambda^*(x)$ *is* $\mathcal{O}(\ell\kappa^5)$*-Hessian Lipschitz.* (Lemma C.2)

These lemmas are the higher-order version of Lemma 4.1, but the proof is much more difficult because $\nabla^2\varphi(x)$ is very complex and contains third-order derivatives. All the complete proof can be found in Appendix C. Based on these lemmas, we can prove the convergence of perturbed F^{2}BA in the following theorem.

Theorem 5.1 *Suppose both Assumption 3.1 and 3.2 hold. Define* $\Delta := \varphi(x_0) - \inf_{x\in\mathbb{R}^{d_x}} \varphi(x)$ *and* $R := \|y_0 - y^*(x_0)\|^2$. *Let* $\eta_x \asymp \ell^{-1}\kappa^{-3}$, $\lambda \asymp \max\left\{\kappa/R,\ \ell\kappa^2/\Delta,\ \ell\kappa^3/\epsilon,\ \kappa^{3.5}\sqrt{\ell/\epsilon}\right\}$ *and set other parameters in Algorithm 1 as*

$$\eta_z = \frac{1}{L_g},\ \eta_y = \frac{1}{2\lambda L_g},\ r = \mathcal{O}(\epsilon),\ K_t = \tilde{\mathcal{O}}\left(\frac{L_g \log \delta_t}{\mu}\right),$$

where δ_t *is defined via the recursion*

$$\delta_{t+1} = \frac{1}{2}\delta_t + \frac{34L_g^2}{\mu^2}\|x_{t+1} - x_t\|^2, \quad \delta_0 = \mathcal{O}(R), \tag{8}$$

then it can find an ϵ*-second-order stationary point of* $\varphi(x)$ *with probability at least* $1-\delta$ *within* $\tilde{\mathcal{O}}\left(\ell\kappa^4\epsilon^{-2}\right)$ *first-order oracle calls, where* ℓ, κ *are defined in Definition 3.3 and the notation* $\tilde{\mathcal{O}}(\,\cdot\,)$ *hides logarithmic factors of* $d_x, \kappa, \ell,$ *and* δ, ϵ.

We defer the proof to Appendix F. The above complexity for finding ϵ-second-order stationary points matches that for finding ϵ-first-order stationary points (Theorem 4.1), up to logarithmic factors. Therefore, we conclude that F^{2}BA can escape saddle points almost for free by simply adding some small perturbation in each step.

We remark that it is also possible to design the perturbed version of F^{2}BSA to escape saddle points using only stochastic gradient oracles via existing techniques (Xu et al., 2018; Allen-Zhu and Li, 2018). We leave them as potential future extensions.

5.2 Accelerated F^{2}BA

Compared to F^{2}BA, the perturbed version relies on the additional Assumption 3.2 to ensure the Hessian Lipschitz continuity of $\varphi(x)$. This addition assumption not only allows escaping saddle points, but also makes further acceleration being possible.

In this section, we combine our F^{2}BA with the recently proposed acceleration technique for nonconvex optimization (Li and Lin, 2023) to achieve a faster rate of $\tilde{\mathcal{O}}(\epsilon^{-1.75})$ for Hessian Lipschitz objectives. This accelerated F^{2}BA algorithm is presented in Algorithm 4. The difference to Algorithm 1 is the uses of Nesterov's momentum (Nesterov, 1983) in x (Line 3), and the restart strategy by (Li and Lin, 2023) in Line 13-15.

Theorem 5.2 *Suppose both Assumption 3.1 and 3.2 hold. Define* $\Delta := \varphi(x_0) - \inf_{x\in\mathbb{R}^{d_x}} \varphi(x)$ *and* $R := \|y_0 - y^*(x_0)\|^2$. *Let* $\eta_x \asymp \ell^{-1}\kappa^{-3}$, $\lambda \asymp \max\left\{\kappa/R,\ \ell\kappa^2/\Delta,\ \ell\kappa^3/\epsilon,\ \kappa^{3.5}\sqrt{\ell/\epsilon}\right\}$ *and set other parameters in Algorithm 1 as*

$$\eta_z = \eta_y = \frac{1}{2\lambda L_g}, \quad K_t = \tilde{\mathcal{O}}\left(\frac{L_g \log \delta_t}{\mu}\right),$$
$$T \asymp \frac{\chi}{\theta},\ B \asymp \frac{1}{\chi^2}\sqrt{\frac{\epsilon}{\ell\kappa^3}}, \quad \theta \asymp \left(\frac{\ell\epsilon}{\kappa}\right)^{1/4},\ r = \mathcal{O}(\epsilon),$$

where $\chi = \mathcal{O}(\log(d_x/\delta\epsilon))$, *where* δ_t *is defined via the recursion*

$$\delta_{t+1} = \frac{1}{2}\delta_t + \frac{34L_g^2}{\mu^2}\|x_{t+1/2} - x^-_{t+1/2}\|^2, \quad \delta_0 = \mathcal{O}(R), \tag{9}$$

Algorithm 4 Accelerated $\mathrm{F}^2\mathrm{BA}(x_0, y_0)$

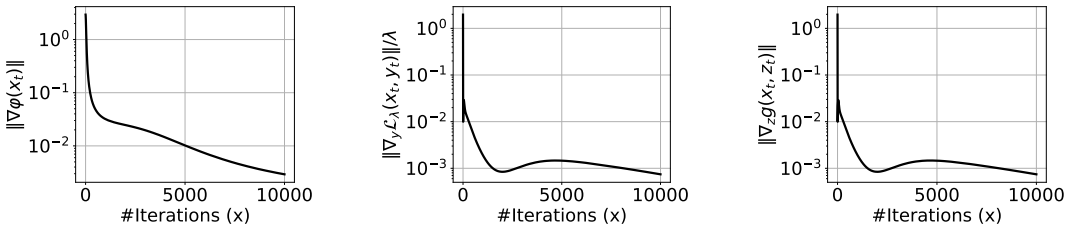
$z_0 = y_0$, $x_{-1} = x_0$
while $t < T$
  $x_{t+1/2} = x_t + (1-\theta)(x_t - x_{t-1})$
  $y_t^0 = y_t,\ z_t^0 = z_t$
  for $k = 0, 1, \cdots, K_t - 1$
    $z_t^{k+1} = z_t^k - \eta_z \lambda \nabla_y g(x_{t+1/2}, z_t^k)$
    $y_t^{k+1} = y_t^k - \eta_y \left(\nabla_y f(x_{t+1/2}, y_t^k) + \lambda \nabla_y g(x_{t+1/2}, y_t^k)\right)$
  end for
  $\hat{\nabla}\mathcal{L}_\lambda^*(x_{t+1/2}) = \nabla_x f(x_{t+1/2}, y_t^{K_t}) + \lambda(\nabla_x g(x_{t+1/2}, y_s^{K_t}) - \nabla_x g(x_{t+1/2}, z_t^{K_t}))$
  $x_{t+1/2}^- = x_{t+1/2}$
  $x_{t+1} = x_{t+1/2} - \eta_x \hat{\nabla}\mathcal{L}_\lambda^*(x_{t+1/2})$
  $t = t + 1$
  if $t \sum_{j=0}^{t-1} \|x_{t+1} - x_t\|^2 \geq B^2$
    $t = 0$, $x_{-1} = x_0 = x_t + \xi_t \mathbf{1}_{\|\hat{\nabla}\mathcal{L}_\lambda^*(x_{t+1/2})\| \leq \frac{B}{2\eta_x}}$, where $\xi_t \sim \mathbb{B}(r)$
  end if
end while
$T_0 = \arg\min_{\lfloor \frac{T}{2} \rfloor \leq t \leq T-1} \|x_{t+1} - x_t\|$
return $x_{\text{out}} = \frac{1}{T_0+1} \sum_{t=0}^{T_0} x_{t+1/2}$

then it can find an ϵ-second-order stationary point of $\varphi(x)$ with probability at least $1-\delta$ within $\tilde{\mathcal{O}}\left(\kappa \ell^{1/2} \rho^{1/4} \epsilon^{-1.75}\right) = \tilde{\mathcal{O}}\left(\kappa^{3.75} \epsilon^{-1.75}\right)$ first-order oracle calls, where ℓ, κ are defined in Definition 3.3 and the notation $\tilde{\mathcal{O}}(\cdot)$ hides logarithmic factors of d_x, κ, ℓ, and δ, ϵ.

The $\tilde{\mathcal{O}}(\epsilon^{-1.75})$ complexity matches the state-of-the-art methods for Hessian Lipschitz objectives (Jin et al., 2018; Carmon et al., 2017; Li and Lin, 2023). However, it is still an open problem whether this rate is optimal because there still exists a gap between this upper bound and the current best $\Omega(\epsilon^{-1.714})$ lower bound by Carmon et al. (2021).

6. Experiments

We conduct experiments to showcase the superiority of our proposed methods. We implement the algorithms using PyTorch (Paszke et al., 2019). In all the experiments, we tune tune the step size in the grid $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}\}$ and present the best result of each algorithm.



(a) Convergence of x (b) Convergence of y (c) Convergence of z

Figure 1: Numerical verification of F^{2}BA on Problem (10).

6.1 Tuning a Single Regularizer on Linear Regression

Let $\mathcal{D}^{\text{tr}} = (A^{\text{tr}}, b^{\text{tr}})$ and $\mathcal{D}^{\text{val}} = (A^{\text{val}}, b^{\text{val}})$ are the training and validation sets respectively, We first validate the convergence rate suggested by the theory on a simple problem:

$$\begin{aligned} &\min_{x\in\mathbb{R}} \frac{1}{2}\|A^{\text{val}} y^*(x) - b^{\text{val}}\|^2, \\ &\text{where } y^*(x) = \arg\min_{y\in\mathbb{R}^p} \frac{1}{2}\|A^{\text{tr}} y - b^{\text{tr}}\|^2 + \frac{\sigma(x)}{2}\|y\|^2, \end{aligned} \tag{10}$$

where $\sigma(x) = \exp(x)$. We use the "abalone" dataset[1], which contains 4,177 samples and each sample has 8 features ($p = 8$). We split the dataset into the training set and the validation set in a 7:3 ratio. For this problem, we can explicitly solve $y^*(x)$ and calculate $\nabla\varphi(x)$, $\nabla_y\mathcal{L}_\lambda(x, y)$, $\nabla_y g(x, z)$ to measure the convergence the algorithm. We run F^{2}BA (Algorithm 1) with $K = 10$, $\lambda = 10^3$, and the results are shown in Figure 6. It can be seen from the figure that the upper-level variable x converges to a stationary point of the hyper-objective φ, and the lower-level variables y and z converges to the minimizers $y^*_\lambda(x)$ and $y^*(x)$, respectively.

6.2 Tuning 100,000 Regularizers on Logistic Regression

We then compare our deterministic methods on the "learnable regularization" problem of logistic regression. The aim is to find the optimal regularizer for each feature separately. The corresponding bilevel problem formulation is:

$$\begin{aligned} &\min_{x\in\mathbb{R}^p} \left\{ \frac{1}{|\mathcal{D}^{\text{val}}|} \sum_{(a_i,b_i)\in\mathcal{D}^{\text{val}}} \ell(\,\langle a_i, y^*(x)\rangle\,, b_i) \right\}, \\ &\text{where } y^*(x) = \arg\min_{y\in\mathbb{R}^p} \left\{ \frac{1}{|\mathcal{D}^{\text{tr}}|} \sum_{(a_i,b_i)\in\mathcal{D}^{\text{tr}}} \ell\,(\,\langle a_i, y\rangle\,, b_i) + y^\top \Sigma(x)\, y \right\}, \end{aligned} \tag{11}$$

where $\ell(\,\cdot\,,\,\cdot\,)$ is the cross-entropy loss and $\Sigma(x) := \text{diag}(\exp(x))$ determines the regularizers on each feature. We conduct the experiments on the "20 newsgroup" dataset which is

1. Dataset available at `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`

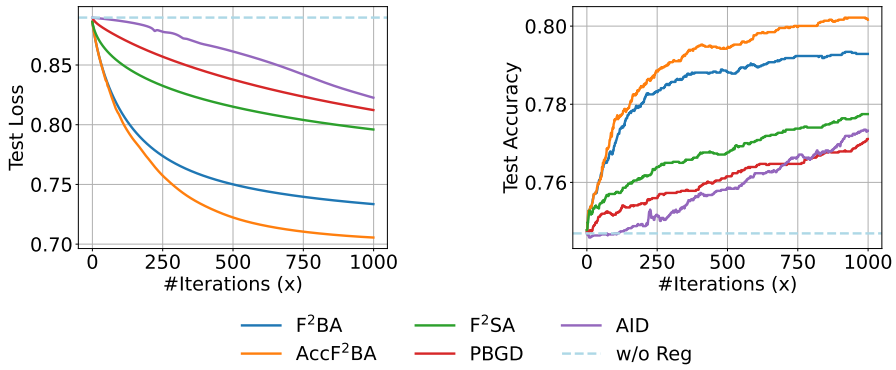


Figure 2: Comparison of deterministic algorithms on Problem (11). Our proposed algorithms F^{2}BA as well as its acceleration AccF^{2}BA outperform baselines.

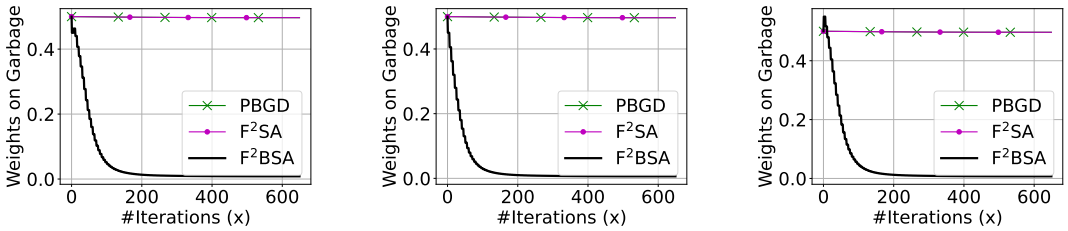
commonly used by previous works (Grazzi et al., 2020; Ji et al., 2021). This dataset is a collection of 18,000 newsgroup documents from 20 different newsgroups. Each document is represented by a p-dimensional vector ($p = 101,631$), where each dimension represents the TF-IDF (Term Frequency-Inverse Document Frequency) of a word.

We compare F^{2}BA (Algorithm 1) as well as its acceleration AccF^{2}BA (Algorithm 4) with a HVP-based method AID (Grazzi et al., 2020; Ji et al., 2021) and recently proposed gradient-based methods including F^{2}SA (Kwon et al., 2023), PBGD (Shen and Chen, 2023). We set the number of inner loops as 10 for all the algorithms, and additionally tune λ from $\{10^1, 10^2, 10^3, 10^4\}$ for penalty based methods F^{2}BA, AccF^{2}BA, F^{2}SA, and PBGD. For AccF^{2}BA, we set $\theta = 0.1$ as commonly used in momentum-based gradient methods. Instead of using a fixed B in the restart mechanism of AccF^{2}BA that may lead to very frequent restarts, we follow the practical implementation suggestion by Li and Lin (2023) to use $B = \max\{B, B_0\}$ with B_0 decaying by a constant factor γ after each restart. According to Theorem 2 (Li and Lin, 2023), this modified algorithm would have the same convergence rate as the original algorithm except for an additional $\mathcal{O}(\log(B_0/\epsilon))$ factor. We take $\gamma = 0.99$ in our experiments.

We present the results in Figure 2, where the dashed line (labeled with "w/o Reg") represents the result without tuning any regularizers. It can be seen from Figure 2 that our proposed F^{2}BA outperforms all existing baselines, and can ever achieve a better performance after acceleration (AccF^{2}BA). It is also interesting that the HVP-based method AID performs badly in this experiment. We think the reason is that $\nabla^2_{yy} g(x, y)$ in our problem can be very close to singular, which makes it very difficult to calculate its inverse numerically. In contrast, penalty-based methods do not suffer from the numerical instability.

6.3 Data Hyper-Cleaning for GPT-2

To demonstrate the scalability of our proposed method, we consider a large-scale data hyper-cleaning problem similar to the setting in (Pan et al., 2024). Let $y \in \mathbb{R}^p$ be the parameters of a neural network, which in our experiment is a GPT-2 model with 124M



(a) $p = 0.5$ (b) $p = 0.9$ (c) $p = 0.99$

Figure 3: Comparison of stochastic algorithms on Problem (12) under different corruption ratios p. Due to the use of two-time-scale step size, our proposed F^{2}BSA significantly outperforms single-time-scale baselines PBGD and F^{2}SA.

parameters (Radford et al., 2019). The model is trained on a dataset from multiple sources. The loss on each individual data source is denoted as $\ell_{\text{tr}}^i(\cdot)$, where $i = 1, \cdots, m$ and m is the total number of data sources. The goal is to find the optimal weights of different data sources (parameterized by $x \in \mathbb{R}^m$), such that the trained model can have low validation loss $\ell_{\text{val}}(\cdot)$ This problem can be formulated as a bilevel optimization problem with upper-level and lower-level functions given by:

$$\begin{aligned} f(x, y) &:= \ell_{\text{val}}(y), \\ g(x, y) &:= \sum_{i=1}^{m} \sigma(x_i) \ell_{\text{tr}}^i(y). \end{aligned} \tag{12}$$

where $\sigma(x_i)$ is the Softmax function such that $\sigma(x_i) = \exp(x_i)/\sum_{j=1}^m \exp(x_j)$.

In our experiment, we use the "Alpaca" dataset (Taori et al., 2023), which contains 52k instructions and demonstration outputs generated by OpenAI's "text-davinci-003" engine. We split the dataset into a training set and a validation set in an $8:2$ ratio, and then corrupt the training set with a proportion of p. The corruption is done by replacing the demonstration outputs with an empty string "". This naturally divides the dataset into two sources: useful data and garbage data. We expect the algorithm can learn to assign zero weights to the garbage data and use it to measure the performance of the algorithm.

We run the experiments on 8 $\times$ A40 GPUs. We compared our proposed stochastic method F^{2}BSA with related works F^{2}SA (Kwon et al., 2023) and PBGD (Shen and Chen, 2023). We do not compare any HVP-based methods as we encounter difficulties in obtaining HVP oracles in multi-GPU training (also see Appendix G). Although the original PBGD (Shen and Chen, 2023) only focuses on the deterministic case, we also implemented a stochastic version of PBGD. For all the algorithms, we set batch size as 64, and a fixed penalty with $\lambda = 10^3$. The results of different corruption ratios p is shown in Figure 6.3. It can be seen from the figure that our proposed F^{2}BSA converges much faster than baselines F^{2}SA (Kwon et al., 2023) and PBGD (Shen and Chen, 2023). The reason is that the unreasonable choice of single-time-scale step sizes in baselines (the setting of $\zeta = 1$ in (5) (Kwon et al., 2023) and the joint update of (x, y) in Line 4, Algorithm 1 (Shen and Chen, 2023)) may significantly slow down the optimization process. This also reaffirms the importance

of using the two-time-scale step size. Therefore, whether from a theoretical or a practical standpoint, we recommend always using the two-time-scale approach for penalty methods.

7. Conclusions and Future Directions

This paper proposes a two-time-scale gradient-based method which achieves the the near-optimal complexity in nonconvex-strongly-convex bilevel optimization. Our result closes the gap between gradient-based and HVP-based methods. We also study stochastic extension of this algorithm, its acceleration, and its ability to escape saddle points. The algorithms we proposed improve the current best-known guarantees under various settings. We conclude this paper with several potential directions for future research.

Logarithmic Factors in the Deterministic Case. The complexity of F^{2}BA has an additional $\log(1/\epsilon)$ factor compared with the lower bound for finding first-order stationary points (Carmon et al., 2020). Although this factor can be shaved for HVP-based methods using tighter analysis (Ji et al., 2021), it remains open whether it is avoidable for penalty methods considered in this paper.

Optimality in the Stochastic Case. The $\tilde{\mathcal{O}}(\epsilon^{-6})$ upper bound by F^{2}BSA improves the previous $\tilde{\mathcal{O}}(\epsilon^{-7})$ result by Kwon et al. (2023). However, it is unknown whether this complexity is optimal for gradient-based methods. Recently, Kwon et al. (2024a) proved an $\Omega(\epsilon^{-6})$ lower bound for stochastic optimization with an $\mathcal{O}(\epsilon)$-y^*-aware oracle, which is related to but different from stochastic bilevel problems. Tight lower bounds for stochastic bilevel problems remains open, as also pointed out by Kwon et al. (2024a).

Constrained Bilevel Problems. In this work, we only consider the unconstrained case, *i.e.* we have $x \in \mathbb{R}^{d_x}$ and $y \in \mathbb{R}^{d_y}$. It would also be important to study the convergence of gradient-based algorithms for constrained problems (Khanduri et al., 2023; Tsaknakis et al., 2022; Xiao et al., 2023; Kwon et al., 2024b; Kornowski et al., 2024).

Single-Loop Methods. Our method adopts a double-loop structure. Designing single-loop methods could be more efficient in certain scenarios, but the analysis would also be more challenging (Hong et al., 2023; Khanduri et al., 2021; Chen et al., 2022, 2021, 2024b; Dagréou et al., 2022). We also leave it in future works.

Acknowledgments

Lesi Chen thanks Jeongyeol Kwon for helpful discussions. Jingzhao Zhang is supported by National Key R&D Program of China 2024YFA1015800 and Shanghai Qi Zhi Institute Innovation Program.

Appendix A. Notations for Tensors and Derivatives

We follow the notations of tensors used in Kolda and Bader (2009). For a three-way tensor $\mathcal{X} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$, we use $[\mathcal{X}]_{i_1,i_2,i_3}$ to represent its (i_1, i_2, i_3)-th element. The inner product of two three-way tensors $\mathcal{X}, \mathcal{Y}$ is defined by $\langle \mathcal{X}, \mathcal{Y} \rangle := \sum_{i_1,i_2,i_3} [\mathcal{X}]_{i_1,i_2,i_3} \cdot [\mathcal{Y}]_{i_1,i_2,i_3}$. The operator norm of three-way tensor $\mathcal{X} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ is defined by $\|\mathcal{X}\| := \sup_{\|x_i\|=1} \langle \mathcal{X}, x_1 \circ x_2 \circ x_3 \rangle$, where the elements of $x_1 \circ x_2 \circ x_3 \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ is $[x_1 \circ x_2 \circ x_3]_{i_1,i_2,i_3} := [x_1]_{i_1} \cdot [x_2]_{i_2} \cdot [x_3]_{i_3}$. It can be verified that such a definition generalizes the Euclidean norm of a vector and the spectral norm of a matrix to tensors. For a three-way tensor $\mathcal{X} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ and a vector $v \in \mathbb{R}^{d_1}$, their mode-1 product, denoted by $\mathcal{X} \bar{\times}_1 v$, gives a matrix in $\mathbb{R}^{d_2 \times d_3}$ with elements $[\mathcal{X} \bar{\times}_1 v]_{i_2,i_3} := \sum_{i_1} [\mathcal{X}]_{i_1,i_2,i_3} \cdot [v]_{i_1}$. We define $\bar{\times}_2$ and $\bar{\times}_3$ in a similar way, and it can be verified that $\|\mathcal{X} \bar{\times}_i v\| \leq \|\mathcal{X}\| \|v\|$. For a three-way tensor $\mathcal{X} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ and a matrix $A \in \mathbb{R}^{d_1' \times d_1}$, their mode-1 product, denoted by $\mathcal{X} \times_1 A$, gives a tensor in $\mathbb{R}^{d_1' \times d_2 \times d_3}$ with elements $[\mathcal{X} \times_1 A]_{i_1',i_2,i_3} := \sum_{i_1} [\mathcal{X}]_{i_1,i_2,i_3} \cdot [A]_{i_1',i_1}$. We define $\times_2$ and $\times_3$ in a similar way, and it can also be verified that $\|\mathcal{X} \times_i A\| \leq \|\mathcal{X}\| \|A\|$.

For a function $h(x, y) : \mathbb{R}^{d_x} \times \mathbb{R}^{d_y} \to \mathbb{R}$, we denote $\nabla_x h(x, y) \in \mathbb{R}^{d_x}$ to be the partial gradient with respect to x, with elements given by $[\nabla_x h(x, y)]_i := \partial f(x, y)/\partial [x]_i$. And we define $\nabla_y h(x, y) \in \mathbb{R}^{d_y}$ in a similar way. We denote $\nabla^2_{xx} h(x, y) \in \mathbb{R}^{d_x} \times \mathbb{R}^{d_x}$ to be the partial Hessian with respect to x, with elements given by $[\nabla^2_{xx} h(x, y)]_{i_1,i_2} := \partial^2 h(x, y)/(\partial [x]_{i_1} \partial [x]_{i_2})$. And we define $\nabla^2_{xy} h(x, y) \in \mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$, $\nabla^2_{yx} h(x, y) \in \mathbb{R}^{d_y} \times \mathbb{R}^{d_x}$ and $\nabla^2_{yy} h(x, y) \in \mathbb{R}^{d_y} \times \mathbb{R}^{d_y}$ in a similar way. We denote $\nabla^3_{xxx} h(x, y) \in \mathbb{R}^{d_x} \times \mathbb{R}^{d_x} \times \mathbb{R}^{d_x}$ to be the partial third-order derivative with respect to x, with elements given by $[\nabla^3_{xxx} h(x, y)]_{i_1,i_2,i_3} := \partial^3 h(x, y)/(\partial [x]_{i_1} \partial [x]_{i_2} \partial [x]_{i_3})$. And we define $\nabla^3_{xxy} h(x, y) \in \mathbb{R}^{d_x} \times \mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$, $\nabla^3_{xyx} h(x, y) \in \mathbb{R}^{d_x} \times \mathbb{R}^{d_y} \times \mathbb{R}^{d_x}$, $\nabla^3_{xyy} h(x, y) \in \mathbb{R}^{d_x} \times \mathbb{R}^{d_y} \times \mathbb{R}^{d_y}$, $\nabla^3_{yyy} h(x, y) \in \mathbb{R}^{d_y} \times \mathbb{R}^{d_y} \times \mathbb{R}^{d_y}$, $\nabla^3_{yyx} h(x, y) \in \mathbb{R}^{d_y} \times \mathbb{R}^{d_y} \times \mathbb{R}^{d_x}$, $\nabla^3_{yxy} h(x, y) \in \mathbb{R}^{d_y} \times \mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$ and$\nabla^3_{yxx} h(x, y) \in \mathbb{R}^{d_y} \times \mathbb{R}^{d_x} \times \mathbb{R}^{d_x}$ in a similar way. We denote $\nabla y^*(x) \in \mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$ to be the matrix with elements given by $[\nabla y^*(x)]_{i_1,i_2} := \partial [y^*(x)]_{i_2} / \partial [x]_{i_1}$. We denote $\nabla^2 y^*(x) \in \mathbb{R}^{d_x} \times \mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$ to be the three-way tensor with elements given by $[\nabla^2 y^*(x)]_{i_1,i_2,i_3} := \partial^2 [y^*(x)]_{i_3} / (\partial [x]_{i_1} \partial [x]_{i_2})$. And we define $\nabla y^*_\lambda(x)$ and $\nabla^2 y^*_\lambda(x)$ in a similar way.

Appendix B. Lemmas for Finding First-Order Stationarity

Lemma B.1 (Kwon et al. (2023)) *Under Assumption 3.1, for $\lambda \geq 2L_f/\mu$, $\mathcal{L}_\lambda(x, \cdot)$ is $(\lambda\mu/2)$-strongly convex.*

Lemma B.2 *Under Assumption 3.1, for $\lambda \geq 2L_f/\mu$, it holds that*

$$\|y^*_\lambda(x) - y^*(x)\| \leq \frac{C_f}{\lambda\mu}$$

Proof By the first-order optimality condition, we know that

$$\nabla_y f(x, y^*_\lambda(x)) + \lambda \nabla_y g(x, y^*_\lambda(x)) = 0.$$

Then we have

$$\|y^*_\lambda(x) - y^*(x)\| \leq \frac{1}{\mu} \|\nabla_y g(x, y^*_\lambda(x))\| = \frac{1}{\lambda\mu} \|\nabla_y f(x, y^*_\lambda(x))\| \leq \frac{C_f}{\lambda\mu}$$

■

As a direct consequence, we can show that $\mathcal{L}_x^*$ and $\varphi(x)$ are close.

Lemma B.3 *Under Assumption 3.1, for $\lambda \geq 2L_f/\mu$, it holds that $|\mathcal{L}_\lambda^*(x) - \varphi(x)| \leq D_0/\lambda$, where*

$$D_0 := \left(C_f + \frac{C_f L_g}{2\mu}\right)\frac{C_f}{\mu} = \mathcal{O}(\ell\kappa^2).$$

Proof A simple calculus shows that

$$\begin{aligned}
&|\mathcal{L}_\lambda^*(x) - \varphi(x)| \\
&\leq |f(x, y_\lambda^*(x)) - f(x, y^*(x))| + \lambda |g(x, y_\lambda^*(x)) - g(x, y^*(x))| \\
&\leq C_f \|y_\lambda^*(x) - y^*(x)\| + \frac{\lambda L_g}{2}\|y_\lambda^*(x) - y^*(x)\|^2 \\
&\leq \left(C_f + \frac{C_f L_g}{2\mu}\right)\|y_\lambda^*(x) - y^*(x)\| \\
&\leq \left(C_f + \frac{C_f L_g}{2\mu}\right)\frac{C_f}{\lambda\mu}.
\end{aligned}$$

■

The following result is the key to designing fully first-order methods for bilevel optimization, first proved in Lemma 3.1 in Kwon et al. (2023). We provide the proof here for completeness.

Lemma B.4 *Under Assumption 3.1, for $\lambda \geq 2L_f/\mu$, it holds that $\|\nabla\mathcal{L}_\lambda^*(x) - \nabla\varphi(x)\| \leq D_1/\lambda$, where*

$$D_1 := \left(L_f + \frac{\rho_g L_g}{\mu} + \frac{C_f L_g \rho_g}{2\mu^2} + \frac{C_f \rho_g}{2\mu}\right)\frac{C_f}{\mu} = \mathcal{O}(\ell\kappa^3). \tag{13}$$

Proof Taking total derivative on $\varphi(x) = f(x, y^*(x))$, we obtain the following result:

$$\nabla\varphi(x) = \nabla_x f(x, y^*(x)) - \nabla_{xy}^2 g(x, y^*(x))[\nabla_{yy} g(x, y^*(x))]^{-1}\nabla_y f(x, y^*(x)). \tag{14}$$

Also, we know that

$$\nabla\mathcal{L}_\lambda^*(x) = \nabla_x f(x, y_\lambda^*(x)) + \lambda(\nabla_x g(x, y_\lambda^*(x)) - \nabla_x g(x, y^*(x))).$$

By simple calculus, we have

$$\begin{aligned}
&\nabla\mathcal{L}_\lambda^*(x) - \nabla\varphi(x) \\
&= \nabla_x f(x, y_\lambda^*(x)) - \nabla_x f(x, y^*(x)) \\
&\quad + \nabla_{xy}^2 g(x, y^*(x))[\nabla_{yy} g(x, y^*(x))]^{-1}(\nabla_y f(x, y^*(x)) - \nabla_y f(x, y_\lambda^*(x)) \\
&\quad + \lambda\nabla_{xy}^2 g(x, y^*(x))[\nabla_{yy} g(x, y^*(x))]^{-1}\times \\
&\qquad\qquad (\nabla_{yy}^2 g(x, y^*(x))(y_\lambda^*(x) - y^*(x)) + \nabla_y g(x, y^*(x)) - \nabla_y g(x, y_\lambda^*(x)))
\end{aligned}$$

$$+ \lambda(\nabla_x g(x, y_\lambda^*(x)) - \nabla_x g(x, y^*(x)) - \nabla_{xy}^2 g(x, y^*(x))(y_\lambda^*(x) - y^*(x))).$$

Taking norm on both sides,

$$\begin{aligned}
\|\nabla \mathcal{L}_\lambda^*(x) - \nabla \varphi(x)\| &\le L_f \|y_\lambda^*(x) - y^*(x)\| + \frac{\rho_g L_g}{\mu} \|y_\lambda^*(x) - y^*(x)\| \\
&\quad + \frac{\lambda L_g \rho_g}{2\mu} \|y_\lambda^*(x) - y^*(x)\|^2 + \frac{\lambda \rho_g}{2} \|y_\lambda^*(x) - y^*(x)\|^2 \\
&\le \left(L_f + \frac{\rho_g L_g}{\mu} + \frac{C_f L_g \rho_g}{2\mu^2} + \frac{C_f \rho_g}{2\mu} \right) \|y_\lambda^*(x) - y^*(x)\| \\
&\le \left(L_f + \frac{\rho_g L_g}{\mu} + \frac{C_f L_g \rho_g}{2\mu^2} + \frac{C_f \rho_g}{2\mu} \right) \frac{C_f}{\lambda \mu}
\end{aligned}$$

■

Lemma B.5 *Under Assumption 3.1, for $\lambda \ge 2L_f/\mu$, it holds that $\|\nabla y^*(x) - \nabla y_\lambda^*(x)\| \le D_2/\lambda$, where*

$$D_2 := \left(\frac{1}{\mu} + \frac{2L_g}{\mu^2} \right) \left(L_f + \frac{C_f \rho_g}{\mu} \right) = \mathcal{O}(\kappa^3).$$

Proof Taking derivative on both sides of $\nabla_y g(x, y^*(x)) = 0$ yields

$$\nabla_{xy}^2 g(x, y^*(x)) + \nabla y^*(x) \nabla_{yy}^2 g(x, y^*(x)) = 0. \tag{15}$$

Rearranging, we can obtain

$$\nabla y^*(x) = -\nabla_{xy}^2 g(x, y^*(x)) [\nabla_{yy}^2 g(x, y^*(x))]^{-1}. \tag{16}$$

Similarly, we also have

$$\nabla y_\lambda^*(x) = -\nabla_{xy}^2 \mathcal{L}_\lambda(x, y_\lambda^*(x)) [\nabla_{yy}^2 \mathcal{L}_\lambda(x, y_\lambda^*(x))]^{-1}. \tag{17}$$

Using the matrix identity $A^{-1} - B^{-1} = A^{-1}(B - A)B^{-1}$, we have

$$\begin{aligned}
&\left\| [\nabla_{yy}^2 g(x, y^*(x))]^{-1} - \left[\frac{\nabla_{yy}^2 \mathcal{L}_\lambda(x, y_\lambda^*(x))}{\lambda} \right]^{-1} \right\| \\
&\le \left\| [\nabla_{yy}^2 g(x, y_\lambda^*(x))]^{-1} \right\| \left\| \frac{\nabla_{yy}^2 \mathcal{L}_\lambda(x, y_\lambda^*(x))}{\lambda} - \nabla_{yy}^2 g(x, y^*(x)) \right\| \left\| \left[\frac{\nabla_{yy}^2 \mathcal{L}_\lambda(x, y_\lambda^*(x))}{\lambda} \right]^{-1} \right\| \\
&\le \frac{2}{\mu^2} \left\| \frac{\nabla_{yy}^2 f(x, y_\lambda^*(x))}{\lambda} + \nabla_{yy}^2 g(x, y_\lambda^*(x)) - \nabla_{yy}^2 g(x, y^*(x)) \right\| \\
&\le \frac{2}{\mu^2} \left(\frac{L_f}{\lambda} + \rho_g \|y_\lambda^*(x) - y^*(x)\| \right) \\
&\le \frac{2}{\lambda \mu^2} \left(L_f + \frac{C_f \rho_g}{\mu} \right).
\end{aligned} \tag{18}$$

Note that the setting of $\lambda \geq 2L_f/\mu$ implies $\|\nabla^2_{xy}\mathcal{L}(\,\cdot\,,\,\cdot\,)\| \leq 2\lambda L_g$, then we further have

$$\begin{aligned}
&\|\nabla y^*_\lambda(x) - \nabla y^*(x)\| \\
&\leq \left\|\nabla^2_{xy} g(x, y^*(x)) - \frac{\nabla^2_{xy}\mathcal{L}_\lambda(x, y^*_\lambda(x))}{\lambda}\right\| \left\|[\nabla^2_{yy} g(x, y^*(x))]^{-1}\right\| \\
&\quad + \left\|\frac{\nabla^2_{xy}\mathcal{L}_\lambda(x, y^*_\lambda(x))}{\lambda}\right\| \left\|[\nabla^2_{yy} g(x, y^*(x))]^{-1} - \left[\frac{\nabla^2_{yy}\mathcal{L}_\lambda(x, y^*_\lambda(x))}{\lambda}\right]^{-1}\right\| \\
&\leq \frac{1}{\mu}\left\|\nabla^2_{xy} g(x, y^*(x)) - \nabla^2_{xy} g(x, y^*_\lambda(x)) - \frac{\nabla^2_{xy} f(x, y^*_\lambda(x))}{\lambda}\right\| + \frac{2L_g}{\lambda\mu^2}\left(L_f + \frac{C_f\rho_g}{\mu}\right) \\
&\leq \left(\frac{1}{\lambda\mu} + \frac{2L_g}{\lambda\mu^2}\right)\left(L_f + \frac{C_f\rho_g}{\mu}\right).
\end{aligned}$$

■

It is clear that $\|\nabla y^*(x)\| \leq L_g/\mu$, therefore $y^*(x)$ is (L_g/μ)-Lipschitz. Below, we show that a similar result also holds for $y^*_\lambda(x)$.

Lemma B.6 *Under Assumption 3.1, for $\lambda \geq 2L_f/\mu$, it holds that $\|\nabla y^*_\lambda(x)\| \leq 4L_g/\mu$.*

Proof Recall Equation 17 that

$$\nabla y^*_\lambda(x) = -\nabla^2_{xy}\mathcal{L}_\lambda(x, y^*_\lambda(x))[\nabla^2_{yy}\mathcal{L}_\lambda(x, y^*_\lambda(x))]^{-1}.$$

We arrive at the conclusion by noting that $\nabla^2_{xy}\mathcal{L}_\lambda(\,\cdot\,,\,\cdot\,) \preceq 2\lambda L_g$ and $\nabla^2_{yy}\mathcal{L}_\lambda(\,\cdot\,,\,\cdot\,) \succeq \lambda\mu/2$ by Lemma B.1.

■

This implies that $y^*_\lambda(x)$ is $(4L_g/\mu)$-Lipschitz.

Lemma B.7 *Under Assumption 3.1, for $\lambda \geq 2L_f/\mu$, $\nabla\mathcal{L}^*_\lambda(x)$ is D_3-Lipschitz, where*

$$D_3 := L_f + \frac{4L_fL_g}{\mu} + \frac{C_f\rho_g}{\mu} + \frac{C_fL_g\rho_g}{\mu^2} + L_gD_2 = \mathcal{O}(\kappa^3).$$

Proof Note that

$$\nabla\mathcal{L}^*_\lambda(x) = \underbrace{\nabla_x f(x, y^*_\lambda(x))}_{A(x)} + \underbrace{\lambda(\nabla_x g(x, y^*_\lambda(x)) - \nabla_x g(x, y^*(x)))}_{B(x)}. \tag{19}$$

where $A(x)$ and $B(x)$ are both mappings $\mathbb{R}^{d_x} \to \mathbb{R}^{d_x}$. From Lemma B.6 we know that $y^*_\lambda(x)$ is $(4L_g/\mu)$-Lipschitz. This further implies that $A(x)$ is $(1 + 4L_g/\mu)L_f$-Lipschitz. Next, we bound the Lipschitz coefficient of $B(x)$ via its derivative $\nabla B(x) : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x} \times \mathbb{R}^{d_x}$, which has the following form by taking total derivative on $B(x)$:

$$\begin{aligned}
\nabla B(x) = \lambda(&\nabla^2_{xx} g(x, y^*_\lambda(x)) - \nabla^2_{xx} g(x, y^*(x))) \\
&+ \lambda\left(\nabla y^*_\lambda(x)\nabla^2_{yx} g(x, y^*_\lambda(x)) - \nabla y^*(x)\nabla^2_{yx} g(x, y^*(x))\right).
\end{aligned} \tag{20}$$

And we can bound the operator norm of $\nabla B(x)$ by:

$$\begin{aligned}\|\nabla B(x)\| &\le \lambda\|\nabla^2_{xx} g(x, y^*_\lambda(x)) - \nabla^2_{xx} g(x, y^*(x))\| \\ &\quad + \lambda\|\nabla y^*(x)\|\|\nabla^2_{yx} g(x, y^*_\lambda(x)) - \nabla^2_{yx} g(x, y^*(x))\| \\ &\quad + \lambda\|\nabla y^*_\lambda(x) - \nabla y^*(x)\|\|\nabla^2_{yx} g(x, y^*_\lambda(x))\| \\ &\le \lambda\rho_g\left(1 + \frac{L_g}{\mu}\right)\|y^*_\lambda(x) - y^*(x)\| + \lambda L_g\|\nabla y^*_\lambda(x) - \nabla y^*(x)\| \\ &\le \left(1 + \frac{L_g}{\mu}\right)\frac{C_f\rho_g}{\mu} + L_g D_2,\end{aligned}$$

where we use Lemma B.6 in the second inequality; Lemma B.2 and B.5 in the third one. ■

Appendix C. Lemmas for Finding Second-Order Stationarity

Lemma C.1 *Under Assumption 3.1 and 3.2, for $\lambda \ge 2L_f/\mu$, we have $\|\nabla^2 y^*(x) - \nabla^2 y^*_\lambda(x)\| \le D_4/\lambda$, where*

$$D_4 := \frac{2\rho_g}{\mu^2}\left(1 + \frac{L_g}{\mu}\right)^2\left(L_f + \frac{C_f\rho_g}{\mu}\right) + \frac{14L_g\rho_g D_2}{\mu^2} + \frac{50L_g^2}{\mu^3}\left(\frac{C_f\nu_g}{\mu} + \rho_f\right) = \mathcal{O}(\kappa^5).$$

Proof First of all, we calculate the explicit form of $\nabla^2 y^*(x)$ and $\nabla^2 y^*_\lambda(x)$.

By taking the derivative with respect to x on

$$\nabla^2_{xy} g(x, y^*(x)) + \nabla y^*(x)\nabla^2_{yy} g(x, y^*(x)) = 0,$$

we obtain

$$\begin{aligned}&\nabla^3_{xxy} g(x, y^*(x)) + \nabla^3_{yxy} g(x, y^*(x)) \times_1 \nabla y^*(x) + \nabla^2 y^*(x) \times_3 \nabla^2_{yy} g(x, y^*(x)) \\ &\quad + \nabla^3_{xyy} g(x, y^*(x)) \times_2 \nabla y^*(x) + \nabla^3_{yyy} g(x, y^*(x)) \times_1 \nabla y^*(x) \times_2 \nabla y^*(x) = 0.\end{aligned}$$

Rearranging to get

$$\begin{aligned}\nabla^2 y^*(x) = &-\left(\nabla^3_{xxy} g(x, y^*(x)) + \nabla^3_{yxy} g(x, y^*(x)) \times_1 \nabla y^*(x)\right) \times_3 [\nabla^2_{yy} g(x, y^*(x))]^{-1} \\ &- \nabla^3_{xyy} g(x, y^*(x)) \times_2 \nabla y^*(x) \times_3 [\nabla^2_{yy} g(x, y^*(x))]^{-1} \\ &- \nabla^3_{yyy} g(x, y^*(x)) \times_1 \nabla y^*(x) \times_2 \nabla y^*(x) \times_3 [\nabla^2_{yy} g(x, y^*(x))]^{-1}.\end{aligned} \tag{21}$$

Similarly,

$$\begin{aligned}\nabla^2 y^*_\lambda(x) = &-\left(\nabla^3_{xxy} \mathcal{L}_\lambda(x, y^*_\lambda(x)) + \nabla^3_{yxy} \mathcal{L}_\lambda(x, y^*_\lambda(x)) \times_1 \nabla y^*_\lambda(x)\right) \times_3 [\nabla^2_{yy} \mathcal{L}_\lambda(x, y^*_\lambda(x))]^{-1} \\ &- \nabla^3_{xyy} \mathcal{L}_\lambda(x, y^*_\lambda(x)) \times_2 \times_3 [\nabla^2_{yy} \mathcal{L}_\lambda(x, y^*_\lambda(x))]^{-1} \\ &- \nabla y^*_\lambda(x) + \nabla^3_{yyy} \mathcal{L}_\lambda(x, y^*_\lambda(x)) \times_1 \nabla y^*_\lambda(x) \times_2 \nabla y^*_\lambda(x) \times_3 [\nabla^2_{yy} \mathcal{L}_\lambda(x, y^*_\lambda(x))]^{-1}.\end{aligned} \tag{22}$$

We then prove $\|\nabla^2 y^*(x) - \nabla^2 y_\lambda^*(x)\| = \mathcal{O}(1/\lambda)$. We prove this by showing that the difference between each corresponding term of $\nabla^2 y^*(x)$ and $\nabla^2 y_\lambda^*(x)$ is $\mathcal{O}(1/\lambda)$. Note that

$$\left\|\nabla^3_{xxy} g(x, y^*(x)) - \frac{\nabla^3_{xxy}\mathcal{L}_\lambda(x, y_\lambda^*(x))}{\lambda}\right\| \le \nu_g \|y_\lambda^*(x) - y^*(x)\| + \frac{\rho_f}{\lambda} \le \frac{1}{\lambda}\left(\frac{C_f \nu_g}{\mu} + \rho_f\right),$$

and

$$\begin{aligned}
&\left\|\nabla^3_{yxy} g(x, y^*(x)) \times_1 \nabla y^*(x) - \frac{\nabla^3_{yxy}\mathcal{L}_\lambda(x, y_\lambda^*(x)) \times_1 \nabla y_\lambda^*(x)}{\lambda}\right\| \\
&\le \|\nabla y^*(x) - \nabla y_\lambda^*(x)\| \|\nabla^3_{yxy} g(x, y^*(x))\| + \|\nabla y_\lambda^*(x)\| \left\|\nabla^3_{yxy} g(x, y^*(x)) - \frac{\nabla^3_{yxy}\mathcal{L}_\lambda(x, y_\lambda^*(x))}{\lambda}\right\| \\
&\le \frac{\rho_g D_2}{\lambda} + \frac{4L_g}{\lambda\mu}\left(\frac{C_f \nu_g}{\mu} + \rho_f\right),
\end{aligned}$$

and

$$\begin{aligned}
&\left\|\nabla^3_{yyy} g(x, y^*(x)) \times_1 \nabla y^*(x) \times_2 \nabla y^*(x) - \frac{\nabla^3_{yyy}\mathcal{L}_\lambda(x, y_\lambda^*(x)) \times_1 \nabla y_\lambda^*(x) \times_2 \nabla y_\lambda^*(x)}{\lambda}\right\| \\
&\le \|\nabla y^*(x)\| \|\nabla^3_{yyy} g(x, y^*(x))\| \|\nabla y^*(x) - \nabla y_\lambda^*(x)\| \\
&\quad + \|\nabla y_\lambda^*(x)\| \|\nabla^3_{yyy} g(x, y^*(x))\| \|\nabla y^*(x) - \nabla y_\lambda^*(x)\| \\
&\quad + \|\nabla y_\lambda^*(x)\|^2 \left\|\nabla^3_{xxy} g(x, y^*(x)) - \frac{\nabla^3_{xxy}\mathcal{L}_\lambda(x, y_\lambda^*(x))}{\lambda}\right\| \\
&\le \frac{5L_g \rho_g D_2}{\lambda\mu} + \frac{16L_g^2}{\lambda\mu^2}\left(\frac{C_f \nu_g}{\mu} + \rho_f\right),
\end{aligned}$$

we can obtain that

$$\begin{aligned}
&\|\nabla^2 y^*(x) - \nabla^2 y_\lambda^*(x)\| \\
&\le \rho_g \left(1 + \frac{L_g}{\mu}\right)^2 \left\|[\nabla^2_{yy} g(x, y^*(x))]^{-1} - \left[\frac{\nabla^2_{yy}\mathcal{L}_\lambda(x, y_\lambda^*(x))}{\lambda}\right]^{-1}\right\| \\
&\quad + \left(\frac{7L_g \rho_g D_2}{\lambda\mu} + \frac{25L_g^2}{\lambda\mu^2}\left(\frac{C_f \nu_g}{\mu} + \rho_f\right)\right) \left\|\left[\frac{\nabla^2_{yy}\mathcal{L}_\lambda(x, y_\lambda^*(x))}{\lambda}\right]^{-1}\right\| \\
&\le \frac{2\rho_g}{\lambda\mu^2}\left(1 + \frac{L_g}{\mu}\right)^2 \left(L_f + \frac{C_f \rho_g}{\mu}\right) + \frac{14L_g \rho_g D_2}{\lambda\mu^2} + \frac{50L_g^2}{\lambda\mu^3}\left(\frac{C_f \nu_g}{\mu} + \rho_f\right),
\end{aligned}$$

where we use Equation 18 in the second inequality. ■

Lemma C.2 *Under Assumption 3.1 and 3.2, for $\lambda \ge 2L_f/\mu$, $\nabla^2 \mathcal{L}_\lambda^*(x)$ is D_5-Lipschitz, where*

$$D_5 := \left(1 + \frac{4L_g}{\mu}\right)^2 \left(3\rho_f + \frac{2L_f \rho_g}{\mu}\right) + \left(1 + \frac{L_g}{\mu}\right)^2 \frac{C_f \nu_g}{\mu}$$

$$+\left(2+\frac{5L_g}{\mu}\right)D_2\rho_g+\left(1+\frac{L_g}{\mu}\right)^2\frac{C_f\rho_g^2}{\mu^2}+L_gD_4=\mathcal{O}(\ell\kappa^5).$$

Proof Similar to the proof of Lemma B.7, we split $\nabla^2\mathcal{L}_\lambda^*(x)$ into two terms:

$$\nabla^2\mathcal{L}_\lambda^*(x)=\nabla A(x)+\nabla B(x),$$

where both the mappings $A(x)$ and $B(x)$ both follow the definitions in Equation 19. rTaking total derivative on $A(x)$, we obtain

$$\nabla A(x)=\nabla^2_{xx}f(x,y_\lambda^*(x))+\nabla y_\lambda^*(x)\nabla^2_{yx}f(x,y_\lambda^*(x)).$$

And recall Equation 20 that

$$\begin{aligned}\nabla B(x)=&\lambda(\nabla^2_{xx}g(x,y_\lambda^*(x))-\nabla^2_{xx}g(x,y^*(x)))\\&+\lambda\left(\nabla y_\lambda^*(x)\nabla^2_{yx}g(x,y_\lambda^*(x))-\nabla y^*(x)\nabla^2_{yx}g(x,y^*(x))\right).\end{aligned}$$

Then we bound the Lipschitz coefficient of $\nabla A(x)$ and $\nabla B(x)$, respectively.

From Equation 21 we can calculate that

$$\|\nabla^2y^*(x)\|\le\left(1+\frac{L_g}{\mu}\right)^2\frac{\rho_g}{\mu}. \tag{23}$$

Similarly, from Equation 22 we can calculate that

$$\|\nabla^2y_\lambda^*(x)\|\le\left(1+\frac{4L_g}{\mu}\right)^2\left(\frac{\rho_f}{\lambda}+\rho_g\right)\frac{2}{\mu}\le\left(1+\frac{4L_g}{\mu}\right)^2\left(\frac{2\rho_f}{L_f}+\frac{2\rho_g}{\mu}\right). \tag{24}$$

From Lemma B.6 we know that $y_\lambda^*(x)$ is $(4L_g/\mu)$-Lipschitz. This further implies that both $\nabla^2_{xx}f(x,y_\lambda^*(x))$ and $\nabla^2_{yx}f(x,y_\lambda^*(x))$ are $(1+4L_g/\mu)\rho_f$-Lipschitz. Then, for any $x_1,x_2\in\mathbb{R}^{d_x}$, we have

$$\begin{aligned}&\|\nabla A(x_1)-\nabla A(x_2)\|\\\le&\|\nabla^2_{xx}f(x_1,y_\lambda^*(x_1))-\nabla^2_{xx}f(x_2,y_\lambda^*(x_2))\|\\&+\|\nabla y_\lambda^*(x_1)\nabla^2_{yx}f(x_1,y_\lambda^*(x_1))-\nabla y_\lambda^*(x_2)\nabla^2_{yx}f(x_1,y_\lambda^*(x_1))\|\\&+\|\nabla y_\lambda^*(x_2)\nabla^2_{yx}f(x_1,y_\lambda^*(x_1))-\nabla y_\lambda^*(x_2)\nabla^2_{yx}f(x_2,y_\lambda^*(x_2))\|\\\le&\|\nabla^2_{xx}f(x_1,y_\lambda^*(x_1))-\nabla^2_{xx}f(x_2,y_\lambda^*(x_2))\|\\&+\|\nabla y_\lambda^*(x_1)-\nabla y_\lambda^*(x_2)\|\|\nabla^2_{yx}f(x_1,y_\lambda^*(x_1))\|\\&+\|\nabla y_\lambda^*(x_2)\|\|\nabla^2_{yx}f(x_1,y_\lambda^*(x_1))-\nabla^2_{yx}f(x_2,y_\lambda^*(x_2))\|\\\le&\underbrace{\left(\left(1+\frac{4L_g}{\mu}\right)^2\rho_f+\left(1+\frac{4L_g}{\mu}\right)^2\left(\frac{2\rho_f}{L_f}+\frac{2\rho_g}{\mu}\right)L_f\right)}_{C_1}\|x_1-x_2\|,\end{aligned}$$

where C_1 gives the upper bound of the Lipschitz coefficient of the mapping $\nabla A(x)$.

To bound the Lipschitz coefficient of $\nabla B(x)$, we first derive the explicit form of the mapping $\nabla^2 B(x) : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x} \times \mathbb{R}^{d_x} \times \mathbb{R}^{d_x}$ by:

$$\begin{aligned}
\nabla^2 B(x) = {} & \nabla^2 y_\lambda^*(x) \times_3 [\nabla^2_{yx} f(x, y_\lambda^*(x))]^\top \\
& + \lambda(\nabla^3_{xxx} g(x, y_\lambda^*(x)) - \nabla^3_{xxx} g(x, y^*(x))) \\
& + \lambda(\nabla^3_{yxx} g(x, y_\lambda^*(x)) \times_1 \nabla y_\lambda^*(x) - \nabla^3_{yxx} g(x, y^*(x)) \times_1 \nabla y^*(x)) \\
& + \lambda \left(\nabla^3_{xyx} g(x, y_\lambda^*(x)) \times_2 \nabla y_\lambda^*(x) - \nabla^3_{xyx} g(x, y^*(x)) \times_2 \nabla y^*(x)\right) \\
& + \lambda \left(\nabla^3_{yyx} g(x, y_\lambda^*(x)) \times_1 \nabla y_\lambda^*(x) \times_2 \nabla y_\lambda^*(x) - \nabla^3_{yyx} g(x, y^*(x)) \times_1 \nabla y^*(x) \times_2 \nabla y^*(x)\right) \\
& + \lambda \left(\nabla^2 y_\lambda^*(x) \times_3 [\nabla^2_{yx} g(x, y_\lambda^*(x))]^\top - \nabla^2 y^*(x) \times_3 [\nabla^2_{yx} g(x, y^*(x))]^\top\right).
\end{aligned}$$

Then we can bound the Lipschitz coefficient of $\nabla B(x)$ via the operator norm of $\nabla^2 B(x)$:

$$\begin{aligned}
C_2 := {} & \|\nabla^2 B(x)\| \\
\leq {} & \|\nabla^3_{xxx} g(x, y^*(x)) - \nabla^3_{xxx} g(x, y_\lambda^*(x))\| \\
& + \lambda \|\nabla y^*(x)\| \|\nabla^3_{yxx} g(x, y^*(x)) - \nabla^3_{yxx} g(x, y_\lambda^*(x))\| \\
& + \lambda \|\nabla y_\lambda^*(x) - \nabla y^*(x)\| \|\nabla^3_{yxx} g(x, y_\lambda^*(x))\| \\
& + \lambda \|\nabla y^*(x)\| \|\nabla^3_{xyx} g(x, y^*(x)) - \nabla^3_{xyx} g(x, y_\lambda^*(x))\| \\
& + \lambda \|\nabla y^*(x) - \nabla y_\lambda^*(x)\| \|\nabla^3_{xyx} g(x, y_\lambda^*(x))\| \\
& + \lambda \|\nabla y^*(x)\| \|\nabla^3_{yyx} g(x, y^*(x))\| \|\nabla y_\lambda^*(x) - \nabla y^*(x)\| \\
& + \lambda \|\nabla y_\lambda^*(x)\| \|\nabla^3_{yyx} g(x, y^*(x))\| \|\nabla y_\lambda^*(x) - \nabla y^*(x)\| \\
& + \lambda \|\nabla y^*(x)\|^2 \|\nabla^3_{yyx} g(x, y^*(x)) - \nabla^3_{yyx} g(x, y_\lambda^*(x))\| \\
& + \lambda \|\nabla^2 y^*(x)\| \|\nabla^2_{yx} g(x, y^*(x)) - \nabla^2_{yx} g(x, y_\lambda^*(x))\| \\
& + \lambda \|\nabla^2 y^*(x) - \nabla^2 y_\lambda^*(x)\| \|\nabla^2_{yx} g(x, y_\lambda^*(x))\|,
\end{aligned}$$

which only requires using triangle inequality multiple times. Then we plug

$$\begin{aligned}
\|y_\lambda^*(x) - y^*(x)\| &= \mathcal{O}(1/\lambda), \text{ By Lemma B.2} \\
\|\nabla y_\lambda^*(x) - \nabla y^*(x)\| &= \mathcal{O}(1/\lambda), \text{ By Lemma B.5.} \\
\|\nabla^2 y_\lambda^*(x) - \nabla^2 y^*(x)\| &= \mathcal{O}(1/\lambda), \text{ By Lemma C.1.}
\end{aligned} \tag{25}$$

and

$$\begin{aligned}
\|\nabla y^*(x)\| &= \mathcal{O}(1), \text{ By Equation 16.} \\
\|\nabla y_\lambda^*(x)\| &= \mathcal{O}(1), \text{ By Lemma B.6.} \\
\|\nabla^2 y^*(x)\| &= \mathcal{O}(1), \text{ By Equation 23.}
\end{aligned} \tag{26}$$

into the bound for C_2 to obtain that

$$\begin{aligned}
C_1 + C_2 \leq {} & \left(1 + \frac{4L_g}{\mu}\right)^2 \left(3\rho_f + \frac{2L_f \rho_g}{\mu}\right) + \left(1 + \frac{L_g}{\mu}\right)^2 \frac{C_f \nu_g}{\mu} \\
& + \left(2 + \frac{5L_g}{\mu}\right) D_2 \rho_g + \left(1 + \frac{L_g}{\mu}\right)^2 \frac{C_f \rho_g^2}{\mu^2} + L_g D_4.
\end{aligned}$$

■

Lemma C.3 *Under Assumption 3.1 and 3.2, for $\lambda \geq 2L_f/\mu$, we have $\|\nabla^2\mathcal{L}_\lambda^*(x) - \nabla^2\varphi(x)\| \leq D_6/\lambda$, where*

$$D_6 := 2L_gD_2^2 + \left(1+\frac{L_g}{\mu}\right)^2\left(\frac{C_f\rho_f}{\mu} + \frac{C_fL_f\rho_g}{\mu^2} + \frac{C_f^2\nu_g}{2\mu^2} + \frac{C_f^2\rho_g^2}{2\mu^3}\right) = \mathcal{O}(\ell\kappa^6).$$

Proof Taking total derivative on $\nabla\varphi(x) = \nabla_x f(x, y^*(x)) + \nabla y^*(x)\nabla_y f(x, y^*(x))$ yields

$$\begin{aligned}\nabla^2\varphi(x) = {}& \nabla_{xx}^2 f(x, y^*(x)) + \nabla y^*(x)\nabla_{yx}^2 f(x, y^*(x)) + \nabla^2 y^*(x)\bar{\times}_3\nabla_y f(x, y^*(x)) \\ & + \nabla_{xy}^2 f(x, y^*(x))[\nabla y^*(x)]^\top + \nabla y^*(x)\nabla_{yy}^2 f(x, y^*(x))[\nabla y^*(x)]^\top.\end{aligned} \tag{27}$$

Plug into the close form of $\nabla^2 y^*(x)$ given by Equation 21, we arrive at

$$\begin{aligned}&\nabla^2\varphi(x) \\ &= \underbrace{\nabla_{xx}^2 f(x, y^*(x)) - \nabla_{xxy}^3 g(x, y^*(x)) \times_3 [\nabla_{yy}^2 g(x, y^*(x))]^{-1}\bar{\times}_3\nabla_y f(x, y^*(x))}_{\text{(I)}} \\ &\quad + \underbrace{\left(\nabla_{yx}^2 f(x, y^*(x)) - \nabla_{yxy}^3 g(x, y^*(x)) \times_3 [\nabla_{yy}^2 g(x, y^*(x)]^{-1}\bar{\times}_3\nabla_y f(x, y^*(x))\right) \times_1 \nabla y^*(x)}_{\text{(II)}} \\ &\quad + \underbrace{\left(\nabla_{xy}^2 f(x, y^*(x)) - \nabla_{xyy}^3 g(x, y^*(x)) \times_3 [\nabla_{yy}^2 g(x, y^*(x)]^{-1}\bar{\times}_3\nabla_y f(x, y^*(x))\right) \times_2 \nabla y^*(x)}_{\text{(III)}} \\ &\quad + \underbrace{\left(\nabla_{yy}^2 f(x, y^*(x)) - \nabla_{yyy}^3 g(x, y^*(x)) \times_3 [\nabla_{yy}^2 g(x, y^*(x)]^{-1}\bar{\times}_3\nabla_y f(x, y^*(x))\right) \times_1 \nabla y^*(x) \times_2 \nabla y^*(x)}_{\text{(IV)}}.\end{aligned}$$

Recall that

$$\begin{aligned}\nabla^2\mathcal{L}_\lambda^*(x) = {}& \nabla_{xx}^2 f(x, y_\lambda^*(x)) + \lambda(\nabla_{xx}^2 g(x, y_\lambda^*(x)) - \nabla_{xx}^2 g(x, y^*(x))) \\ & + \nabla y_\lambda^*(x)\nabla_{yx}^2\mathcal{L}_\lambda(x, y_\lambda^*(x)) - \lambda\nabla y^*(x)\nabla_{yx}^2 g(x, y^*(x)).\end{aligned}$$

Our goal is show that $\nabla\mathcal{L}_\lambda^*(x) \approx \nabla^2\varphi(x)$. At first glance, these two quantities are very different and we can not directly bound their difference: $\nabla^2\varphi(x)$ takes the form of $A + BC + C^\top B^\top + BDB^\top$, while $\nabla^2\mathcal{L}_\lambda^*(x)$ looks different. Below, we introduce an intermediary quantity $\tilde{\nabla}^2\mathcal{L}_\lambda^*(x)$ which takes a similar form as $\nabla^2\varphi(x)$ to serves as a bridge:

$$\begin{aligned}\tilde{\nabla}^2\mathcal{L}_\lambda^*(x) = {}& \underbrace{\nabla_{xx}^2 f(x, y_\lambda^*(x)) + \lambda(\nabla_{xx}^2 g(x, y_\lambda^*(x)) - \nabla_{xx}^2 g(x, y^*(x)))}_{\text{(I}')} \\ & + \underbrace{\nabla y^*(x)\left(\nabla_{yx}^2\mathcal{L}_\lambda(x, y_\lambda^*(x)) - \lambda\nabla_{yx}^2 g(x, y^*(x))\right)}_{\text{(II}')} \\ & + \underbrace{\left(\nabla_{xy}^2\mathcal{L}_\lambda(x, y_\lambda^*(x)) - \lambda\nabla_{xy}^2 g(x, y^*(x))\right)[\nabla y^*(x)]^\top}_{\text{(III}')} \\ & + \underbrace{\nabla y^*(x)\left(\nabla_{yy}^2\mathcal{L}_\lambda(x, y_\lambda^*(x)) - \lambda\nabla_{yy}^2 g(x, y^*(x))\right)[\nabla y^*(x)]^\top}_{\text{(IV}')}.\end{aligned} \tag{28}$$

Now we show $\nabla^2\mathcal{L}_\lambda^*(x) \approx \tilde{\nabla}^2\mathcal{L}_\lambda^*(x) \approx \nabla^2\varphi(x)$.

$$
\begin{aligned}
&\tilde{\nabla}^2\mathcal{L}_\lambda^*(x) - \nabla^2\mathcal{L}_\lambda^*(x) \\
&= (\nabla y^*(x) - \nabla y_\lambda^*(x))\nabla_{yx}^2\mathcal{L}_\lambda(x, y_\lambda^*(x)) + \\
&\quad + \nabla_{xy}^2\mathcal{L}_\lambda(x, y_\lambda^*(x))[\nabla y^*(x)]^\top + \nabla y^*(x)\nabla_{yy}^2\mathcal{L}_\lambda(x, y_\lambda^*(x))[\nabla y^*(x)]^\top \\
&= (\nabla y^*(x) - \nabla y_\lambda^*(x))\nabla_{yx}^2\mathcal{L}_\lambda(x, y_\lambda^*(x)) + \nabla_{xy}^2\mathcal{L}_\lambda(x, y_\lambda^*(x))[\nabla y^*(x) - \nabla y_\lambda^*(x)]^\top \\
&\quad + \nabla y^*(x)\nabla_{yy}^2\mathcal{L}_\lambda(x, y_\lambda^*(x))[\nabla y^*(x)]^\top - \nabla y_\lambda^*(x)\nabla_{yy}^2\mathcal{L}_\lambda(x, y_\lambda^*(x))[\nabla y_\lambda^*(x)]^\top \\
&= (\nabla y^*(x) - \nabla y_\lambda^*(x))\nabla_{yx}^2\mathcal{L}_\lambda(x, y_\lambda^*(x)) + \nabla_{xy}^2\mathcal{L}_\lambda(x, y_\lambda^*(x))[\nabla y^*(x) - \nabla y_\lambda^*(x)]^\top \\
&\quad + (\nabla y^*(x) - \nabla y_\lambda^*(x))\nabla_{yy}^2\mathcal{L}_\lambda(x, y_\lambda^*(x))[\nabla y_\lambda^*(x)]^\top \\
&\quad + \nabla y_\lambda^*(x)\nabla_{yy}^2\mathcal{L}_\lambda(x, y_\lambda^*(x))(\nabla y^*(x) - \nabla y_\lambda^*(x))^\top \\
&\quad + (\nabla y^*(x) - \nabla y_\lambda^*(x))\nabla_{yy}^2\mathcal{L}_\lambda(x, y_\lambda^*(x))[\nabla y^*(x) - \nabla y_\lambda^*(x)]^\top \\
&= (\nabla y^*(x) - \nabla y_\lambda^*(x))\nabla_{yy}^2\mathcal{L}_\lambda(x, y_\lambda^*(x))[\nabla y^*(x) - \nabla y_\lambda^*(x)]^\top,
\end{aligned}
$$

where we use the identity

$$UAU^\top - VAV^\top = (U-V)AV^\top + VA(U-V)^\top + (U-V)A(U-V)^\top$$

in the second last step, and we cancel the first four terms by Equation 17 in the final step. Noting that $\nabla_{yy}^2\mathcal{L}_\lambda(\,\cdot\,,\,\cdot\,) \preceq 2\lambda L_g$, we have

$$\|\tilde{\nabla}^2\mathcal{L}_\lambda^*(x) - \nabla^2\mathcal{L}_\lambda^*(x)\| \le 2\lambda L_g\|\nabla y^*(x) - \nabla y_\lambda^*(x)\|^2 \le \frac{2L_gD_2^2}{\lambda}.$$

Now, we have successfully simplified our goal to showing $\tilde{\nabla}^2\mathcal{L}_\lambda^*(x) \approx \nabla^2\varphi(x)$, which can be done by showing (I) $\approx$ (I′), (II) $\approx$ (II′), (III) $\approx$ (III′), and (IV) $\approx$ (IV′), separately. Firstly,

$$
\begin{aligned}
&\text{(I)} - \text{(I}') \\
&= \nabla_{xx}^2 f(x, y^*(x)) - \nabla_{xx}^2 f(x, y_\lambda^*(x)) \\
&\quad + \nabla_{xxy}^3 g(x, y^*(x)) \times_3 [\nabla_{yy}^2 g(x, y^*(x))]^{-1} \bar{\times}_3 (\nabla_y f(x, y_\lambda^*(x)) - \nabla_y f(x, y^*(x))) \\
&\quad + \lambda \left(\nabla_{xx}^2 g(x, y^*(x)) - \nabla_{xx}^2 g(x, y_\lambda^*(x)) + \nabla_{xxy}^3 g(x, y^*(x)) \bar{\times}_3 (y_\lambda^*(x) - y^*(x))\right) \\
&\quad + \lambda \nabla_{xxy}^3 g(x, y^*(x)) \times_3 [\nabla_{yy}^2 g(x, y^*(x))]^{-1} \\
&\qquad\qquad \bar{\times}_3 \left(\nabla_y g(x, y_\lambda^*(x)) - \nabla_y g(x, y^*(x)) - \nabla_{yy}^2 g(x, y^*(x))(y_\lambda^*(x) - y^*(x))\right).
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
\|\text{(I)} - \text{(I}')\| &\le \rho_f\|y^*(x) - y_\lambda^*(x)\| + \frac{L_f\rho_g}{\mu}\|y^*(x) - y_\lambda^*(x)\| \\
&\quad + \frac{\lambda\nu_g}{2}\|y^*(x) - y_\lambda^*(x)\|^2 + \frac{\lambda\rho_g^2}{2\mu}\|y^*(x) - y_\lambda^*(x)\|^2 \\
&\le \frac{C_f\rho_f}{\lambda\mu} + \frac{C_fL_f\rho_g}{\lambda\mu^2} + \frac{C_f^2\nu_g}{2\lambda\mu^2} + \frac{C_f^2\rho_g^2}{2\lambda\mu^3}.
\end{aligned}
$$

Using $\|\nabla y^*(x)\| \le L_g/\mu$, we can similarly bound the difference between (II) and (II$'$) by

$$\begin{aligned}
&\|(\mathrm{II}) - (\mathrm{II}')\| \\
&\le \|\nabla y^*(x)\| \Big\| \nabla^2_{yx} f(x, y^*(x)) - \nabla^3_{yxy} g(x, y^*(x)) \times_3 [\nabla^2_{yy} g(x, y^*(x)]^{-1} \bar{\times}_3 \nabla_y f(x, y^*(x)) \\
&\quad - \nabla^2_{yx} f(x, y^*_\lambda(x)) - \lambda \left(\nabla^2_{yx} g(x, y^*_\lambda(x)) - \nabla^2_{yx} g(x, y^*(x)) \right) \Big\| \\
&\le \frac{L_g}{\mu} \left(\frac{C_f \rho_f}{\lambda \mu} + \frac{C_f L_f \rho_g}{\lambda \mu^2} + \frac{C_f^2 \nu_g}{2\lambda \mu^2} + \frac{C_f^2 \rho_g^2}{2\lambda \mu^3} \right),
\end{aligned}$$

And the bound for (III) and (III$'$) is the same. Finally, we know that

$$\begin{aligned}
&\big\|(\mathrm{IV}) - (\mathrm{IV}')\big\| \\
&\le \|\nabla y^*(x)\|^2 \Big\| \nabla^2_{yy} f(x, y^*(x)) - \nabla^3_{yyy} g(x, y^*(x)) \times_3 [\nabla^2_{yy} g(x, y^*(x))]^{-1} \bar{\times}_3 \nabla_y f(x, y^*(x)) \\
&\quad - \nabla^2_{yy} f(x, y^*_\lambda(x)) - \lambda \left(\nabla^2_{yy} g(x, y^*_\lambda(x)) - \nabla^2_{yy} g(x, y^*(x)) \right) \Big\| \\
&\le \frac{L_g^2}{\mu^2} \left(\frac{C_f \rho_f}{\lambda \mu} + \frac{C_f L_f \rho_g}{\lambda \mu^2} + \frac{C_f^2 \nu_g}{2\lambda \mu^2} + \frac{C_f^2 \rho_g^2}{2\lambda \mu^3} \right).
\end{aligned}$$

Combing the above bounds completes our proof. ■

Appendix D. Proofs of Finding First-Order Stationarity

We recall the standard result for gradient descent for strongly convex functions, which is used for proving the linear convergence of $y_t^k \to y_\lambda^*(x_t)$ and $z_t^k \to y^*(x_t)$.

Theorem D.1 (Bubeck et al. (2015, Theorem 3.10)) *Suppose $h(x) : \mathbb{R}^d \to \mathbb{R}$ is β-gradient Lipschitz and α-strongly convex. Consider the following update of gradient descent:*

$$x_{t+1} = x_t - \frac{1}{\beta} \nabla h(x_t).$$

Let $x^* = \arg\min_{x \in \mathbb{R}^d} h(x)$. Then it holds that

$$\|x_{t+1} - x^*\|^2 \le \left(1 - \frac{\alpha}{\beta}\right) \|x_t - x^*\|^2.$$

Below, we prove that F^{2}BA achieves the near-optimal first-order oracle complexity.

Theorem 4.1 *Suppose Assumption 3.1 holds. Define $\Delta := \varphi(x_0) - \inf_{x \in \mathbb{R}^{d_x}} \varphi(x)$ and $R := \|y_0 - y^*(x_0)\|^2$. Let $\eta_x \asymp \ell^{-1}\kappa^{-3}$, $\lambda \asymp \max\{\kappa/R,\ \ell\kappa^2/\Delta,\ \ell\kappa^3/\epsilon\}$ and set other parameters in Algorithm 1 as*

$$\eta_z = \eta_y = \frac{1}{2\lambda L_g},\ K = \mathcal{O}\left(\frac{L_g}{\mu} \log\left(\frac{\lambda L_g}{\mu} \right) \right),$$

then it can find an ϵ-first-order stationary point of $\varphi(x)$ within $\mathcal{O}(\ell\kappa^4\epsilon^{-2}\log(\ell\kappa/\epsilon))$ first-order oracle calls, where ℓ, κ are defined in Definition 3.1.

Proof Let L be the gradient Lipschitz coefficient of $\mathcal{L}_\lambda^*(x)$. According to Lemma 4.1 and the setting of λ, we have

a. $\sup_{x\in\mathbb{R}^{d_x}} \|\nabla \mathcal{L}_\lambda^*(x) - \nabla\varphi(x)\| = \mathcal{O}(\epsilon)$.

b. $\mathcal{L}_\lambda^*(x_0) - \inf_{x\in\mathbb{R}^{d_x}} \mathcal{L}_\lambda^*(x) = \mathcal{O}(\Delta)$.

c. $L := \sup_{x\in\mathbb{R}^{d_x}} \|\nabla^2 \mathcal{L}_\lambda^*(x)\| = \mathcal{O}(\ell\kappa^3)$.

Due to Lemma B.2, we also have $\|y_0 - y_\lambda^*(x_0)\|^2 + \|y_0 - y^*(x_0)\|^2 = \mathcal{O}(R)$. Now it suffices to show that the algorithm can find an ϵ-first-order stationary of $\mathcal{L}_\lambda^*(x)$. We begin from the following descent lemma for gradient descent. Let $\eta_x \le 1/(2L)$, then

$$\begin{aligned}
\mathcal{L}_\lambda^*(x_{t+1}) &\le \mathcal{L}_\lambda^*(x_t) + \langle \nabla \mathcal{L}_\lambda^*(x_t), x_{t+1} - x_t\rangle + \frac{L}{2}\|x_{t+1} - x_t\|^2 \\
&= \mathcal{L}_\lambda^*(x_t) - \frac{\eta_x}{2}\|\nabla \mathcal{L}_\lambda^*(x_t)\|^2 - \left(\frac{\eta_x}{2} - \frac{\eta_x^2 L}{2}\right)\|\hat{\nabla} \mathcal{L}_\lambda^*(x_t)\|^2 + \frac{\eta_x}{2}\|\hat{\nabla} \mathcal{L}_\lambda^*(x_t) - \nabla \mathcal{L}_\lambda^*(x_t)\|^2 \\
&\le \mathcal{L}_\lambda^*(x_t) - \frac{\eta_x}{2}\|\nabla \mathcal{L}_\lambda^*(x_t)\|^2 - \frac{1}{4\eta_x}\|x_{t+1} - x_t\|^2 + \frac{\eta_x}{2}\|\hat{\nabla} \mathcal{L}_\lambda^*(x_t) - \nabla \mathcal{L}_\lambda^*(x_t)\|^2.
\end{aligned} \tag{29}$$

Note that

$$\|\hat{\nabla} \mathcal{L}_\lambda^*(x_t) - \nabla \mathcal{L}_\lambda^*(x_t)\| \le 2\lambda L_g \|y_t^K - y_\lambda^*(x_t)\| + \lambda L_g \|z_t^K - y^*(x_t)\|.$$

By Theorem D.1, we have

$$\begin{aligned}
\|y_t^K - y_\lambda^*(x_t)\|^2 &\le \exp\left(-\frac{\mu K}{4L_g}\right)\|y_t^0 - y_\lambda^*(x_t)\|^2 \\
\|z_t^K - y^*(x_t)\|^2 &\le \exp\left(-\frac{\mu K}{L_g}\right)\|z_t^0 - y^*(x_t)\|^2
\end{aligned}$$

Therefore, we have

$$\|\hat{\nabla} \mathcal{L}_\lambda^*(x_t) - \nabla \mathcal{L}_\lambda^*(x_t)\|^2 \le 4\lambda^2 L_g^2 \exp\left(-\frac{\mu K}{4L_g}\right)\left(\|y_t^0 - y_\lambda^*(x_t)\|^2 + \|z_t^0 - y^*(x_t)\|^2\right) \tag{30}$$

By Young's inequality,

$$\begin{aligned}
\|y_{t+1}^0 - y_\lambda^*(x_{t+1})\|^2 &\le 2\|y_t^K - y_\lambda^*(x_t)\|^2 + 2\|y_\lambda^*(x_{t+1}) - y_\lambda^*(x_t)\|^2 \\
&\le 2\exp\left(-\frac{\mu K}{4L_g}\right)\|y_t^0 - y_\lambda^*(x_t)\|^2 + \frac{32L_g^2}{\mu^2}\|x_{t+1} - x_t\|^2,
\end{aligned}$$

where the second inequality follows Lemma B.6 that $y_\lambda^*(x)$ is $(4L_g/\mu)$-Lipschitz. Similarly, we can derive the recursion about $\|z_t^0 - y^*(x_t)\|^2$.

Put them together and let $K \ge 8L_g/\mu$, we have

$$\|y_{t+1}^0 - y_\lambda^*(x_{t+1})\|^2 + \|z_{t+1}^0 - y^*(x_{t+1})\|^2 \tag{31}$$

$$\leq 2\exp\left(-\frac{\mu K}{4L_g}\right)\left(\|y_t^0 - y_\lambda^*(x_t)\|^2 + \|z_t^0 - y^*(x_t)\|^2\right) + \frac{34L_g^2}{\mu^2}\|x_{t+1} - x_t\|^2 \tag{32}$$

$$\leq \frac{1}{2}\left(\|y_t^0 - y_\lambda^*(x_t)\|^2 + \|z_t^0 - y^*(x_t)\|^2\right) + \frac{34L_g^2}{\mu^2}\|x_{t+1} - x_t\|^2. \tag{33}$$

Telescoping over t yields

$$\begin{aligned}&\|y_t^0 - y_\lambda^*(x_t)\|^2 + \|z_t^0 - y^*(x_t)\|^2\\ &\leq \underbrace{\left(\frac{1}{2}\right)^t\left(\|y_0 - y_\lambda^*(x_0)\|^2 + \|y_0 - y^*(x_0)\|^2\right) + \frac{34L_g^2}{\mu^2}\sum_{j=0}^{t-1}\left(\frac{1}{2}\right)^{t-1-j}\|x_{j+1} - x_j\|^2}_{:=(*)}.\end{aligned}$$

Plug into Equation 30, which, in conjunction with Equation 29, yields that

$$\mathcal{L}_\lambda^*(x_{t+1}) \leq \mathcal{L}_\lambda^*(x_t) - \frac{\eta_x}{2}\|\nabla\mathcal{L}_\lambda^*(x_t)\|^2 - \frac{1}{4\eta_x}\|x_{t+1} - x_t\|^2 + 2\eta_x \times \underbrace{\lambda^2 L_g^2\exp\left(-\frac{\mu K}{4L_g}\right)}_{:=\gamma} \times (*).$$

Telescoping over t further yields

$$\begin{aligned}\frac{\eta_x}{2}\sum_{t=0}^{T-1}\|\nabla\mathcal{L}_\lambda^*(x_t)\|^2 \leq \mathcal{L}_\lambda^*(x_0) - \inf_{x\in\mathbb{R}^{d_x}}\mathcal{L}_\lambda^*(x) + 4\eta_x\gamma\left(\|y_0 - y_\lambda^*(x_0)\|^2 + \|y_0 - y^*(x_0)\|^2\right)\\ - \left(\frac{1}{4\eta_x} - \frac{136\eta_x\gamma L_g^2}{\mu^2}\right)\sum_{t=0}^{T-1}\|x_{t+1} - x_t\|^2\end{aligned} \tag{34}$$

Let $K = \mathcal{O}(\kappa\log(\lambda\kappa)) = \mathcal{O}(\kappa\log(\ell\kappa/\epsilon))$ such that γ is sufficiently small with

$$\gamma \leq \min\left\{\frac{\mu^2}{1088\eta_x^2 L_g^2}, \frac{1}{4\eta_x}\right\}.$$

Then we have,

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla\mathcal{L}_\lambda^*(x_t)\|^2 \leq \frac{2}{\eta_x T}\left(\mathcal{L}_\lambda^*(x_0) - \inf_{x\in\mathbb{R}^{d_x}}\mathcal{L}_\lambda^*(x) + \|y_0 - y_\lambda^*(x_0)\|^2 + \|y_0 - y^*(x_0)\|^2\right).$$

■

Below, we focus on the stochastic setting. We need to use the following theorem for large batch SGD to bound the error from inner loop.

Theorem D.2 *Suppose $h(x) : \mathbb{R}^d \to \mathbb{R}$ is β-gradient Lipschitz and α-strongly convex. Consider the following update of stochastic gradient descent (SGD):*

$$x_{t+1} = x_t - \frac{1}{\beta}\nabla h(x_t; \mathcal{B}_t),$$

where the mini-batch gradient satisfies

$$\mathbb{E}_{\mathcal{B}_t}\left[\nabla h(x_t;\mathcal{B}_t)\right] = \nabla h(x_t), \quad \mathbb{E}_{\mathcal{B}_t}\|\nabla h(x_t;\mathcal{B}_t) - \nabla h(x_t)\|^2 \le \frac{\sigma^2}{B}.$$

Then it holds that

$$\mathbb{E}\left[\|x_T - x^*\|^2\right] \le \left(1-\frac{\alpha}{\beta}\right)^T \frac{\beta}{\alpha}\|x_0 - x^*\|^2 + \frac{\sigma^2}{\alpha^2 B},$$

where $x^* = \arg\min_{x\in\mathbb{R}^d} h(x)$.

Proof We have that

$$\begin{aligned}
\mathbb{E}\left[h(x_{t+1}) - h^*\right] &\le \mathbb{E}\left[h(x_t) - h^* + \nabla h(x_t)^\top (x_{t+1} - x_t) + \frac{\beta}{2}\|x_{t+1} - x_t\|^2\right] \\
&= \mathbb{E}\left[h(x_t) - h^* - \frac{1}{2\beta}\|\nabla h(x_t)\|^2 + \frac{1}{2\beta}\|\nabla h(x_t) - \nabla h(x_t;\mathcal{B}_t)\|^2\right] \\
&\le \left(1-\frac{\alpha}{\beta}\right)\left(h(x_t) - h^*\right) + \frac{\sigma^2}{2\beta B}.
\end{aligned}$$

Telescoping gives

$$\mathbb{E}\left[h(x_T) - h^*\right] \le \left(1-\frac{\alpha}{\beta}\right)^T \left(h(x_0) - h^*\right) + \frac{\sigma^2}{2\alpha B}.$$

We conclude the proof by using the fact that

$$\frac{\alpha}{2}\|x - x^*\|^2 \le h(x) - h^* \le \frac{\beta}{2}\|x - x^*\|^2.$$

■

Theorem 4.2 *Suppose Assumption 3.1 and 4.1 hold. Let* ℓ, κ *defined as Definition 3.1. Define* $\Delta := \varphi(x_0) - \inf_{x\in\mathbb{R}^{d_x}} \varphi(x)$ *and* $R := \|y_0 - y^*(x_0)\|^2$. *Let* $\eta_x \asymp \ell^{-1}\kappa^{-3}$, $\lambda \asymp \max\left\{\kappa/R,\ \ell\kappa^2/\Delta,\ \ell\kappa^3/\epsilon\right\}$ *and set other parameters in Algorithm 2 as*

$$\begin{aligned}
&\eta_z = \eta_y = \frac{1}{2\lambda L_g},\ K_t = \tilde{\mathcal{O}}\left(\frac{L_g \log\delta_t}{\mu}\right), \\
&B_{\text{out}} \asymp \frac{\sigma_f^2 + \lambda^2\sigma_g^2}{\epsilon^2},\quad B_{\text{in}} \asymp \frac{L_f^2 + \lambda^2 L_g^2}{\lambda^2\mu^2}\cdot B_{\text{out}}.
\end{aligned}$$

where δ_t *is defined via the recursion*

$$\delta_{t+1} = \frac{1}{2}\delta_t + \frac{34L_g^2}{\mu^2}\|x_{t+1} - x_t\|^2 + \frac{\sigma_g^2}{2L_g B_{\text{in}}}, \quad \delta_0 = \mathcal{O}(R),$$

then it can output a point such that $\mathbb{E}\|\nabla\varphi(x)\| \le \epsilon$ *within* $T = \mathcal{O}(\ell\kappa^3\epsilon^{-2})$ *iterations. The total number of stochastic first-order oracle calls is bounded by*

$$\begin{cases}
\mathcal{O}(\ell\kappa^6\epsilon^{-4}\log(\ell\kappa/\epsilon)), & \sigma_f > 0, \sigma_g = 0; \\
\mathcal{O}(\ell^3\kappa^{12}\epsilon^{-6}\log(\ell\kappa/\epsilon)), & \sigma_f > 0, \sigma_g > 0.
\end{cases}$$

Proof By taking expectation in Equation 29, we get

$$\begin{aligned}&\mathbb{E}[\mathcal{L}_\lambda^*(x_{t+1})]\\ &\leq \mathbb{E}\left[\mathcal{L}_\lambda^*(x_t) - \frac{\eta_x}{2}\|\nabla\mathcal{L}_\lambda^*(x_t)\|^2 - \frac{1}{4\eta_x}\|x_{t+1}-x_t\|^2 + \frac{\eta_x}{2}\|G_t - \nabla\mathcal{L}_\lambda^*(x_t)\|^2\right].\end{aligned} \tag{35}$$

Then to prove that this algorithm can output a point such that $\mathbb{E}\|\nabla\varphi(x)\| \leq \epsilon$, it suffices to show $\mathbb{E}\|G_t - \nabla\mathcal{L}_\lambda^*(x_t)\|^2 = \mathcal{O}(\epsilon^2)$ holds for all t. This requires

a. $\mathbb{E}\|\nabla_x f(x_t, y_t^K; B_{\text{out}}) - \nabla_x f(x_t, y_\lambda^*(x_t))\|^2 = \mathcal{O}(\epsilon^2)$.

b. $\mathbb{E}\|\nabla_x g(x_t, y_t^K; B_{\text{out}}) - \nabla_x g(x_t, y_\lambda^*(x_t))\|^2 = \mathcal{O}\left(\epsilon^2/\lambda^2\right)$.

c. $\mathbb{E}\|\nabla_x g(x_t, z_t^K; B_{\text{out}}) - \nabla_x g(x_t, y^*(x_t))\|^2 = \mathcal{O}\left(\epsilon^2/\lambda^2\right)$.

Our setting of B_{out} shows that it suffices to have

$$\max\left\{\mathbb{E}\|y_t^K - y_\lambda^*(x_t)\|^2, \mathbb{E}\|z_t^K - y^*(x_t)\|^2\right\} = \mathcal{O}\left(\frac{\epsilon^2}{L_f^2 + \lambda^2 L_g^2}\right),$$

And our setting of B_{in} and K_t fulfills these conditions if $\|y_t^0 - y_\lambda^*(x_t)\|^2 + \|z_t^0 - y^*(x_t)\|^2 \leq \delta_t$. Now we use Theorem D.2 to establish the recursion of δ_t. Note that

$$\begin{aligned}&\mathbb{E}\|y_{t+1}^0 - y_\lambda^*(x_{t+1})\|^2\\ &\leq 2\mathbb{E}\|y_t^K - y_\lambda^*(x_t)\|^2 + 2\mathbb{E}\|y_\lambda^*(x_t) - y_\lambda^*(x_{t+1})\|^2\\ &\leq \frac{4L_g}{\mu}\exp\left(-\frac{\mu K_t}{2L_g}\right)\mathbb{E}\|y_t^0 - y_\lambda^*(x_t)\|^2 + \frac{32L_g^2}{\mu^2}\|x_{t+1}-x_t\|^2 + \frac{\sigma_g^2}{4L_g B_{\text{in}}}\\ &\leq \frac{1}{2}\mathbb{E}\|y_t^0 - y_\lambda^*(x_t)\|^2 + \frac{32L_g^2}{\mu^2}\|x_{t+1}-x_t\|^2 + \frac{\sigma_g^2}{4L_g B_{\text{in}}}\end{aligned} \tag{36}$$

where the second last inequality uses Theorem D.2 and the last inequality holds once we get $K_t = \Omega(\kappa\log(\kappa))$. A similar bound also holds for $\mathbb{E}\|z_{t+1}^0 - y^*(x_{t+1})\|^2$. Putting them together, we get that

$$\delta_{t+1} \leq \frac{1}{2}\delta_t + \frac{34L_g^2}{\mu^2}\|x_{t+1}-x_t\|^2 + \frac{\sigma_g^2}{2L_g B_{\text{in}}}$$

Then we telescope the recursion of δ_t and take expectation to obtain that

$$\begin{aligned}\sum_{t=0}^{T-1}\mathbb{E}[\delta_t] &\leq 2\delta_0 + \frac{68L_g^2}{\mu^2}\sum_{t=0}^{T-1}\mathbb{E}[\|x_{t+1}-x_t\|^2] + \frac{\sigma_g^2}{L_g B_{\text{in}}}\\ &= 2\delta_0 + \mathcal{O}(\eta_x\Delta + T\eta_x^2\epsilon^2) + \frac{\sigma_g^2}{L_g B_{\text{in}}},\end{aligned} \tag{37}$$

where we use Equation 35 in the last step. Let C be a constant that contains logarithmic factors. We can the bound the expected total iteration number via

$$\sum_{t=0}^{T-1}\mathbb{E}[K_t] = \frac{CL_g}{\mu}\sum_{t=0}^{T-1}\mathbb{E}[\log(\delta_t)] \leq \frac{CL_g T}{\mu}\log\left(\frac{\sum_{t=0}^{T-1}\mathbb{E}[\delta_t]}{T}\right),$$

Algorithm 5 F^{3}BSA(x, y_0)

$z_0 = y_0$
for $t = 0, 1, \cdots, T-1$
  Sample random index (ϕ_f^t, ϕ_g^t)
  $(G_t, y_{t+1}, z_{t+1}) =$ MLMC-HyperGradEst$(x_t, y_t, z_t, \phi_f^t, \phi_g^t, M_t, N_t, S_t)$
  $x_{t+1} = x_t - \eta_x G_t$
end for

Algorithm 6 MLMC-HyperGradEst$(x, y_0, z_0, \phi_f, \phi_g, M, N, S)$

for $m = 1, \cdots, M$
  Draw $J \sim$ Geom$(1/2)$
  Write $(G_m^{(j)}, y_m^{(j)}, z_m^{(j)}) =$ HyperGradEst(x, y_0, z_0, N, j)
  $G_m = G_m^{(0)} + 2^J (G_m^{(J)} - G_m^{(J-1)}) \mathbb{I}[J \le S]$
end for
$\hat{G} = \frac{1}{M} \sum_{m=1}^M G_m$, $\hat{y} = y_1^{(0)}$, $\hat{z} = y_1^{(0)}$
return $(\hat{G}, \hat{y}, \hat{z})$

where we use Jensen's inequality and the concavity of $\log(\cdot)$. Note that $\sum_{t=0}^{T-1} \mathbb{E}[\delta_t]$ can be upper bounded via Equation 37. Finally, the total sample complexity of the algorithm is

$$B_{\text{in}} \sum_{t=0}^{T-1} K_t + T B_{\text{out}},$$

which is the claimed complexity by plugging the choice of $T, B_{\text{in}}, B_{\text{out}}$. ■

Appendix E. Removing the Large Batches in F^{2}BSA

Theorem 4.2 requires large batch size $B_{\text{out}} \asymp (\sigma_f^2 + \lambda^2 \sigma_g^2)/\epsilon^2$ to track the deterministic algorithm. Below, we get rid of the large batches via a more intricate algorithm design. We consider the *single batch scenario*, where each SGD update samples a single random index (ϕ_f, ϕ_g). Define the following stochastic gradient

$$\begin{aligned} &\nabla \mathcal{L}_\lambda^*(x; \phi_f, \phi_g) = \nabla \mathcal{L}_\lambda(x, y_\lambda^*(x); \phi_f, \phi_g), \\ &\text{where } \ \nabla \mathcal{L}_\lambda(x, y; \phi_f, \phi_g) = \nabla_x f(x, y; \phi_f) + \lambda(\nabla_x g(x, y; \phi_g) - \nabla g^*(x; \phi_g)). \end{aligned} \tag{38}$$

It is clear that $\mathbb{E} \nabla \mathcal{L}_\lambda^*(x; \phi_f, \phi_g) = \nabla \mathcal{L}_\lambda^*(x)$. Our improved convergence rate in Theorem 4.2 requires the following condition on the hyper-gradient estimator G.

Assumption E.1 *Assume that the hyper-gradient estimator G satisfies*

$$\begin{aligned} \|\mathbb{E} G - \nabla \mathcal{L}_\lambda^*(x; \phi_f, \phi_g)\|^2 \le \zeta_{\text{bias}} = \mathcal{O}(\min\{\epsilon^2, \sigma_f^2 + \lambda^2 \sigma_g^2\}), \\ \mathbb{E} \|G - \mathbb{E} G\|^2 \le \zeta_{\text{varinace}} = \mathcal{O}(\sigma_f^2 + \lambda^2 \sigma_g^2), \end{aligned} \tag{39}$$

Algorithm 7 HyperGradEst$(x, y_0, z_0, \phi_f, \phi_g, N, K)$

1: $\hat{y} = \mathrm{SGD}^{\mathrm{SC}}(f(x, \cdot; \phi_f) + \lambda g(x, \cdot; \phi_g), y_0, N, K)$
2: $\hat{z} = \mathrm{SGD}^{\mathrm{SC}}(\lambda g(x, \cdot; \phi_g), z_0, N, K)$
3: $\hat{G} = \nabla_x f(x, \hat{y}; \phi_f) + \lambda(\nabla_x g(x, \hat{y}; \phi_g) - \nabla_x g(x, \hat{z}; \phi_g))$
4: **return** $(\hat{G}, \hat{y}, \hat{z})$

Theorem E.1 *Suppose Assumption 3.1 and 4.1 hold. Let ℓ, κ defined as Definition 3.1. Let $\lambda \asymp \max\{\ell\kappa^2/\Delta,\ \ell\kappa^3/\epsilon\}$, where $\Delta := \varphi(x_0) - \inf_{x\in\mathbb{R}^{d_x}} \varphi(x)$. If for any t the hyper-gradient estimator G_t satisfies Assumption E.1, then iterating $x_{t+1} = x_t - \eta_x G_t$ with*

$$\eta_x \asymp \frac{\epsilon^2}{(\sigma_f^2 + \lambda^2\sigma_g^2)\ell\kappa^3}, \quad T \asymp \frac{\Delta}{\eta_x \epsilon^2} \tag{40}$$

output $\bar{x}_T = \frac{1}{T}\sum_{t=0}^{T-1} x_t$ such that $\mathbb{E}\|\nabla\varphi(\bar{x}_T)\| \le \epsilon$.

In Assumption E.1, a nearly unbiased hyper-gradient estimator G_t is required to recover the same convergence rate as SGD on $\varphi(x)$. In F^{2}BSA (Algorithm 2), this condition of bias $\lesssim \epsilon^2$ is achieved by running inner-loop SGD with large batch to accuracy ϵ^2. In the following, we show it is possible to use existing technique (Asi et al., 2021; Hu et al., 2021) for improving biased SGD, which helps relax the accuracy of inner-loop SGD to $\sigma_f^2 + \lambda^2\sigma_g^2$. In this case, using single-batch SGD would not harm the convergence rate. We call this improved algorithm F^{3}BSA (see Algorithm 5). The ultimate algorithms look a little complicated at first glance, but they all come from standard techniques in the literature. We summarize the key steps below.

1. F^{3}BSA (Algorithm 5) conducts a SGD update on variable x with the MLMC hyper-gradient estimator G_t given by Algorithm 6.

2. The design of MCMC-HyperGradEst (Algorithm 6) has the same structure as the standard MLMC estimator (Giles, 2008; Blanchet and Glynn, 2015). Following (Asi et al., 2021; Hu et al., 2021), we wrap a high-bias hyper-gradient estimator (Algorithm 7) into the MLMC structure to return a low-bias estimator.

3. HyperGradEst (Algorithm 7) applies SGD updates to solve the lower-level problems with respect to variables y and z in Equation 6. Since the naive SGD update in F^{2}BSA can not meet the requirements in MLMC, we use the SGD$^{\mathrm{SC}}$ algorithm (Allen-Zhu, 2018a, Algorithm 2), which is presented in Section E.2 for completeness. It runs multiple epochs of SGD with different step sizes like (Hazan and Kale, 2014) to achieve the optimal rate in smooth strongly-convex stochastic first-order optimization.

The following theorem shows Algorithm 6 can output a nearly unbiased hyper-gradient estimator at a low cost. The proof follows from (Asi et al., 2021).

Theorem E.2 *Suppose Assumption 3.1 and 4.1 hold. Let ℓ, κ defined as Definition 3.1. For any given $x \in \mathbb{R}^{d_x}$, then Algorithm 6 with*

$$N_t = S_t \asymp \log\left(\frac{\max\{\lambda^2\ell^2\delta_t, \kappa(\sigma_f^2+\lambda^2\sigma_g^2)\}}{\zeta_{\text{bias}}}\right), \quad M_t \asymp \frac{\kappa(\sigma_f^2+\lambda^2\sigma_g^2)N_t}{\zeta_{\text{variance}}} \tag{41}$$

outputs an hyper-gradient estimator G_t satisfying Condition 39 in $\mathcal{O}(\kappa N_t M_t)$ SFO complexity, where δ_t is the upper bound of $\|y_t^0 - y_\lambda^*(x_t)\|^2 + \|z_t^0 - y^*(x_t)\|^2$.

Combining Theorem E.1 and E.2, we obtain the complexity of Algorithm 5 as follows.

Theorem E.3 *Suppose Assumption 3.1 and 4.1 hold. Let ℓ, κ defined as Definition 3.1. Define $\Delta := \varphi(x_0) - \inf_{x\in\mathbb{R}^{d_x}} \varphi(x)$ and $R := \|y_0 - y^*(x_0)\|^2$. Let $\lambda \asymp \max\left\{\kappa/R,\ \ell\kappa^2/\Delta,\ \ell\kappa^3/\epsilon\right\}$ and set other parameters in Algorithm 5 according to Equation 40 and Equation 41, where δ_t is defined via the recursion*

$$\delta_{t+1} = \frac{1}{2}\delta_t + \frac{34L_g^2}{\mu^2}\|x_{t+1}-x_t\|^2 + \frac{24(\sigma_f^2+\lambda^2\sigma_g^2)}{\lambda\mu(L_f+\lambda L_g)}, \quad \delta_0 = \mathcal{O}(R), \tag{42}$$

then it can output a point such that $\mathbb{E}\|\nabla\varphi(x)\| \le \epsilon$ in the total SFO complexity of

$$\begin{cases} \mathcal{O}(\ell\kappa^5\epsilon^{-4}\log(\ell\kappa/\epsilon)), & \sigma_f > 0, \sigma_g = 0; \\ \mathcal{O}(\ell^3\kappa^{11}\epsilon^{-6}\log(\ell\kappa/\epsilon)), & \sigma_f > 0, \sigma_g > 0. \end{cases}$$

The above theorem shows that F^{3}BSA achieves the $\tilde{\mathcal{O}}(\epsilon^{-4})$ and $\tilde{\mathcal{O}}(\epsilon^{-6})$ SFO complexity for the partially and fully stochastic cases as F^{2}BSA while using a single batch at each iteration. Another side advantage of F^{3}BSA is that it also improves the dependency in κ because its inner loop uses the optimal stochastic algorithm SGD$^{\text{SC}}$ (Allen-Zhu, 2018a, Algorithm 2) to achieve better bias-variance tradeoff, instead of using naive SGD in F^{2}BSA.

E.1 Missing Proofs for F^{3}BSA

We present the missing proofs in this section.

Theorem E.1 *Suppose Assumption 3.1 and 4.1 hold. Let ℓ, κ defined as Definition 3.1. Let $\lambda \asymp \max\{\ell\kappa^2/\Delta,\ \ell\kappa^3/\epsilon\}$, where $\Delta := \varphi(x_0) - \inf_{x\in\mathbb{R}^{d_x}}\varphi(x)$. If for any t the hyper-gradient estimator G_t satisfies Assumption E.1, then iterating $x_{t+1} = x_t - \eta_x G_t$ with*

$$\eta_x \asymp \frac{\epsilon^2}{(\sigma_f^2+\lambda^2\sigma_g^2)\ell\kappa^3}, \quad T \asymp \frac{\Delta}{\eta_x\epsilon^2} \tag{40}$$

output $\bar{x}_T = \frac{1}{T}\sum_{t=0}^{T-1} x_t$ such that $\mathbb{E}\|\nabla\varphi(\bar{x}_T)\| \le \epsilon$.

Proof Let L be the constant of gradient Lipschitz continuity of $\mathcal{L}_\lambda^*(x)$. From Lemma 4.1 we know that $L = \mathcal{O}(\ell\kappa^3)$. We have that

$$\mathbb{E}[\mathcal{L}_\lambda^*(x_{t+1})] \le \mathbb{E}\left[\mathcal{L}_\lambda^*(x_t) + \langle\nabla\mathcal{L}_\lambda^*(x_t), x_{t+1}-x_t\rangle + \frac{L}{2}\|x_{t+1}-x_t\|^2\right]$$

$$
\begin{aligned}
&= \mathbb{E}\left[\mathcal{L}_\lambda^*(x_t) - \eta_x \nabla \mathcal{L}_\lambda^*(x_t)^\top G_t + \frac{\eta_x^2 L}{2}\|G_t\|^2\right] \\
&\le \mathcal{L}_\lambda^*(x_t) - \left(\frac{\eta_x}{2} - \frac{\eta_x^2 L}{2}\right)\|\nabla \mathcal{L}_\lambda^*(x_t)\|^2 \\
&\quad + \frac{\eta_x}{2}\|\mathbb{E}G_t - \nabla \mathcal{L}_\lambda^*(x_t;\phi_f^t,\phi_g^t)\|^2 + \frac{\eta_x^2 L}{2}\mathbb{E}\|G_t - \nabla \mathcal{L}_\lambda^*(x_t)\|^2 \\
&\le \mathcal{L}_\lambda^*(x_t) - \left(\frac{\eta_x}{2} - \frac{\eta_x^2 L}{2}\right)\|\nabla \mathcal{L}_\lambda^*(x_t)\|^2 + \frac{\eta_x}{2}\|\mathbb{E}G_t - \nabla \mathcal{L}_\lambda^*(x_t;\phi_f^t,\phi_g^t)\|^2 \\
&\quad + \eta_x^2 L(\mathbb{E}\|G_t - \nabla \mathcal{L}_\lambda^*(x_t;\phi_f^t,\phi_g^t)\|^2 + \mathbb{E}\|\nabla \mathcal{L}_\lambda^*(x_t;\phi_f^t,\phi_g^t) - \nabla \mathcal{L}_\lambda^*(x_t;\phi_f^t,\phi_g^t)\|^2).
\end{aligned}
$$

Plugging in Equation 39 and Equation 40, we have that

$$
\mathbb{E}[\mathcal{L}_\lambda^*(x_{t+1})] \le \mathcal{L}_\lambda^*(x_t) - \frac{\eta_x}{4}\|\nabla \mathcal{L}_\lambda^*(x_t)\|^2 + \mathcal{O}(\eta_x \epsilon^2).
$$

Telescope over $t = 0, 1, \cdots, T-1$, we get

$$
\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla \mathcal{L}_\lambda^*(x_t)\|^2 \le \frac{4(\mathcal{L}_\lambda^*(x_t) - \inf_{x\in\mathbb{R}^{d_x}}\mathcal{L}_\lambda^*(x))}{\eta_x T} + \mathcal{O}(\epsilon^2), \tag{43}
$$

which outputs an ϵ-stationary point in $T = \mathcal{O}(\Delta/(\eta_x\epsilon^2))$ iterations. ■

Theorem E.2 *Suppose Assumption 3.1 and 4.1 hold. Let ℓ, κ defined as Definition 3.1. For any given $x \in \mathbb{R}^{d_x}$, then Algorithm 6 with*

$$
N_t = S_t \asymp \log\left(\frac{\max\{\lambda^2\ell^2\delta_t, \kappa(\sigma_f^2 + \lambda^2\sigma_g^2)\}}{\zeta_{\text{bias}}}\right), \quad M_t \asymp \frac{\kappa(\sigma_f^2 + \lambda^2\sigma_g^2)N_t}{\zeta_{\text{variance}}} \tag{41}
$$

outputs an hyper-gradient estimator G_t satisfying Condition 39 in $\mathcal{O}(\kappa N_t M_t)$ SFO complexity, where δ_t is the upper bound of $\|y_t^0 - y_\lambda^*(x_t)\|^2 + \|z_t^0 - y^*(x_t)\|^2$.

Proof By applying Theorem D.2, we know that Algorithm 7 ensures that

$$
\begin{aligned}
\mathbb{E}\|G_m^{(j)} - \nabla \mathcal{L}_\lambda^*(x_t)\|^2 &\le (L_f^2 + \lambda^2 L_g^2)\cdot\left(\|y_m^{(j)} - y_\lambda^*(x_t)\|^2 + \|z_m^{(j)} - y^*(x_t)\|^2\right) \\
&\le (L_f^2 + \lambda^2 L_g^2)\cdot\left(\frac{\delta_t}{2^{N+2K}} + \frac{\sigma_f^2 + \lambda^2\sigma_g^2}{2^{K-3}\lambda\mu(L_f + \lambda L_g)}\right),
\end{aligned} \tag{44}
$$

The fact $\mathbb{P}(J = j) = 2^{-j}$ implies that

$$
\mathbb{E}\hat{G} = \mathbb{E}G_1 = \mathbb{E}G_1^{(0)} + \sum_{j=1}^{S_t}\mathbb{P}(J=j)2^j(G_1^{(j)} - G_1^{(j-1)}) = \mathbb{E}\left[G_1^{S_t}\right].
$$

Therefore, using Equation 44 the bias of our estimator can be upper bounded by

$$
\|\mathbb{E}\hat{G} - \nabla \mathcal{L}_\lambda^*(x_t)\|^2 = \|\mathbb{E}G_1^{S_t} - \nabla \mathcal{L}_\lambda^*(x_t)\|^2 \le \mathbb{E}\|G_1^{S_t} - \nabla \mathcal{L}_\lambda^*(x_t)\|^2
$$

$$\leq (L_f^2 + \lambda^2 L_g^2) \cdot \left(\frac{\delta_t}{2^{N_t+2S_t}} + \frac{\sigma_f^2 + \lambda^2 \sigma_g^2}{2^{S_t-3}\lambda\mu(L_f + \lambda L_g)} \right).$$

It is easy to check that $\|\mathbb{E}\hat{G} - \nabla \mathcal{L}_\lambda^*(x)\|^2 \leq \zeta_{\text{bias}}$ holds by our hyper-parameter setting. Below, we analyze the variance of our estimator. Note that

$$\begin{aligned}
&\mathbb{E}\|G_m - G_m^{(0)}\|^2 \\
&= \sum_{j=1}^{S_t} \mathbb{P}(J=j) 2^{2j} \mathbb{E}\|G_m^{(j)} - G_m^{(j-1)}\|^2 \\
&= \sum_{j=1}^{S_t} 2^j \mathbb{E}\|G_m^{(j)} - G_m^{(j-1)}\|^2 \\
&\leq \sum_{j=1}^{S_t} 2^j \cdot \left(2\mathbb{E}\|G_m^{(j)} - \nabla \mathcal{L}_\lambda^*(x_t)\|^2 + 2\mathbb{E}\|G_m^{(j-1)} - \nabla \mathcal{L}_\lambda^*(x_t)\|^2 \right) \\
&\leq (L_f^2 + \lambda^2 L_g^2) \sum_{j=1}^{S_t} \frac{10\delta_t}{2^{N+j}} + \frac{48(\sigma_f^2 + \lambda^2\sigma_g^2)}{\lambda\mu(L_f + \lambda L_g)} \\
&\leq (L_f^2 + \lambda^2 L_g^2) \cdot \left(\frac{10\delta_t}{2^N} + \frac{48(\sigma_f^2 + \lambda^2\sigma_g^2)}{\lambda\mu(L_f + \lambda L_g)} \cdot S_t \right),
\end{aligned}$$

where the second last line uses Equation 44. Therefore, we have that

$$\begin{aligned}
&\mathbb{E}\|\hat{G} - \mathbb{E}\hat{G}\|^2 = \frac{1}{M_t}\mathbb{E}\|G_1 - \mathbb{E}G_1\|^2 \leq \frac{1}{M_t}\mathbb{E}\|G_1 - \nabla \mathcal{L}_\lambda^*(x_t)\|^2 \\
&\leq \frac{2}{M_t}\mathbb{E}\|G_1 - G_1^{(0)}\|^2 + \frac{2}{M_t}\mathbb{E}\|G_1^{(0)} - \nabla \mathcal{L}_\lambda^*(x_t)\|^2 \\
&\leq \frac{L_f^2 + \lambda^2 L_g^2}{M_t} \cdot \left(\frac{22\delta_t}{2^{N_t}} + \frac{100(\sigma_f^2 + \lambda^2\sigma_g^2)}{\lambda\mu(L_f + \lambda L_g)} \cdot S_t \right).
\end{aligned}$$

It is also easy to check that $\mathbb{E}\|\hat{G} - \mathbb{E}\hat{G}\|^2 \leq \zeta_{\text{variance}}$ holds by our hyper-parameter setting. Finally, the expected number of SFO calls is

$$\frac{8(L_f + \lambda L_g)}{\lambda\mu} \left(N_t + \sum_{j=1}^{S_t} \mathbb{P}(J=j) 2^j \right) M_t = \frac{8(L_f + \lambda L_g)}{\lambda\mu} (N_t + S_t) M_t = \mathcal{O}(\kappa N_t M_t).$$

■

Theorem E.3 *Suppose Assumption 3.1 and 4.1 hold. Let ℓ, κ defined as Definition 3.1. Define $\Delta := \varphi(x_0) - \inf_{x\in\mathbb{R}^{d_x}} \varphi(x)$ and $R := \|y_0 - y^*(x_0)\|^2$. Let $\lambda \asymp \max\left\{\kappa/R,\ \ell\kappa^2/\Delta,\ \ell\kappa^3/\epsilon\right\}$ and set other parameters in Algorithm 5 according to Equation 40 and Equation 41, where δ_t is defined via the recursion*

$$\delta_{t+1} = \frac{1}{2}\delta_t + \frac{34L_g^2}{\mu^2}\|x_{t+1} - x_t\|^2 + \frac{24(\sigma_f^2 + \lambda^2\sigma_g^2)}{\lambda\mu(L_f + \lambda L_g)}, \quad \delta_0 = \mathcal{O}(R), \tag{42}$$

then it can output a point such that $\mathbb{E}\|\nabla\varphi(x)\| \le \epsilon$ *in the total SFO complexity of*

$$\begin{cases} \mathcal{O}(\ell\kappa^5\epsilon^{-4}\log(\ell\kappa/\epsilon)), & \sigma_f > 0, \sigma_g = 0; \\ \mathcal{O}(\ell^3\kappa^{11}\epsilon^{-6}\log(\ell\kappa/\epsilon)), & \sigma_f > 0, \sigma_g > 0. \end{cases}$$

Proof By Theorem E.1 the outer-loop complexity is $T = \mathcal{O}\left((\sigma_f^2 + \lambda^2\sigma_g^2)\ell\kappa^3\Delta\epsilon^{-4}\right)$. By Theorem E.2 the inner-loop complexity in the t-th iteration is $\tilde{\mathcal{O}}\left(\kappa^2\log^2\delta_t\right)$ for δ_t satisfying $\|y_t^0 - y_\lambda^*(x_t)\|^2 + \|z_t^0 - y^*(x_t)\|^2 \le \delta_t$. Similar to Theorem 4.2, we also require to specify the recursion for δ_t with known problem parameters. Similar to Equation 36,we have that

$$\begin{aligned} \mathbb{E}\|y_{t+1} - y_\lambda^*(x_{t+1})\|^2 &\le 2\mathbb{E}\|y_{t+1} - y_\lambda^*(x_t)\|^2 + 2\mathbb{E}\|y_\lambda^*(x_t) - y_\lambda^*(x_{t+1})\|^2 \\ &\le \frac{1}{2}\|y_t - y_\lambda^*(x_t)\|^2 + \frac{16(\sigma_f^2 + \lambda^2\sigma_g^2)}{\lambda\mu(L_f + \lambda L_g)} + \frac{32L_g^2}{\mu^2}\|x_{t+1} - x_t\|^2, \end{aligned}$$

where the last step follows Theorem E.5 with $N_t \ge 8$ and Lemma B.6 that $y_\lambda^*(x)$ is $(4L_g/\mu)$-Lipschitz continuous. Similarly, we also have that

$$\begin{aligned} \mathbb{E}\|z_{t+1} - y^*(x_{t+1})\|^2 &\le 2\mathbb{E}\|z_{t+1} - y^*(x_t)\|^2 + 2\mathbb{E}\|y^*(x_t) - y^*(x_{t+1})\|^2 \\ &\le \frac{1}{2}\|z_t - y^*(x_t)\|^2 + \frac{8\sigma_g^2}{\mu\lambda L_g} + \frac{2L_g^2}{\mu^2}\|x_{t+1} - x_t\|^2, \end{aligned}$$

where the last step follows Theorem E.5 with $N_t \ge 8$ and that $y^*(x)$ is (L_g/μ)-Lipschitz continuous. Combining the above two inequalities yields Equation 42. Telescoping Equation 42 yields that

$$\begin{aligned} \sum_{t=0}^{T-1}\mathbb{E}[\delta_t] &\le 2\delta_0 + \frac{48(\sigma_f^2 + \lambda^2\sigma_g^2)}{\lambda\mu(L_f + \lambda L_g)} + \frac{68L_g^2}{\mu^2}\sum_{t=0}^{T-1}\mathbb{E}\|x_{t+1} - x_t\|^2 \\ &\le 2\delta_0 + \frac{48(\sigma_f^2 + \lambda^2\sigma_g^2)}{\lambda\mu(L_f + \lambda L_g)} + \frac{136L_g^2\eta_x^2}{\mu^2}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla\mathcal{L}_\lambda^*(x_t)\|^2 + \|G_t - \nabla\mathcal{L}_\lambda^*(x_t)\|^2] \\ &\le 2\delta_0 + \frac{48(\sigma_f^2 + \lambda^2\sigma_g^2)}{\lambda\mu(L_f + \lambda L_g)} + \frac{136L_g^2\eta_x^2 T}{\mu^2}\cdot\left(\epsilon^2 + \sigma_f^2 + \lambda^2\sigma_g^2\right). \end{aligned}$$

Therefore, the total SFO complexity (in expectation) is

$$\tilde{\mathcal{O}}\left(\kappa^2\sum_{t=0}^{T-1}\log^2\delta_t\right) \le \tilde{\mathcal{O}}\left(\kappa^2 T\max_{0\le t\le T-1}\log^2\delta_t\right) = \tilde{\mathcal{O}}(\kappa^2 T),$$

where T is given in Equation 40. ■

E.2 The SGD Subroutine

This section recalls the optimal stochastic algorithm for smooth strongly-convex optimization (Allen-Zhu, 2018a), which be used as the subroutines in F^{3}BSA (Algorithm 5).

Algorithm 8 SGD (h, x_0, η, K)

for $k = 0, 1, \cdots, K-1$
  Sample a random index ϕ_t
  $x_{k+1} = x_t - \eta \nabla h(x_t; \phi_t)$
end for
return $\bar{x}_K = \frac{1}{K}\sum_{k=1}^{K} x_k$

Assumption E.2 *Assume that the stochastic gradient satisfies*

$$\mathbb{E}_{\phi_t}\left[\nabla h(x_t;\phi_t)\right] = \nabla h(x_t), \quad \mathbb{E}_{\phi_t}\|\nabla h(x_t;\phi_t) - \nabla h(x_t)\|^2 \le \sigma^2.$$

Theorem E.4 *Allen-Zhu (2018a, Theorem 4.1 (a)) Suppose $h(x) : \mathbb{R}^d \to \mathbb{R}$ is β-gradient Lipschitz and Assumption E.2 holds for the stochastic gradient. Algorithm 8 outputs $\bar{x}_K$ such that*

$$\mathbb{E}h(\bar{x}_K) - h(x^*) \le \frac{\|x_0 - x^*\|^2}{2\eta K} + \frac{\eta\sigma^2}{2(1-\eta\beta)},$$

where $x^* = \arg\min_{x\in\mathbb{R}^{d_x}} h(x)$.

Based on the above result, we can analyze the SGD^{SC} algorithm (Allen-Zhu, 2018a, Algorithm 2), which achieves the optimal rate for smooth strongly-convex stochastic optimization. We present this algorithm in Algorithm 9. The convergence of Algorithm 9 can be found in (Allen-Zhu, 2018a, Theorem 4.1 (b)). However, they did not explicitly calculate the constants in the convergence rates. Therefore, we provide a self-contained proof below.

Algorithm 9 SGD^{SC} (h, x_0, N, K)

for $k = 1, \cdots, N$ **do** $x_k = \text{SGD}(h, x_{k-1}, 1/(2\beta), 4\beta/\alpha)$
for $k = 1, \cdots, K$ **do** $x_{N+k} = \text{SGD}(h, x_{N+k-1}, 1/(2^k\beta), 2^{k+2}\beta/\alpha)$
return x_{N+K}

Theorem E.5 *Suppose $h(x) : \mathbb{R}^d \to \mathbb{R}$ is β-gradient Lipschitz and α-strongly convex. Algorithm 9 outputs x_{N+K} satisfying*

$$\mathbb{E}\|x_{N+K} - x^*\|^2 \le \frac{\|x_0 - x^*\|^2}{2^{N+2K}} + \frac{\sigma^2}{2^{K-2}\alpha\beta}. \tag{45}$$

The total stochastic first-order oracle (SFO) complexity is bounded by $\lceil 4\kappa(N + 2^K)\rceil$, where $\kappa = \beta/\alpha$ is the condition number.

Proof Under the strong convexity assumption, Theorem E.4 yields that

$$\mathbb{E}\|\bar{x}_K - x^*\|^2 \le \frac{2}{\alpha}[\mathbb{E}h(\bar{x}_K) - h(x^*)] \le \frac{\|x_0 - x^*\|^2}{\alpha\eta K} + \frac{\eta\sigma^2}{\alpha(1-\eta\beta)}. \tag{46}$$

For the first N epochs, by the parameter setting in Algorithm 9 and Equation 46, we have

$$\mathbb{E}\|x_k - x^*\|^2 \leq \frac{\|x_{k-1} - x^*\|^2}{2} + \frac{\sigma^2}{\alpha\beta},$$

Telescoping for all the N epochs yields

$$\mathbb{E}\|x_N - x^*\|^2 \leq \frac{\|x_0 - x^*\|^2}{2^N} + \frac{2\sigma^2}{\alpha\beta}. \tag{47}$$

For the subsequent K epochs, by the parameter setting in Algorithm 9 and Equation 46, we have that

$$\mathbb{E}\|x_{N+k} - x^*\|^2 \leq \frac{\|x_{N+k-1} - x^*\|^2}{4} + \frac{\sigma^2}{2^{k-1}\alpha\beta}.$$

Then it is easy to prove by induction that

$$\mathbb{E}\|x_{N+k} - x^*\|^2 \leq \frac{\|x_0 - x^*\|^2}{2^{N+2k}} + \frac{\sigma^2}{2^{k-2}\alpha\beta}. \tag{48}$$

In the above, the induction base follows Equation 47. If we suppose that Equation 53 holds for $k = j - 1$, then for $k = j$ we have that

$$\begin{aligned}\mathbb{E}\|x_{N+j+1} - x^*\|^2 &\leq \frac{1}{4} \cdot \left(\frac{\|x_0 - x^*\|^2}{2^{N+2(j-1)}} + \frac{\sigma^2}{2^{j-3}\alpha\beta}\right) + \frac{\sigma^2}{2^{j-1}\alpha\beta} \\ &\leq \frac{\|x_0 - x^*\|^2}{2^{N+2j}} + \frac{\sigma^2}{2^{j-2}\alpha\beta},\end{aligned}$$

which completes the induction of Equation 48. ■

Appendix F. Proofs of Finding Second-Order Stationarity

The following theorem generalizes the result of Perturbed Gradient Descent (Jin et al., 2017) to Inexact Perturbed Gradient Descent by allowing an inexact gradient $\hat{\nabla}h(x)$ that is close to the exact gradient $\nabla h(x)$.

Algorithm 10 Inexact Perturbed Gradient Descent

for $t = 0, 1, \cdots, T-1$
  if $\|\hat{\nabla}h(x_t)\| \leq \frac{4}{5}\epsilon$ **and** no perturbation added in the last $\mathcal{T}$ steps
    $x_t = x_t - \eta\xi_t$, where $\xi_t \sim \mathbb{B}(r)$
  end if
  $x_{t+1} = x_t - \eta\hat{\nabla}h(x_t)$
end for

Theorem F.1 (Huang et al. (2025, Lemma 2.9)) *Suppose $h(z) : \mathbb{R}^d \to \mathbb{R}$ is L-gradient Lipschitz and ρ-Hessian Lipschitz. Set the parameters in Algorithm 10 as*

$$\eta = \frac{1}{L}, \quad r = \frac{\epsilon}{400\iota^3}, \quad \mathcal{T} = \frac{L}{\sqrt{\rho\epsilon}} \cdot \iota \ \text{ with } \ \iota \geq 1 \ \text{ and } \ \delta \geq \frac{L\sqrt{d}}{\sqrt{\rho\epsilon}} \iota^2 2^{8-\iota}. \tag{49}$$

Once the following condition holds in each iteration:

$$\|\hat{\nabla} h(x_t) - \nabla h(x_t)\| \leq \underbrace{\min\left\{\frac{1}{20\iota^2}, \frac{1}{16\iota^2 2^\iota}\right\}}_{:=\zeta} \epsilon, \tag{50}$$

then we can find a point x_t satisfying

$$\|\nabla h(x_t)\| \leq \epsilon, \quad \nabla^2 h(x_t) \succeq -\sqrt{\rho\epsilon}\, I_d$$

within $T = \mathcal{O}\left(\iota^4 \Delta L \epsilon^{-2}\right)$ iterations with probability $1-\delta$, where $\Delta = h(x_0) - \inf_{x\in\mathbb{R}^d} h(x)$.

Then we can show the convergence of perturbed F^{2}BA.

Theorem 5.1 *Suppose both Assumption 3.1 and 3.2 hold. Define $\Delta := \varphi(x_0) - \inf_{x\in\mathbb{R}^{d_x}} \varphi(x)$ and $R := \|y_0 - y^*(x_0)\|^2$. Let $\eta_x \asymp \ell^{-1}\kappa^{-3}$, $\lambda \asymp \max\left\{\kappa/R,\ \ell\kappa^2/\Delta,\ \ell\kappa^3/\epsilon,\ \kappa^{3.5}\sqrt{\ell/\epsilon}\right\}$ and set other parameters in Algorithm 1 as*

$$\eta_z = \frac{1}{L_g},\ \eta_y = \frac{1}{2\lambda L_g},\ r = \mathcal{O}(\epsilon),\ K_t = \tilde{\mathcal{O}}\left(\frac{L_g \log \delta_t}{\mu}\right),$$

where δ_t is defined via the recursion

$$\delta_{t+1} = \frac{1}{2}\delta_t + \frac{34L_g^2}{\mu^2}\|x_{t+1} - x_t\|^2, \quad \delta_0 = \mathcal{O}(R), \tag{8}$$

then it can find an ϵ-second-order stationary point of $\varphi(x)$ with probability at least $1-\delta$ within $\tilde{\mathcal{O}}\left(\ell\kappa^4\epsilon^{-2}\right)$ first-order oracle calls, where ℓ, κ are defined in Definition 3.3 and the notation $\tilde{\mathcal{O}}(\,\cdot\,)$ hides logarithmic factors of d_x, κ, ℓ, and δ, ϵ.

Proof Let L be the gradient Lipschitz coefficient and ρ be the Hessian Lipschitz coefficient of $\mathcal{L}_\lambda^*(x)$, respectively. According to Lemma 4.1, Lemma 5.1, Lemma B.2 and the setting of λ, we have

a. $\sup_{x\in\mathbb{R}^{d_x}} \|\nabla\mathcal{L}_\lambda^*(x) - \nabla\varphi(x)\| = \mathcal{O}(\epsilon)$.

b. $\sup_{x\in\mathbb{R}^{d_x}} \|\nabla^2\mathcal{L}_\lambda^*(x) - \nabla^2\varphi(x)\| = \mathcal{O}(\sqrt{\rho\epsilon})$.

c. $\mathcal{L}_\lambda^*(x_0) - \inf_{x\in\mathbb{R}^{d_x}} \mathcal{L}_\lambda^*(x) = \mathcal{O}(\Delta)$.

d. $\|y_0 - y_\lambda^*(x_0)\|^2 + \|y_0 - y^*(x_0)\|^2 = \mathcal{O}(R)$.

e. $L := \sup_{x\in\mathbb{R}^{d_x}} \|\nabla^2\mathcal{L}_\lambda^*(x)\| = \mathcal{O}(\ell\kappa^3)$.

f. $\rho := \sup_{x\in\mathbb{R}^{d_x}} \|\nabla^3 \mathcal{L}_\lambda^*(x)\| = \mathcal{O}(\ell\kappa^5)$.

Then it suffices to show that the algorithm can find an ϵ-second-order stationary point of $\mathcal{L}_\lambda^*(x)$ under our choice of parameters. We apply Theorem F.1 to prove this result. Recall ζ defined in Equation 50. It remains to show that

$$\|\nabla \mathcal{L}_\lambda^*(x_t) - \nabla \mathcal{L}_\lambda^*(x_t)\| \le \zeta \tag{51}$$

holds for all t. Recall that

$$\|\hat{\nabla} \mathcal{L}_\lambda^*(x_t) - \nabla \mathcal{L}_\lambda^*(x_t)\| \le 2\lambda L_g \|y_t^K - y_\lambda^*(x_t)\| + \lambda L_g \|z_t^K - y^*(x_t)\|.$$

If we let

$$K_t = \frac{4L_g}{\mu} \log\left(\frac{4\lambda L_g \delta_t}{\zeta}\right), \tag{52}$$

where $\delta_t \ge \|y_t^0 - y_\lambda^*(x_t)\|^2 + \|z_t^0 - y^*(x_t)\|^2$. Then by Theorem D.1, we have

$$\max\left\{\|y_t^K - y_\lambda^*(x_t)\|,\ \|z_t^K - y^*(x_t)\|\right\} \le \frac{\zeta}{4\lambda L_g}, \tag{53}$$

which implies Equation 51. By Equation 31, if we let $K_t \ge 8L_g/\mu$ for all t, then we can show that

$$\delta_{t+1} \le \frac{1}{2}\delta_t + \frac{34L_g^2}{\mu^2}\|x_{t+1} - x_t\|^2. \tag{54}$$

The total number of iterations can be bounded by

$$\sum_{t=0}^{T-1} K_t \le \frac{4L_g}{\mu} \sum_{t=0}^{T-1} \log\left(\frac{4\lambda L_g \delta_t}{\zeta}\right) \le \frac{4L_g T}{\mu} \log\left(\frac{4\lambda L_g \sum_{t=0}^{T-1}\delta_t}{\zeta T}\right) \tag{55}$$

first-order oracle calls. It remains to show that $\sum_{t=0}^{T-1}\delta_t$ is bounded. Telescoping over t in Equation 54, we obtain

$$\sum_{t=0}^{T-1}\delta_t \le 2\delta_0 + \frac{68L_g^2}{\mu^2}\sum_{t=0}^{T-1}\|x_{t+1} - x_t\|^2. \tag{56}$$

By Equation 34, if we let $K_t \ge \Omega(\kappa \log(\lambda\kappa))$ for all t, we can obtain

$$\frac{1}{8\eta}\sum_{t=0}^{T-1}\|x_{t+1} - x_t\|^2 \le \mathcal{L}_\lambda^*(x_0) - \inf_{x\in\mathbb{R}^{d_x}} \mathcal{L}_\lambda^*(x) + \|y_0 - y_\lambda^*(x_0)\|^2 + \|y_0 - y^*(x_0)\|^2.$$

Plugging into Equation 56 and then Equation 55 yields the upper complexity bound of $\tilde{\mathcal{O}}(\ell\kappa^4\epsilon^{-2})$ as claimed. ■

The following theorem generalizes the result of Nonconvex AGD (Li and Lin, 2023) by allowing an inexact gradient. To simplify the description, we define one round of the algorithm between two successive restarts to be one "epoch" by following Li and Lin (2023).

Algorithm 11 Inexact Nonconvex Accelerated Gradient Descent

1: $x_{-1} = x_0$
2: **while** $t < T$
3: $\quad x_{t+1/2} = x_t + (1-\theta)(x_t - x_{t-1})$
4: $\quad x_{t+1} = x_{t+1/2} - \eta\hat{\nabla} h(x_{t+1/2})$
5: $\quad t = t+1$
6: $\quad$ **if** $t\sum_{j=0}^{t-1} \|x_{j+1} - x_j\|^2 > B^2$
7: $\quad\quad t = 0$, $x_{-1} = x_0 = x_t + \xi_t \mathbf{1}_{\|\hat{\nabla} h(x_{t+1/2})\| \le \frac{B}{2\eta}}$, where $\xi_t \sim \mathbb{B}(r)$
8: $\quad$ **end if**
9: **end while**
10: $T_0 = \arg\min_{\lfloor \frac{T}{2} \rfloor \le t \le T-1} \|x_{t+1} - x_t\|$
11: **return** $x_{\text{out}} = \frac{1}{T_0+1}\sum_{t=0}^{T_0} x_{t+1/2}$

Theorem F.2 (Yang et al. (2023, Theorem 4.1)) *Suppose $h(z) : \mathbb{R}^d \to \mathbb{R}$ is L-gradient Lipschitz and ρ-Hessian Lipschitz. Assume that $\sqrt{\epsilon\rho} \le L$. Set the parameters in Algorithm 11 as*

$$\chi = \mathcal{O}(\log(d/\delta\epsilon)), \quad \eta = \frac{1}{4L}, \quad T = \frac{2\chi}{\theta}, \ \theta = \frac{1}{2}(\rho\epsilon\eta^2)^{1/4} < 1,$$
$$B = \frac{1}{288\chi^2}\sqrt{\frac{\epsilon}{\rho}}, \quad r = \min\left\{\frac{B + B^2}{\sqrt{2}}, \frac{\theta B}{20T}, \frac{\theta B^2}{2T}\right\} = \mathcal{O}(\epsilon).$$

Once the following condition holds in each iteration:

$$\|\hat{\nabla} h(x_t) - \nabla h(x_t)\| \le \underbrace{\min\left\{\frac{\rho B \zeta r \theta}{2\sqrt{d}}, \epsilon^2\right\}}_{:=\zeta}. \tag{57}$$

The algorithm ensures that

$$h(x_{\mathcal{T}}) - h(x_0) \le -\frac{\epsilon^{1.5}}{663552\sqrt{\rho}\chi^5}, \tag{58}$$

where $\mathcal{T}$ denotes the iteration number when "if" condition in Algorithm 11 (Line 6) triggers:

$$\mathcal{T} = \min_{t'}\left\{t' \mid t'\sum_{t=0}^{t'-1} \|x_{t+1} - x_t\|^2 > B^2\right\}.$$

Therefore, the algorithm terminates in at most $\mathcal{O}(\Delta\sqrt{\rho}\chi^5\epsilon^{-3/2})$ epochs and $\mathcal{O}(\Delta\sqrt{L}\rho^{1/4}\chi^6\epsilon^{-1.75})$ iterations. In addition, the output satisfies

$$\|\nabla h(x_t)\| \le \epsilon, \quad \nabla^2 h(x_t) \succeq -\sqrt{\rho\epsilon}\ I_d,$$

with probability $1-\delta$, where $\Delta = h(x_0) - \inf_{x\in\mathbb{R}^d} h(x)$.

Theorem 5.2 *Suppose both Assumption 3.1 and 3.2 hold. Define* $\Delta := \varphi(x_0) - \inf_{x\in\mathbb{R}^{d_x}} \varphi(x)$ *and* $R := \|y_0 - y^*(x_0)\|^2$. *Let* $\eta_x \asymp \ell^{-1}\kappa^{-3}$, $\lambda \asymp \max\left\{\kappa/R,\ \ell\kappa^2/\Delta,\ \ell\kappa^3/\epsilon,\ \kappa^{3.5}\sqrt{\ell/\epsilon}\right\}$ *and set other parameters in Algorithm 1 as*

$$\eta_z = \eta_y = \frac{1}{2\lambda L_g}, \quad K_t = \tilde{\mathcal{O}}\left(\frac{L_g \log \delta_t}{\mu}\right),$$
$$T \asymp \frac{\chi}{\theta},\ B \asymp \frac{1}{\chi^2}\sqrt{\frac{\epsilon}{\ell\kappa^3}}, \quad \theta \asymp \left(\frac{\ell\epsilon}{\kappa}\right)^{1/4},\ r = \mathcal{O}(\epsilon),$$

where $\chi = \mathcal{O}(\log(d_x/\delta\epsilon))$, *where* δ_t *is defined via the recursion*

$$\delta_{t+1} = \frac{1}{2}\delta_t + \frac{34L_g^2}{\mu^2}\|x_{t+1/2} - x^-_{t+1/2}\|^2, \quad \delta_0 = \mathcal{O}(R), \tag{9}$$

then it can find an ϵ*-second-order stationary point of* $\varphi(x)$ *with probability at least* $1-\delta$ *within* $\tilde{\mathcal{O}}\left(\kappa\ell^{1/2}\rho^{1/4}\epsilon^{-1.75}\right) = \tilde{\mathcal{O}}\left(\kappa^{3.75}\epsilon^{-1.75}\right)$ *first-order oracle calls, where* ℓ, κ *are defined in Definition 3.3 and the notation* $\tilde{\mathcal{O}}(\cdot)$ *hides logarithmic factors of* d_x, κ, ℓ, *and* δ, ϵ.

Proof Let T' be the number of total iterations. We renumber the iterates in the algorithm as $\{x_t\}_{t=0}^{T'-1}$, $\{x_{t+1/2}\}_{t=0}^{T'-1}$, and let the corresponding inner loop number as $\{K_t\}_{t=0}^{T'-1}$. Identical to the proof of Theorem 5.1 (with a different definition of ζ that only effects the logarithmic factors), we can set $K_t = \mathcal{O}(\kappa\log(\lambda L_g\delta_t/\zeta))$ to ensure Condition 57 to hold, where $\delta_t \geq \|y_t^0 - y_\lambda^*(x_{t+1/2})\|^2 + \|z_t^0 - y^*(x_{t+1/2})\|^2$ is ensured recursively identical to Equation 54. By Equation 55 we have

$$\sum_{t=0}^{T'-1} K_t \leq \frac{4L_gT}{\mu}\log\left(\frac{4\lambda L_g\sum_{t=0}^{T'-1}\delta_t}{\zeta T'}\right)$$

It remains to show that $\sum_{t=0}^{T'-1}\delta_t$ is bounded. By Equation 56, it suffices to show that $\sum_{t=0}^{T'-1}\|x_{t+3/2} - x_{t+1/2}\|^2$ is bounded. Since $x_{t+1/2} = x_t + (1-\theta)(x_t - x_{t-1})$, it suffices to show that $\sum_{t=0}^{T'-1}\|x_{t+1} - x_t\|^2$ is bounded. Because the number of epochs is finite by Theorem F.2, it suffices to bound the term in each epoch. We know that

$$\begin{aligned}
&\sum_{t=0}^{\mathcal{T}-1}\|x_{t+1} - x_t\|^2 \\
&= \sum_{t=0}^{\mathcal{T}-2}\|x_{t+1} - x_t\|^2 + \|x_{\mathcal{T}} - x_{\mathcal{T}-1}\|^2 \\
&\leq B^2 + \|x_{\mathcal{T}} - x_{\mathcal{T}-1}\|^2.
\end{aligned}$$

We continue to bound the last term $\|x_{\mathcal{T}} - x_{\mathcal{T}-1}\|^2$. Since we have

$$x_{t+1} - x_t = (1-\theta)(x_t - x_{t-1}) - \eta_x\hat{\nabla}\mathcal{L}_\lambda^*(x_{t+1/2}).$$

It remains to upper bound $\|\hat{\nabla}\mathcal{L}_\lambda^*(x_{\mathcal{T}-1/2})\|$. As $\|\hat{\nabla}\mathcal{L}_\lambda^*(x_{\mathcal{T}-1/2}) - \nabla\mathcal{L}_\lambda^*(x_{\mathcal{T}-1/2})\| = \mathcal{O}(\zeta)$, it remains to upper bound $\|\nabla\mathcal{L}_\lambda^*(x_{\mathcal{T}-1/2})\|$. By (18) in Yang et al. (2023), we have that

$$\mathcal{L}_\lambda^*(x_{\mathcal{T}}) \leq \mathcal{L}_\lambda^*(x_0) + \frac{B^2}{8\eta_x} - \frac{3\eta_x}{8}\|\nabla\mathcal{L}_\lambda^*(x_{\mathcal{T}-1/2})\|^2 + B\zeta + \frac{5\eta_x\zeta^2\mathcal{T}}{8}.$$

This implies that $\|\nabla\mathcal{L}_\lambda^*(x_{\mathcal{T}-1/2})\|$ is bounded once Δ is bounded. ■

Appendix G. Implement Our Algorithms in the Distributed Setting

Our algorithms can also be easily applied to the distributed scenario, when both the upper and lower-level functions adopt the following finite-sum structure:

$$f(x,y) := \frac{1}{m}\sum_{i=1}^m f_i(x,y), \quad g(x,y) := \frac{1}{m}\sum_{i=1}^m g_i(x,y), \tag{59}$$

where f_i and g_i denote the local function on the i-th agent. Under this setting, each agent i has its own local variables within the optimization algorithm: $X(i) \in \mathbb{R}^{d_x}, Y(i) \in \mathbb{R}^{d_y}, Z(i) \in \mathbb{R}^{d_y}$. For the convenience of presenting the algorithm, we aggregate all the local variables (denoted by row vectors) in one matrix and denote

$$X = \begin{bmatrix} X(1) \\ \vdots \\ X(m) \end{bmatrix} \in \mathbb{R}^{m\times d_x}, \quad Y = \begin{bmatrix} Y(1) \\ \vdots \\ Y(m) \end{bmatrix} \in \mathbb{R}^{m\times d_y} \quad \text{and} \quad Z = \begin{bmatrix} Z(1) \\ \vdots \\ Z(m) \end{bmatrix} \in \mathbb{R}^{m\times d_y},$$

We denote $\mathbf{1} = [1, \cdots, 1]^\top \in \mathbb{R}^m$ and use the lowercase with the bar to represent the mean vector, such as $\bar{x} = \frac{1}{m}\mathbf{1}^\top X$. Let $d = d_x + d_y$ and we similarly define the aggregated gradients

$$\nabla\mathbf{f}(X,Y) = \begin{bmatrix} \nabla f_1(X(1),Y(1)) \\ \vdots \\ \nabla f_m(X(m),Y(m)) \end{bmatrix} \in \mathbb{R}^{m\times d}, \ \nabla\mathbf{g}(X,Y) = \begin{bmatrix} \nabla g_1(X(1),Y(1)) \\ \vdots \\ \nabla g_m(X(m),Y(m)) \end{bmatrix} \in \mathbb{R}^{m\times d}.$$

The classic paradigm for distributed algorithms is to compute the local gradients on each agent in parallel, and then aggregate them on the server. However, challenges arise when extending existing HVP-based methods for bilevel optimization from the single-machine setting to the distributed setting. Given a variable $\bar{x}$, one may want to calculate its hyper-gradient (Equation 2) according to this paradigm by:

$$\hat{\nabla}\varphi(\bar{x}) = \mathbf{\Pi}\left(\nabla_x\mathbf{f}(\mathbf{1}x, \mathbf{1}y^*(\bar{x})) - \nabla_{xy}^2\mathbf{f}(\mathbf{1}x, \mathbf{1}y^*(\bar{x}))[\nabla_{yy}^2\mathbf{g}(\mathbf{1}\bar{x}, \mathbf{1}y^*(\bar{x}))]^{-1}\nabla_y\mathbf{f}(\mathbf{1}\bar{x}, \mathbf{1}y^*(\bar{x}))\right),$$

where $y^*(x) := \arg\min_{y\in\mathbb{R}^{d_y}} g(x,y)$ and $\mathbf{\Pi} := \frac{1}{m}\mathbf{1}\mathbf{1}^\top$ denotes the aggregation operator on the server. But the nested structure of the hyper-gradient indicates that $\hat{\nabla}\varphi(\bar{x}) \neq \nabla\varphi(\bar{x})$. As a consequence, researchers need to pay extra effort to make HVP-based methods work in distributed bilevel problems (Tarzanagh et al., 2022; Chen et al., 2023). It is a non-trivial

Algorithm 12 Distributed F^{2}BA $(\bar{x}_0, \bar{y}_0, \eta_x, \eta_y, \eta_z, \lambda, T, K)$

$X_0 = \mathbf{1}\bar{x}_0,\ Y_0 = \mathbf{1}\bar{y}_0,\ Z_0 = \mathbf{1}\bar{y}_0$
for $t = 0, 1, \cdots, T-1$
  $Y_t^0 = Y_t,\ Z_t^0(i) = Z_t$
  for $k = 0, 1, \cdots, K-1$
    $V_t^k = \lambda \nabla_y \mathbf{g}(X_t, Z_t^k), \quad U_t^k = \nabla_y \mathbf{f}(X_t, Y_t^k) + \lambda \nabla_y \mathbf{g}(X_t, Y_t^k)$
    Aggregate and broadcast $\bar{v}_t^k = \frac{1}{m}\mathbf{1}\mathbf{1}^\top V_t^k,\ \bar{u}_t^k = \frac{1}{m}\mathbf{1}\mathbf{1}^\top U_t^k$
    $Z_t^{k+1} = Z_t^k - \eta_z \mathbf{1}\bar{v}_t^k\ , \quad Y_t^{k+1} = Y_t^k - \eta_y \mathbf{1}\bar{u}_t^k$
  end for
  $H_t = \nabla_x \mathbf{f}(X_t, Y_t^K) + \lambda(\nabla_x \mathbf{g}(X_t, Y_t^K) - \nabla_x \mathbf{g}(X_t, Z_t^K))$
  Aggregate and broadcast $\bar{h}_t = \frac{1}{m}\mathbf{1}\mathbf{1}^\top H_t$
  $X_{t+1} = X_t - \eta_x \mathbf{1}\bar{h}_t$
end for

issue to address, especially when the operator $\mathbf{\Pi}\nabla^2_{yy}\mathbf{g}(\mathbf{1}\bar{x}, \mathbf{1}y^*(\bar{x}))$ is forbidden to avoid an unacceptable $\mathcal{O}(d_y^2)$ communication complexity.

In contrast, the distributed extension of F^{2}BA naturally avoid this challenge since

$$\nabla \mathcal{L}_\lambda^*(\bar{x}) = \mathbf{\Pi}\left(\nabla_x \mathbf{f}(\mathbf{1}x, \mathbf{1}y_\lambda^*(\bar{x})) + \lambda(\nabla_x \mathbf{g}(\mathbf{1}\bar{x}, \mathbf{1}y_\lambda^*(\bar{x})) - \nabla_x \mathbf{g}(\mathbf{1}\bar{x}, \mathbf{1}y^*(\bar{x})))\right),$$

where $y_\lambda^*(x) := \arg\min_{y \in \mathbb{R}^{d_y}} f(x,y) + \lambda g(x,y)$. It means that the previously mentioned classic aggregation paradigm for distributed optimization directly works for F^{2}BA without any additional modification in the algorithm, as stated below.

Proposition G.1 *Running Algorithm 12 is equivalent to running Algorithm 1 on the mean variables*

$$\bar{x} = \frac{1}{m}\mathbf{1}^\top X,\ \bar{y} = \frac{1}{m}\mathbf{1}^\top Y \text{ and } \bar{z} = \frac{1}{m}\mathbf{1}^\top Z.$$

Then the convergence of Algorithm 12 directly follows Theorem 4.1.

Corollary G.1 *Suppose Assumption 3.1 holds. Algorithm 12 with the same parameters in Theorem 4.1 can find X_t satisfying $\|\nabla\varphi(\bar{x}_t)\| \le \epsilon$ within $\mathcal{O}(\ell\kappa^4\epsilon^{-2}\log(\ell\kappa/\epsilon))$ iterations and $\mathcal{O}(\ell\kappa^4\epsilon^{-2}\log(\ell\kappa/\epsilon))$ communication rounds.*

Algorithm 12 is a near-optimal distributed algorithm since both the iteration and communication complexity match the $\Omega(\epsilon^{-2})$ lower bound (Theorem 1 by Lu and De Sa (2021)), up to logarithmic factors. Compared with the HVP-based methods, the distributed F^{2}BA is more practical since it neither requires a $\mathcal{O}(d_y^2)$ communication complexity per iteration (Yang et al., 2022) nor an additional distributed sub-solver for matrix inverse (Tarzanagh et al., 2022; Chen et al., 2023).

We remark that all algorithms in our main text can be implemented in a similar way as we implement F^{2}BA in the distributed scenario.

Appendix H. Discussions on Closely Related Works

In this section, we discuss our method with closely related works, including (Shen and Chen, 2023; Kwon et al., 2023). Both our work and theirs use a penalty method. The main difference in algorithm design is that their method uses a single-time-scale update in (x, y) while we use a two-time-scale update to improve their $\mathcal{O}(\lambda\epsilon^{-2}\log(1/\epsilon))$ first-order complexity to $\mathcal{O}(\epsilon^{-2}\log\lambda)$, where $\lambda \asymp \epsilon^{-1}$ due to Lemma 4.1.

In the following, we give a more detailed comparison of another aspects, including the optimality notion and assumptions of our work and theirs.

H.1 Discussions on the Weaker Notions in (Shen and Chen, 2023)

Shen and Chen (2023) also proposed a similar penalty-based algorithm, and proved that the penalty method can find an ϵ-stationary point (formally defined below) of $\mathcal{L}_\lambda(x, y)$ in $\mathcal{O}(\lambda\epsilon^{-2}\log(1/\epsilon))$ first-order oracle calls. Below, we show that their result implies a $\mathcal{O}(\epsilon^{-3}\log(1/\epsilon)$ first-order complexity to find an ϵ-stationary point of $\varphi(x)$.

Definition H.1 *We say (x, y) is an ϵ-stationary point of $\mathcal{L}_\lambda(x, y)$ if $\|\nabla\mathcal{L}_\lambda(x, y)\| \le \epsilon$.*

We show that their notion and ours can be translated in both directions.

Lemma H.1 *Suppose Assumption 3.1 and 4.1 hold. Set λ as Theorem 4.2.*

1. *If a point x is an ϵ-stationary point of $\varphi(x)$ in terms of Definition 3.2, then we can find a point y such that (x, y) is an 2ϵ-stationary point of $\mathcal{L}_\lambda(x, y)$ in terms of Definition H.1 with $\mathcal{O}(\kappa\log(\kappa/\epsilon))$, $\mathcal{O}(\kappa^2\epsilon^{-2})$, or $\mathcal{O}(\kappa^8\epsilon^{-2})$ first-order oracle calls under the deterministic ($\sigma_f = \sigma_g = 0$), partially stochastic ($\sigma_f > 0, \sigma_g = 0$) and fully stochastic setting ($\sigma_f > 0, \sigma_g > 0$) respectively.*
2. *If a point (x, y) is an ϵ-stationary point of $\mathcal{L}_\lambda(x, y)$ in terms of Definition H.1, then it is a $\mathcal{O}(\kappa\epsilon)$-stationary point of $\varphi(x)$ in terms of Definition 3.2.*

Proof Note that $\varphi(x) = \min_{y\in\mathbb{R}^{d_y}} \mathcal{L}_\lambda(x, y)$. Then by Danskin's lemma, we have $\nabla\varphi(x) = \nabla_x\mathcal{L}_\lambda(x, y_\lambda^*(x))$. Therefore, we further have

$$\begin{aligned}\|\nabla\varphi(x)\| &\le \|\nabla_x\mathcal{L}_\lambda(x, y)\| + 2\lambda L_g\|y_\lambda^*(x) - y^*(x)\| \\ &\le \|\nabla_x\mathcal{L}_\lambda(x, y)\| + \frac{4L_g}{\mu}\|\nabla_y\mathcal{L}_\lambda(x, y)\|,\end{aligned}$$

where we use the fact that $\mathcal{L}_\lambda(x, y)$ is $(2\lambda L_g)$-gradient Lipschitz and $(\lambda\mu/2)$-strongly convex in y. This shows that if (x, y) is an ϵ-stationary point of $\mathcal{L}_\lambda(x, y)$, then x is also an $\mathcal{O}(\kappa\epsilon)$-stationary point of $\varphi(x)$. **Conversely**, if x is an ϵ-stationary point of $\varphi(x)$, the it suffices to find y such that $\|\nabla_y f(x, y)\| \le \mathcal{O}(\epsilon/\kappa)$, then (x, y) is a 2ϵ-stationary point of $\mathcal{L}_\lambda(x, y)$. Under the deterministic setting, this goal can be achieved with $\mathcal{O}(\kappa\log(\kappa/\epsilon))$ first-order oracle calls by Theorem D.1. Under the stochastic setting, this goal can be achieved with $\mathcal{O}(\kappa^2(\sigma_f^2 + \lambda^2\sigma_g^2)\epsilon^{-2})$ first-order oracle calls by (Allen-Zhu, 2018a, Theorem 3). ■

Note that one has to pay an additional $\mathcal{O}(\kappa)$ factor when translating a stationary point of $\mathcal{L}_\lambda(x, y)$ using the notation of Shen and Chen (2023) to that of $\varphi(x)$ using our notation. Therefore, our notation is stronger than that of Shen and Chen (2023).

H.2 Discussion on the Additional Assumptions in (Kwon et al., 2023)

F^{2}SA (Kwon et al., 2023, Algorithm 1) additionally uses the following assumption.

Assumption H.1 *[Assumption 4 and 5 in Kwon et al. (2023)] Besides Assumption 3.1, they also assume that*

1. *$f(x, y)$ is C_f-Lipschitz in x;*
2. *$g(x, y)$ is C_g-Lipschitz in y;*
3. *$f(x, y)$ is ρ_f-Hessian Lipschitz in (x, y).*

The role of Assumption H.1 is to ensure the Lipschitz continuity of $y_\lambda^*(x)$ in λ such that the algorithm can use an increasing λ. However, this additional assumption is unnecessary if we use a fixed λ. Kwon et al. (2023) sets $\lambda_{t+1} = \lambda_t + \delta_t$ with the requirement

$$\frac{\delta_t}{\lambda_t} \leq \frac{1}{16} \min\left\{1, \frac{T\mu}{L_g}\right\}.$$

See (Kwon et al., 2023, Theorem 4, Condition (3b)). Therefore, setting $\delta_t = 0$ also meets the requirement. In this case, when we discuss the algorithm by Kwon et al. (2023), we refer to this version with a fixed λ. This allows a fair comparison with our algorithm. After both algorithms use the same penalty λ, the only difference lies in our step size, which is the key to our improved analysis.

References

Naman Agarwal, Zeyuan Allen-Zhu, Brian Bullins, Elad Hazan, and Tengyu Ma. Finding approximate local minima faster than gradient descent. In *SIGACT*, 2017.

Zeyuan Allen-Zhu. How to make the gradients small stochastically: Even faster convex and nonconvex SGD. In *NeurIPS*, 2018a.

Zeyuan Allen-Zhu. Natasha 2: Faster non-convex optimization than SGD. In *NeurIPS*, 2018b.

Zeyuan Allen-Zhu and Yuanzhi Li. Neon2: Finding local minima via first-order oracles. In *NeurIPS*, 2018.

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *NeurIPS*, 2016.

Yossi Arjevani, Yair Carmon, John C Duchi, Dylan J Foster, Ayush Sekhari, and Karthik Sridharan. Second-order information in non-convex stochastic optimization: Power and limitations. In *COLT*, 2020.

Yossi Arjevani, Yair Carmon, John C. Duchi, Dylan J. Foster, Nathan Srebro, and Blake Woodworth. Lower bounds for non-convex stochastic optimization. *Mathematical Programming*, 199(1-2):165–214, 2023.

Hilal Asi, Yair Carmon, Arun Jambulapati, Yujia Jin, and Aaron Sidford. Stochastic bias-reduced gradient methods. In *NeurIPS*, 2021.

Juhan Bae and Roger B. Grosse. Delta-STN: Efficient bilevel optimization for neural networks using structured response jacobians. In *NeurIPS*, 2020.

Nicholas Bishop, Long Tran-Thanh, and Enrico Gerding. Optimal learning from verified training data. In *NeurIPS*, 2020.

Jose H. Blanchet and Peter W. Glynn. Unbiased monte carlo for optimization and functions of expectations via multi-level randomization. In *WSC*, 2015.

Michael Brückner and Tobias Scheffer. Stackelberg games for adversarial prediction problems. In *SIGKDD*, 2011.

Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

Yair Carmon, John C. Duchi, Oliver Hinder, and Aaron Sidford. "Convex Until Proven Guilty": Dimension-free acceleration of gradient descent on non-convex functions. In *ICML*, 2017.

Yair Carmon, John C. Duchi, Oliver Hinder, and Aaron Sidford. Lower bounds for finding stationary points I. *Mathematical Programming*, 184(1-2):71–120, 2020.

Yair Carmon, John C. Duchi, Oliver Hinder, and Aaron Sidford. Lower bounds for finding stationary points II: first-order methods. *Mathematical Programming*, 185(1):315–355, 2021.

Kartik Chandra, Audrey Xie, Jonathan Ragan-Kelley, and Erik Meijer. Gradient descent: The ultimate optimizer. In *NeurIPS*, 2022.

Lesi Chen, Jing Xu, and Jingzhao Zhang. On finding small hyper-gradients in bilevel optimization: Hardness results and improved analysis. In *COLT*, 2024a.

Tianyi Chen, Yuejiao Sun, and Wotao Yin. Closing the gap: Tighter analysis of alternating stochastic gradient methods for bilevel problems. In *NeurIPS*, 2021.

Tianyi Chen, Yuejiao Sun, and Wotao Yin. A single-timescale stochastic bilevel optimization method. In *AISTATS*, 2022.

Xuxing Chen, Minhui Huang, Shiqian Ma, and Krishnakumar Balasubramanian. Decentralized stochastic bilevel optimization with improved per-iteration complexity. In *ICML*, 2023.

Xuxing Chen, Tesi Xiao, and Krishnakumar Balasubramanian. Optimal algorithms for stochastic bilevel optimization under relaxed smoothness conditions. *JMLR*, 2024b.

Mathieu Dagréou, Pierre Ablin, Samuel Vaiter, and Thomas Moreau. A framework for bilevel optimization that enables stochastic and global variance reduction algorithms. In *NeurIPS*, 2022.

Yann N. Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *NIPS*, 2014.

Stephan Dempe. *Foundations of bilevel programming*. Springer Science & Business Media, 2002.

Justin Domke. Generic methods for optimization-based modeling. In *AISTATS*, 2012.

John C. Duchi, Michael I. Jordan, Martin J. Wainwright, and Andre Wibisono. Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Transactions on Information Theory*, 61(5):2788–2806, 2015.

Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. On the convergence theory of gradient-based model-agnostic meta-learning algorithms. In *AISTATS*, 2020.

Cong Fang, Zhouchen Lin, and Tong Zhang. Sharp analysis for nonconvex SGD escaping from saddle points. In *COLT*, 2019.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

Luca Franceschi, Michele Donini, Paolo Frasconi, and Massimiliano Pontil. Forward and reverse gradient-based hyperparameter optimization. In *ICML*, 2017.

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *ICML*, 2018.

Jiahui Gao, Renjie Pi, LIN Yong, Hang Xu, Jiacheng Ye, Zhiyong Wu, WeiZhong Zhang, Xiaodan Liang, Zhenguo Li, and Lingpeng Kong. Self-guided noise-free data generation for efficient zero-shot learning. In *ICLR*, 2022.

Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. Escaping from saddle points—online stochastic gradient for tensor decomposition. In *COLT*, 2015.

Rong Ge, Jason D. Lee, and Tengyu Ma. Matrix completion has no spurious local minimum. In *NeurIPS*, 2016.

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

Saeed Ghadimi and Mengdi Wang. Approximation methods for bilevel programming. *arXiv preprint arXiv:1802.02246*, 2018.

Michael B. Giles. Multilevel monte carlo path simulation. *Operations research*, 2008.

Riccardo Grazzi, Luca Franceschi, Massimiliano Pontil, and Saverio Salzo. On the iteration complexity of hypergradient computation. In *ICML*, 2020.

Elad Hazan and Satyen Kale. Beyond the regret minimization barrier: optimal algorithms for stochastic strongly-convex optimization. *JMLR*, 2014.

Mingyi Hong, Hoi-To Wai, Zhaoran Wang, and Zhuoran Yang. A two-timescale stochastic algorithm framework for bilevel optimization: Complexity analysis and application to actor-critic. *SIAM Journal on Optimization*, 33(1):147–180, 2023.

Yifan Hu, Xin Chen, and Niao He. On the bias-variance-cost tradeoff of stochastic optimization. In *NeurIPS*, 2021.

Minhui Huang, Kaiyi Ji, Shiqian Ma, and Lifeng Lai. Efficiently escaping saddle points in bilevel optimization. *JMLR*, 2025.

Kaiyi Ji, Junjie Yang, and Yingbin Liang. Bilevel optimization: Convergence analysis and enhanced design. In *ICML*, 2021.

Kaiyi Ji, Junjie Yang, and Yingbin Liang. Theoretical convergence of multi-step model-agnostic meta-learning. *JMLR*, 2022.

Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M. Kakade, and Michael I. Jordan. How to escape saddle points efficiently. In *ICML*, 2017.

Chi Jin, Praneeth Netrapalli, and Michael I Jordan. Accelerated gradient descent escapes saddle points faster than gradient descent. In *COLT*, 2018.

Prashant Khanduri, Siliang Zeng, Mingyi Hong, Hoi-To Wai, Zhaoran Wang, and Zhuoran Yang. A near-optimal algorithm for stochastic bilevel optimization via double-momentum. In *NeurIPS*, 2021.

Prashant Khanduri, Ioannis Tsaknakis, Yihua Zhang, Jia Liu, Sijia Liu, Jiawei Zhang, and Mingyi Hong. Linearly constrained bilevel optimization: A smoothed implicit gradient approach. In *ICML*, 2023.

Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

Guy Kornowski and Ohad Shamir. An algorithm with optimal dimension-dependence for zero-order nonsmooth nonconvex stochastic optimization. *arXiv preprint arXiv:2307.04504*, 2023.

Guy Kornowski, Swati Padmanabhan, Kai Wang, Zhe Zhang, and Suvrit Sra. First-order methods for linearly constrained bilevel optimization. In *NeurIPS*, 2024.

Jeongyeol Kwon, Dohyun Kwon, Stephen Wright, and Robert Nowak. A fully first-order method for stochastic bilevel optimization. In *ICML*, 2023.

Jeongyeol Kwon, Dohyun Kwon, and Hanbaek Lyu. On the complexity of first-order methods in stochastic bilevel optimization. In *ICML*, 2024a.

Jeongyeol Kwon, Dohyun Kwon, Stephen Wright, and Robert D Nowak. On penalty methods for nonconvex bilevel optimization and first-order stochastic approximation. In *ICLR*, 2024b.

Jason D. Lee, Max Simchowitz, Michael I. Jordan, and Benjamin Recht. Gradient descent only converges to minimizers. In *COLT*, 2016.

Haoyang Li, Xin Wang, Ziwei Zhang, and Wenwu Zhu. OOD-GNN: Out-of-distribution generalized graph neural network. *TKDE*, 2022.

Huan Li and Zhouchen Lin. Restarted nonconvex accelerated gradient descent: No more polylogarithmic factor in the in the $\mathcal{O}(\epsilon^{-7/4})$ complexity. *JMLR*, 2023.

Renjie Liao, Yuwen Xiong, Ethan Fetaya, Lisa Zhang, KiJung Yoon, Xaq Pitkow, Raquel Urtasun, and Richard Zemel. Reviving and improving recurrent back-propagation. In *ICML*, 2018.

Gui-Hua Lin, Mengwei Xu, and Jane J. Ye. On solving simple bilevel programs with a nonconvex lower level program. *Mathematical Programming*, 144(1-2):277–305, 2014.

Bo Liu, Mao Ye, Stephen Wright, Peter Stone, and Qiang Liu. BOME! bilevel optimization made easy: A simple first-order approach. In *NeurIPS*, 2022.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *ICLR*, 2019.

Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *AISTATS*, 2020.

Yucheng Lu and Christopher De Sa. Optimal complexity in decentralized training. In *ICML*, 2021.

Matthew Mackay, Paul Vicol, Jonathan Lorraine, David Duvenaud, and Roger Grosse. Self-tuning networks: Bilevel optimization of hyperparameters using structured best-response functions. In *ICML*, 2018.

Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *ICML*, 2015.

Konstantin Mishchenko, Slavomír Hanzely, and Peter Richtárik. Convergence of first-order algorithms for meta-learning with moreau envelopes. *arXiv preprint arXiv:2301.06806*, 2023.

Yurii Nesterov. A method for solving the convex programming problem with convergence rate $\mathcal{O}(1/k2)$. In *Dokl akad nauk Sssr*, volume 269, page 543, 1983.

Yurii Nesterov and Boris T. Polyak. Cubic regularization of newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Jiří V. Outrata. On the numerical solution of a class of stackelberg problems. *Zeitschrift für Operations Research*, 34:255–277, 1990.

Rui Pan, Jipeng Zhang, Xingyuan Pan, Renjie Pi, Xiaoyu Wang, and Tong Zhang. Scalebio: Scalable bilevel optimization for llm data reweighting. *arXiv preprint arXiv:2406.19976*, 2024.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

Fabian Pedregosa. Hyperparameter optimization with approximate gradient. In *ICML*, 2016.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 2019.

Aravind Rajeswaran, Chelsea Finn, Sham M. Kakade, and Sergey Levine. Meta-learning with implicit gradients. In *NeurIPS*, 2019.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *ICML*, 2018.

Alexander Robey, Fabian Latorre, George J. Pappas, Hamed Hassani, and Volkan Cevher. Adversarial training should be cast as a non-zero-sum game. *arXiv preprint arXiv:2306.11035*, 2023.

Amirreza Shaban, Ching-An Cheng, Nathan Hatch, and Byron Boots. Truncated back-propagation for bilevel optimization. In *AISTATS*, 2019.

Han Shen and Tianyi Chen. On penalty-based bilevel gradient descent method. In *ICML*, 2023.

Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. In *NeurIPS*, 2019.

Xingyou Song, Wenbo Gao, Yuxiang Yang, Krzysztof Choromanski, Aldo Pacchiano, and Yunhao Tang. ES-MAML: Simple hessian-free meta learning. In *ICLR*, 2019.

Daouda Sow, Kaiyi Ji, and Yingbin Liang. On the convergence theory for hessian-free bilevel algorithms. In *NeurIPS*, 2022.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

Davoud Ataee Tarzanagh, Mingchen Li, Christos Thrampoulidis, and Samet Oymak. Fednest: Federated bilevel, minimax, and compositional optimization. In *ICML*, 2022.

Nilesh Tripuraneni, Mitchell Stern, Chi Jin, Jeffrey Regier, and Michael I. Jordan. Stochastic cubic regularization for fast nonconvex optimization. In *NeurIPS*, 2018.

Ioannis Tsaknakis, Prashant Khanduri, and Mingyi Hong. An implicit gradient-type method for linearly constrained bilevel problems. In *ICASSP*, 2022.

Jiali Wang, He Chen, Rujun Jiang, Xudong Li, and Zihao Li. Fast algorithms for stackelberg prediction game with least squares loss. In *ICML*, 2021.

Jiali Wang, Wen Huang, Rujun Jiang, Xudong Li, and Alex L. Wang. Solving stackelberg prediction game with least squares loss via spherically constrained least squares reformulation. In *ICML*, 2022a.

Nuozhou Wang, Junyu Zhang, and Shuzhong Zhang. Efficient first order method for saddle point problems with higher order smoothness. *SIAM Journal on Optimization*, 34(4): 3342–3370, 2024.

Xiaoxing Wang, Wenxuan Guo, Jianlin Su, Xiaokang Yang, and Junchi Yan. ZARTS: On zero-order optimization for neural architecture search. In *NeurIPS*, 2022b.

Quan Xiao, Han Shen, Wotao Yin, and Tianyi Chen. Alternating projected SGD for equality-constrained bilevel optimization. In *AISTATS*, 2023.

Yi Xu, Rong Jin, and Tianbao Yang. First-order stochastic algorithms for escaping from saddle points in almost linear time. In *NeurIPS*, 2018.

Haikuo Yang, Luo Luo, Chris Junchi Li, and Michael I. Jordan. Accelerating inexact hypergradient descent for bilevel optimization. *arXiv preprint arXiv:2307.00126*, 2023.

Shuoguang Yang, Xuezhou Zhang, and Mengdi Wang. Decentralized gossip-based stochastic bilevel optimization over communication networks. In *NeurIPS*, 2022.

Jane J. Ye and Daoli Zhu. Optimality conditions for bilevel programming problems. *Optimization*, 33(1):9–27, 1995.

Jane J. Ye and Daoli Zhu. New necessary optimality conditions for bilevel programs by combining the mpec and value function approaches. *SIAM Journal on Optimization*, 20 (4):1885–1905, 2010.

Lin Yong, Renjie Pi, Weizhong Zhang, Xiaobo Xia, Jiahui Gao, Xiao Zhou, Tongliang Liu, and Bo Han. A holistic view of label noise transition matrix in deep learning and beyond. In *ICLR*, 2022.

Chenyi Zhang and Tongyang Li. Escape saddle points by a simple gradient-descent based algorithm. In *NeurIPS*, 2021.

Miao Zhang, Steven W Su, Shirui Pan, Xiaojun Chang, Ehsan M Abbasnejad, and Reza Haffari. iDARTS: Differentiable architecture search with stochastic implicit gradients. In *ICML*, 2021.

Yihua Zhang, Guanhua Zhang, Prashant Khanduri, Mingyi Hong, Shiyu Chang, and Sijia Liu. Revisiting and advancing fast adversarial training through the lens of bi-level optimization. In *ICML*, 2022.

Dongruo Zhou, Pan Xu, and Quanquan Gu. Stochastic nested variance reduction for nonconvex optimization. *JMLR*, 2020.

Pan Zhou, Xiaotong Yuan, Huan Xu, Shuicheng Yan, and Jiashi Feng. Efficient meta learning via minibatch proximal update. In *NeurIPS*, 2019.

Xiao Zhou, Yong Lin, Renjie Pi, Weizhong Zhang, Renzhe Xu, Peng Cui, and Tong Zhang. Model agnostic sample reweighting for out-of-distribution learning. In *ICML*, 2022.

Barret Zoph and Quoc Le. Neural architecture search with reinforcement learning. In *ICLR*, 2016.