Faster Gradient Methods for Highly-smooth Stochastic Bilevel Optimization

Lesi Chen CHENLC23@MAILS.TSINGHUA.EDU.CN
Junru Li JR-LI24@MAILS.TSINGHUA.EDU.CN
Tsinghua University, China
El Mahdi Chayti EL-MAHDI.CHAYTI@EPFL.CH
EPFL, Switzerland
Jingzhao Zhang JINGZHAOZ@MAIL.TSINGHUA.EDU.CN
Tsinghua University, China

Abstract

This paper studies the complexity of finding an ϵ-stationary point for stochastic bilevel optimization when the upper-level problem is nonconvex and the lower-level problem is strongly convex. Recent work proposed the first-order method, F^{2}SA, achieving the $\tilde{\mathcal{O}}(\epsilon^{-6})$ upper complexity bound for first-order smooth problems. This is slower than the optimal $\Omega(\epsilon^{-4})$ complexity lower bound in its single-level counterpart. In this work, we show that faster rates are achievable for higher-order smooth problems. We first reformulate F^{2}SA as approximating the hyper-gradient with a forward difference. Based on this observation, we propose a class of methods F^{2}SA-p that uses pth-order finite difference for hyper-gradient approximation and improves the upper bound to $\tilde{\mathcal{O}}(p\epsilon^{-4-2/p})$ for pth-order smooth problems. Finally, we demonstrate that the $\Omega(\epsilon^{-4})$ lower bound also holds for stochastic bilevel problems when the high-order smoothness holds for the lower-level variable, indicating that the upper bound of F^{2}SA-p is nearly optimal in the highly smooth region $p = \Omega(\log \epsilon^{-1} / \log\log \epsilon^{-1})$.

1. Introduction

Many machine learning problems, such as meta-learning [39], hyper-parameter tuning [5, 17, 36], and adversarial training [19] can be abstracted as solving the bilevel optimization problem:

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

We call f and g the upper-level and lower-level functions, respectively, and call φ the *hyper-objective*. In this paper, we consider the most common *nonconvex-strongly-convex* setting where $f : \mathbb{R}^{d_x} \to \mathbb{R}$ is smooth and possibly nonconvex, and $g : \mathbb{R}^{d_y} \to \mathbb{R}$ is smooth jointly in $(\boldsymbol{x}, \boldsymbol{y})$ and strongly convex in $\boldsymbol{y}$. Under the lower-level strong convexity assumption, the implicit function theorem indicates the following closed form of the hyper-gradient [18]:

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

Following the works in nonconvex optimization [3, 6, 7], we consider the task of finding an ϵ-stationary point of φ, *i.e.*, a point $\boldsymbol{x} \in \mathbb{R}^{d_x}$ such that $\|\nabla\varphi(\boldsymbol{x})\| \leq \epsilon$. Motivated by many real machine learning tasks, we study the stochastic setting, where the algorithms only have access to the stochastic derivative estimators of both f and g.

The first efficient algorithm BSA Ghadimi and Wang [18] for solving the stochastic bilevel problem leverages both stochastic gradient and Hessian-vector-product (HVP) oracles to find an ϵ-stationary point of $\varphi(\boldsymbol{x})$. Subsequently, Ji et al. [22] proposed stocBiO by incorporating multiple enhanced designs to improve the complexity. Both BSA and stocBiO require the stochastic Hessian assumption (6) on the lower-level function, which means g has an unbiased stochastic Hessian estimator with bounded variance. For finite-sum problems, such an assumption is stronger than standard SGD assumptions and equivalent to the mean-squared-smoothness assumption (7) on the lower-level gradient estimator G [2, Observation 1 and 2].

To avoid estimating HVP oracles, Kwon et al. [28] proposed the first fully first-order method F^{2}SA that works under standard SGD assumptions on both f and g (Assumption 2.1). The main idea is to solve the following penalty problem [33, 34, 41]:

$$\min_{\boldsymbol{x}\in\mathbb{R}^{d_x},\boldsymbol{y}\in\mathbb{R}^{d_y}} f(\boldsymbol{x},\boldsymbol{y}) + \lambda\left(g(\boldsymbol{x},\boldsymbol{y}) - \min_{\boldsymbol{z}\in\mathbb{R}^{d_y}} g(\boldsymbol{x},\boldsymbol{z})\right), \tag{3}$$

where λ is taken to be sufficiently large such that $\lambda = \Omega(\epsilon^{-1})$. Thanks to Danskin's theorem, the gradient of the penalty function in Eq. (3) only involves gradient information. Therefore, F^{2}SA does not require the stochastic Hessian assumptions (6). More importantly, by directly leveraging gradient oracles instead of more expensive HVP oracles, the F^{2}SA is more efficient in practice [23, 40, 42] and it is also the only method that can be scaled to 32B sized large language model (LLM) training [38].

Kwon et al. [28] proved that the F^{2}SA method finds an ϵ-stationary point of $\varphi(\boldsymbol{x})$ with $\tilde{\mathcal{O}}(\epsilon^{-3})$ first-order oracle calls in the deterministic case and $\tilde{\mathcal{O}}(\epsilon^{-7})$ stochastic first-order oracle (SFO) calls in the stochastic case. Recently, Chen et al. [11] showed the two-time-scale stepsize strategy improves the upper complexity bound of F^{2}SA method to $\tilde{\mathcal{O}}(\epsilon^{-2})$ in the deterministic case, which is optimal up to logarithmic factors. However, the direct extension of their method in the stochastic case leads to the $\tilde{\mathcal{O}}(\epsilon^{-6})$ SFO complexity [11, 29] , which still has a significant gap between the $\Omega(\epsilon^{-4})$ lower bound for SGD [3]. It remains open whether optimal rates for stochastic bilevel problems can be achieved for fully first-order methods.

In this work, we revisit F^{2}SA and interpret it as using forward difference to approximate the hyper-gradient. Our novel interpretation in turn leads to straightforward algorithm extensions for the F^{2}SA method. Observing that the forward difference used by F^{2}SA only has a first-order error guarantee, a natural idea to improve the error guarantee is to use higher-order finite difference methods. For instance, we know that the central difference has an improved second-order error guarantee. Based on this fact, we can derive the F^{2}SA-2 method that solves the following symmetric penalty problem:

$$\min_{\boldsymbol{x}\in\mathbb{R}^{d_x},\boldsymbol{y}\in\mathbb{R}^{d_y}} \frac{1}{2}\left(f(\boldsymbol{x},\boldsymbol{y}) + \lambda g(\boldsymbol{x},\boldsymbol{y}) - \min_{\boldsymbol{z}\in\mathbb{R}^{d_y}} \left(-f(\boldsymbol{x},\boldsymbol{z}) + \lambda g(\boldsymbol{x},\boldsymbol{z})\right)\right). \tag{4}$$

A similar approach has recently been discovered by Chayti and Jaggi [8] in the context of meta-learning, but they only show its empirical benefit without rigorous theoretical justifications. In this work, we show that F^{2}SA-2 returns an ϵ-estimation to $\nabla\varphi(\boldsymbol{x})$ under the setting $\lambda = \Omega(\epsilon^{-1/2})$ instead of $\Omega(\epsilon^{-1})$ in F^{2}SA, which further improves the SFO complexity of F^{2}SA from $\tilde{\mathcal{O}}(\epsilon^{-6})$ to $\tilde{\mathcal{O}}(\epsilon^{-5})$ for second-order smooth problems. Our idea is generalizable for any pth-order problems. We recall that in numerical analysis there exists the pth-order central difference that uses p

points to construct an estimator to the derivative of a unitary function with pth-order error guarantee (Lemma C.1). Motivated by this fact, we propose the F^{2}SA-p algorithm and show that it allows $\lambda = \Omega(\epsilon^{-1/p})$ for pth-order smooth problems, which further leads to the improved $\tilde{\mathcal{O}}(p\epsilon^{-4-2/p})$ SFO complexity for finding an ϵ-stationary point stated by our Theorem 3.1.

To examine the tightness of our upper bounds, we further extend the $\Omega(\epsilon^{-4})$ lower bound for SGD [3] from single-level optimization to bilevel optimization. Note that existing constructions for bilevel lower bound [14, 29] do not satisfy all our smoothness conditions in Definition 2.2. We demonstrate in Theorem F.1 that a fully separable construction for upper- and lower-level variables can immediately yield a valid $\Omega(\epsilon^{-4})$ lower bound for the problem class we study, showing that F^{2}SA-p is optimal up to logarithmic factors when $p = \Omega(\log \epsilon^{-1} / \log\log \epsilon^{-1})$. We summarize our main results, including both the lower and upper bounds, in Table 1 and discuss open problems in the following.

Method	Smoothness	Reference	Complexity
F^{2}SA	1st-order	[28]	$\tilde{\mathcal{O}}(\text{poly}(\kappa)\epsilon^{-7})$
F^{2}SA	1st-order	[29]	$\tilde{\mathcal{O}}(\text{poly}(\kappa)\epsilon^{-6})$
F^{2}SA	1st-order	[11]	$\tilde{\mathcal{O}}(\kappa^{12}\epsilon^{-6})$
F^{2}SA-p	1st-order + pth-order in $\boldsymbol{y}$	Theorem 3.1	$\tilde{\mathcal{O}}(p\kappa^{9+2/p}\epsilon^{-4-2/p})$
Lower Bound		Theorem F.1	$\Omega(\epsilon^{-4})$

Table 1: The SFO complexity of different methods to find an ϵ-stationary point for pth-order smooth first-order bilevel problems with condition number κ under standard SGD assumptions.

Notations. We use $\|\cdot\|$ to denote the Euclidean norm for vectors and the spectral norm for matrices and tensors. We use $\tilde{\mathcal{O}}(\,\cdot\,)$ and $\tilde{\Omega}(\,\cdot\,)$ to hide logarithmic factors in $\mathcal{O}(\,\cdot\,)$ and $\Omega(\,\cdot\,)$. We also use $h_1 \lesssim h_2$ to mean $h_1 = \mathcal{O}(h_2)$, $h_1 \gtrsim h_2$ to mean $h_1 = \Omega(h_2)$, and $h_1 \asymp h_2$ to mean that both $h_1 \lesssim h_2$ and $h_1 \gtrsim h_2$ hold. Additional notations for tensors are introduced in Appendix A.

2. Preliminaries

The goal of bilevel optimization is to minimize the hyper-objective $\varphi(\boldsymbol{x})$, which is in general nonconvex. Since finding a global minimizer of a general nonconvex function requires exponential complexity in the worst case [37, § 1.6], we follow the literature [6, 7] to consider the task of finding an approximate stationary point.

Definition 2.1 *Let $\varphi : \mathbb{R}^{d_x} \to \mathbb{R}$ be the hyper-objective defined in Eq. (1). We say $\boldsymbol{x} \in \mathbb{R}^{d_x}$ is an ϵ-hyper-stationary point if $\|\nabla \varphi(\boldsymbol{x})\| \le \epsilon$.*

Next, we introduce the assumptions used in this paper.

Assumption 2.1 *There exists stochastic gradient estimators $F(\boldsymbol{x},\boldsymbol{y})$ and $G(\boldsymbol{x},\boldsymbol{y})$ such that*

$$\mathbb{E}F(\boldsymbol{x},\boldsymbol{y};\xi)=\nabla f(\boldsymbol{x},\boldsymbol{y}),\quad \mathbb{E}\|F(\boldsymbol{x},\boldsymbol{y})-\nabla f(\boldsymbol{x},\boldsymbol{y})\|^2\le\sigma^2;$$
$$\mathbb{E}G(\boldsymbol{x},\boldsymbol{y};\zeta)=\nabla g(\boldsymbol{x},\boldsymbol{y}),\quad \mathbb{E}\|G(\boldsymbol{x},\boldsymbol{y})-\nabla g(\boldsymbol{x},\boldsymbol{y})\|^2\le\sigma^2,$$

where $\sigma>0$ is the variance of the stochastic gradient estimators. We also partition $F=(F_x,F_y)$ and $G=(G_x,G_y)$ such that F_x,F_y,G_x,G_y are estimators to $\nabla_x f,\nabla_y f,\nabla_x g,\nabla_y g$, respectively.

Assumption 2.2 *The hyper-objective defined in Eq. (1) is lower bounded, and we have*

$$\varphi(\boldsymbol{x}_0)-\inf_{\boldsymbol{x}\in\mathbb{R}^{d_x}}\varphi(\boldsymbol{x})\le\Delta,$$

where $\Delta>0$ is the initial suboptimality gap and we assume $\boldsymbol{x}_0=\mathbf{0}$ without loss of generality.

Assumption 2.3 *$g(\boldsymbol{x},\boldsymbol{y})$ is μ-strongly convex in $\boldsymbol{y}$, i.e., for any $\boldsymbol{y}_1,\boldsymbol{y}_2\in\mathbb{R}^{d_y}$, we have*

$$g(\boldsymbol{x},\boldsymbol{y}_2)\ge g(\boldsymbol{x},\boldsymbol{y}_1)+\langle\nabla_y g(\boldsymbol{x},\boldsymbol{y}_1),\boldsymbol{y}_2-\boldsymbol{y}_1\rangle+\frac{\mu}{2}\|\boldsymbol{y}_1-\boldsymbol{y}_2\|^2,$$

where $\mu>0$ is the strongly convex parameter.

Assumption 2.4 *For the upper-lower function f and lower-level function g, we assume that*

1. *$f(\boldsymbol{x},\boldsymbol{y})$ is L_0-Lipschitz in $\boldsymbol{y}$.*
2. *$\nabla f(\boldsymbol{x},\boldsymbol{y})$ and $\nabla g(\boldsymbol{x},\boldsymbol{y})$ are L_1-Lipschitz jointly in $(\boldsymbol{x},\boldsymbol{y})$.*
3. *$\nabla^2_{xy} g(\boldsymbol{x},\boldsymbol{y})$ and $\nabla^2_{yy} g(\boldsymbol{x},\boldsymbol{y})$ are L_2-Lipschitz jointly in $(\boldsymbol{x},\boldsymbol{y})$.*

We refer to the problem class that jointly satisfies all the above Assumption 2.1, 2.2, 2.3 and 2.4 as first-order smooth bilevel problems, for which [11, 29] showed the F^{2}SA method achieves the $\tilde{\mathcal{O}}(\epsilon^{-6})$ upper complexity bound. In this work, we show an improved bound under the following additional higher-order smoothness assumption on lower-level variable $\boldsymbol{y}$.

Assumption 2.5 (High order smoothness in y) *Given $p\in\mathbb{N}_+$, we assume that*

1. *$\frac{\partial^q}{\partial\boldsymbol{y}^q}\nabla f(\boldsymbol{x},\boldsymbol{y})$ is L_{q+1}-Lipschitz for all $q=1,\cdots,p-1$.*
2. *$\frac{\partial^{q+1}}{\partial\boldsymbol{y}^{q+1}}\nabla g(\boldsymbol{x},\boldsymbol{y})$ is L_{q+2}-Lipschitz in $\boldsymbol{y}$ for all $q=1,\cdots,p-1$.*

We refer to problems jointly satisfying all the above assumptions as pth-order smooth bilevel problems, and also formally define their condition numbers as follows.

Definition 2.2 (pth-order smooth bilevel problems) *Given $p\in\mathbb{N}_+$, $\Delta>0$, $L_0,L_1,\cdots,L_{p+1}>0$, and $\mu\le L_1$, we use $\mathcal{F}^{\text{nc-sc}}(L_0,\cdots,L_{p+1},\mu,\Delta)$ to denote the set of all bilevel instances satisfying Assumption 2.2, 2.3, 2.4 and 2.5. For this problem class, we define the largest smoothness constant $\bar{L}=\max_{0\le j\le p}L_j$ and condition number $\kappa=\bar{L}/\mu$.*

There are also other prior works demonstrating that additional assumptions can lead to acceleration in bilevel optimization. We compare these works in Appendix B.

Algorithm 1 F^{2}SA-p $(\boldsymbol{x}_0, \boldsymbol{y}_0)$, even p

```
$\boldsymbol{y}_0^j = \boldsymbol{y}_0, \ \forall j \in \mathbb{N}$
for $t = 0, 1, \cdots, T-1$
    Sample random i.i.d indexes $\{(\xi_j^y, \zeta_j^y)\}_{j=-p/2}^{p/2}$ and $\{(\xi_i^x, \zeta_i^x)\}_{i=1}^{S}$.
    for $j = -p/2, -p/2+1, \cdots, p/2$
        $\boldsymbol{y}_t^{j,0} = \boldsymbol{y}_t^j$
        for $k = 0, 1, \cdots, K-1$
            $\boldsymbol{y}_t^{j,k+1} = \boldsymbol{y}_t^{j,k} - \eta_y \left( j\nu F_y(\boldsymbol{x}_t, \boldsymbol{y}_t^{j,k}; \xi_j^y) + G_y(\boldsymbol{x}_t, \boldsymbol{y}_t^{j,k}; \zeta_j^y) \right)$
        end for
        $\boldsymbol{y}_{t+1}^j = \boldsymbol{y}_t^{j,K}$
    end for
    Let $\{\alpha_j\}_{j=-p/2}^{p/2}$ be the $p$th-order central coefficients defined in Lemma C.1.
    $\Phi_t = \frac{1}{S}\sum_{i=1}^{S}\sum_{j=-p/2}^{p/2} \alpha_j \left( jF_x(\boldsymbol{x}_t, \boldsymbol{y}_{t+1}^j; \xi_i^x) + \frac{G_x(\boldsymbol{x}_t, \boldsymbol{y}_{t+1}^j; \zeta_i^x)}{\nu} \right)$
    $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta_x \Phi_t / \|\Phi_t\|$
end for
```

3. The F^{2}SA-p Method

Let $\ell_\nu(\boldsymbol{x}) = \min_{\boldsymbol{y} \in \mathbb{R}^{d_y}} \{ g_\nu(\boldsymbol{x}, \boldsymbol{y}) := \nu f(\boldsymbol{x}, \boldsymbol{y}) + g(\boldsymbol{x}, \boldsymbol{y}) \}$ and $\boldsymbol{y}_\nu^*(\boldsymbol{x}) = \arg\min_{\boldsymbol{y} \in \mathbb{R}^{d_y}} g_\nu(\boldsymbol{x}, \boldsymbol{y})$. Under this notation, we can interpret [28, Lemma 3.1] as stating that the partial derivatives with respect to $\boldsymbol{x}$ and ν are commutable, *i.e.*, $\frac{\partial^2}{\partial \nu \partial \boldsymbol{x}} \ell_\nu(\boldsymbol{x})|_{\nu=0} = \frac{\partial^2}{\partial \boldsymbol{x} \partial \nu} \ell_\nu(\boldsymbol{x})|_{\nu=0} = \nabla \varphi(\boldsymbol{x})$. Now, let $\nu = 1/\lambda$ in Eq. (3). Then we can observe that the F^{2}SA method [11, 28] is exactly using forward difference to approximate $\nabla \varphi(\boldsymbol{x})$, *i.e.*, $\frac{\frac{\partial}{\partial \boldsymbol{x}} \ell_\nu(\boldsymbol{x}) - \frac{\partial}{\partial \boldsymbol{x}} \ell_0(\boldsymbol{x})}{\nu} \approx \nabla \varphi(\boldsymbol{x})$. However, the forward difference is not the only way to approximate a derivative. Essentially, it falls into a general class of pth-order finite difference [4] that can guarantee an $\mathcal{O}(\nu^p)$ approximation error (Lemma C.1). Motivated by this fact, we propose a method called F^{2}SA-p that applies the pth-order finite difference for hyper-gradient approximation and also analyze its theoretical guarantee.

Due to space limitations, we only present Algorithm 1 designed for even p in the main text. The algorithm for odd p can be designed similarly, and we defer the concrete algorithm to Appendix E. Our Algorithm 1 follows the double-loop structure of F^{2}SA [11, 29] and modifies the hyper-gradient estimator according to the pth-order finite difference (Lemma C.1). In the following, we give a detailed introduction to the procedures of the two loops of F^{2}SA-p:

1. In the outer loop, the algorithm first samples a mini-batch with size S and uses Lemma C.1 to construct Φ_t via the linear combination of $\frac{\partial}{\partial \boldsymbol{x}} \ell_{j\nu}(\boldsymbol{x}_t)$ for $j = -p/2, \cdots, p/2$ every iteration. After obtaining Φ_t as an approximation to $\nabla \varphi(\boldsymbol{x}_t)$, the algorithm then performs a normalized gradient descent step $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta_x \Phi_t / \|\Phi_t\|$ with total T iterations.

2. The inner loop returns an approximation to $\frac{\partial}{\partial \boldsymbol{x}} \ell_{j\nu}(\boldsymbol{x}_t)$ for all $j = -p/2, \cdots, p/2$. Note that Danskin's theorem indicates $\frac{\partial}{\partial \boldsymbol{x}} \ell_{j\nu}(\boldsymbol{x}_t) = \frac{\partial}{\partial \boldsymbol{x}} g_{j\nu}(\boldsymbol{x}_t, \boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_t))$. It suffices to approximate $\boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_t)$ to sufficient accuracy, which is achieved by taking a K-step single-batch SGD subroutine with stepsize η_y on each function $g_{j\nu}(\boldsymbol{x}, \cdot)$.

For the pth-order finite difference to have an improved error guarantee, we require $\frac{\partial}{\partial \boldsymbol{x}}\ell_\nu(\boldsymbol{x})$ being pth-order Lipschitz continuous in ν. We formally show it in the following lemma.

Lemma 3.1 *Let $\nu \in (0, 1/(2\kappa)]$. For any instance in the pth-order smooth bilevel problem class $\mathcal{F}^{nc\text{-}sc}(L_0, \cdots, L_{p+1}, \mu, \Delta)$ in Definition 2.2, $\frac{\partial^{p+1}}{\partial \nu^p \partial \boldsymbol{x}}\ell_\nu(\boldsymbol{x})$ is $\mathcal{O}(\kappa^{2p+1}\bar{L})$-Lipschitz continuous in ν.*

Our result generalizes the prior result for $p = 1$ [28] to any $p \in \mathbb{N}_+$ and also tightens the prior bounds for $p = 2$ [11] as we discuss in Remark D.1.

Theorem 3.1 (Main theorem) *For any instance in the pth-order smooth bilevel problem class $\mathcal{F}^{nc\text{-}sc}(L_0, \cdots, L_{p+1}, \mu, \Delta)$ as per Definition 2.2, set the hyper-parameters as*

$$\begin{gathered} \nu \asymp \min\left\{\frac{R}{\kappa}, \left(\frac{\epsilon}{\bar{L}\kappa^{2p+1}}\right)^{1/p}\right\}, \quad \eta_x \asymp \frac{\epsilon}{L_1\kappa^3}, \quad \eta_y \asymp \frac{\nu^2\epsilon^2}{L_1\kappa\sigma^2}, \\ S \asymp \frac{\sigma^2}{\nu^2\epsilon^2}, \quad K \asymp \frac{\kappa^2\sigma^2}{\nu^2\epsilon^2}\log\left(\frac{RL_1\kappa}{\nu\epsilon}\right), \quad T \asymp \frac{\Delta}{\eta_x\epsilon}, \end{gathered} \tag{5}$$

where $R = \|\boldsymbol{y}_0 - \boldsymbol{y}^*(\boldsymbol{x}_0)\|$. Run Algorithm 1 if p is even or Algorithm 2 if p is odd. Then we can provably find an ϵ-stationary point of $\varphi(\boldsymbol{x})$ with the total SFO calls upper bounded by

$$pT(S+K) = \mathcal{O}\left(\frac{p\Delta L_1 \bar{L}^{2/p}\sigma^2\kappa^{9+2/p}}{\epsilon^{4+2/p}}\log\left(\frac{RL_1\bar{L}\kappa}{\epsilon}\right)\right).$$

The above theorem shows that the F^{2}SA-p method can achieve the $\tilde{\mathcal{O}}(p\kappa^{9+2/p}\epsilon^{-4-2/p}\log(\kappa/\epsilon))$ SFO complexity for pth-order smooth bilevel problems. In the following, we give several remarks on the complexity in different regions of p.

Remark 3.1 (First-order smooth region) *Under the case of $p = 1$, our upper bound becomes $\tilde{\mathcal{O}}(\kappa^{11}\epsilon^{-6})$, which improves the $\tilde{\mathcal{O}}(\kappa^{12}\epsilon^{-6})$ bound in [11] by a factor of κ. The improvement comes from a tighter analysis in the lower-level SGD update and a careful parameter setting.*

Remark 3.2 (Highly smooth region) *Under the case of $p = \Omega(\log(\kappa/\epsilon)/\log\log(\kappa/\epsilon))$, the complexity of $\mathcal{O}(p\kappa^9\epsilon^{-4}(\kappa/\epsilon)^{2/p}\log(\kappa/\epsilon))$ in Theorem 3.1 can simplify to $\mathcal{O}(\kappa^9\epsilon^{-4}\log^3(\kappa/\epsilon)/\log\log(\kappa/\epsilon))$ $= \tilde{\mathcal{O}}(\kappa^9\epsilon^{-4})$, which matches the best-known complexity for Hessian-vector-product-based methods under stochastic Hessian assumption (6) established by Ji et al. [22]. In Theorem F.1, we show an $\Omega(\epsilon^{-4})$ lower bound via a reduction to single-level minimization problems [3], demonstrating that the upper bound of F^{2}SA-p in this region is nearly optimal when κ is a constant.*

4. Conclusions and Future Works

This paper proposes a class of fully first-order method F^{2}SA-p that achieves the $\tilde{\mathcal{O}}(p\epsilon^{-4-2/p})$ SFO complexity for pth-order smooth bilevel problems. Our result generalized the best-known $\tilde{\mathcal{O}}(\epsilon^{-6})$ result [11, 29] from $p = 1$ to any $p \in \mathbb{N}_+$. We also complement our result with an $\Omega(\epsilon^{-4})$ lower bound to show that our method is near-optimal when $p = \Omega(\log\epsilon^{-1}/\log\log\epsilon^{-1})$. Nevertheless, a gap still exists when p is small, and we still do not know how to fill it even for the basic setting $p = 1$. Another possible direction is to extend our theory to structured nonconvex-nonconvex bilevel problems studied by many recent works [9, 10, 23, 30, 42, 43].

References

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

[2] 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.

[3] 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):165–214, 2023.

[4] Kendall Atkinson and Weimin Han. Finite difference method. *Theoretical Numerical Analysis: A Functional Analysis Framework*, pages 249–271, 2005.

[5] Fan Bao, Guoqiang Wu, Chongxuan Li, Jun Zhu, and Bo Zhang. Stability and generalization of bilevel programming in hyperparameter optimization. In *NeurIPS*, 2021.

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

[7] 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.

[8] El Mahdi Chayti and Martin Jaggi. A new first-order meta-learning algorithm with convergence guarantees. *arXiv preprint arXiv:2409.03682*, 2024.

[9] He Chen, Jiajin Li, and Anthony Man-cho So. Set smoothness unlocks clarke hyper-stationarity in bilevel optimization. In *NeurIPS*, 2025.

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

[11] Lesi Chen, Yaohua Ma, and Jingzhao Zhang. Near-optimal nonconvex-strongly-convex bilevel optimization with fully first-order oracles. *JMLR*, 2025.

[12] Ashok Cutkosky and Harsh Mehta. Momentum improves normalized SGD. In *ICML*, 2020.

[13] Ashok Cutkosky and Francesco Orabona. Momentum-based variance reduction in non-convex SGD. In *NeurIPS*, 2019.

[14] Mathieu Dagréou, Thomas Moreau, Samuel Vaiter, and Pierre Ablin. A lower bound and a near-optimal algorithm for bilevel empirical risk minimization. In *AISTATS*, 2024.

[15] Aaron Defazio and Léon Bottou. On the ineffectiveness of variance reduced optimization for deep learning. In *NeurIPS*, 2019.

[16] Cong Fang, Chris Junchi Li, Zhouchen Lin, and Tong Zhang. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. In *NeurIPS*, 2018.

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

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

[19] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

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

[21] Minhui Huang, Xuxing Chen, Kaiyi Ji, Shiqian Ma, and Lifeng Lai. Efficiently escaping saddle points in bilevel optimization. *JMLR*, 26(1):1–61, 2025.

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

[23] Liuyuan Jiang, Quan Xiao, Lisha Chen, and Tianyi Chen. Beyond value functions: Single-loop bilevel optimization under flatness conditions. *arXiv preprint arXiv:2507.20400*, 2025.

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

[25] Chi Jin, Praneeth Netrapalli, Rong Ge, Sham M. Kakade, and Michael I. Jordan. On nonconvex optimization for machine learning: Gradients, stochasticity, and saddle points. *Journal of the ACM (JACM)*, 68(2):1–29, 2021.

[26] 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.

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

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

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

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

[31] 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.

[32] Martin W Licht. Higher-order chain rules for tensor fields, generalized bell polynomials, and estimates in orlicz-sobolev-slobodeckij and total variation spaces. *Journal of Mathematical Analysis and Applications*, 534(1):128005, 2024.

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

[34] Risheng Liu, Xuan Liu, Shangzhi Zeng, Jin Zhang, and Yixuan Zhang. Value-function-based sequential minimization for bi-level optimization. *TPAMI*, 45(12):15930–15948, 2023.

[35] Luo Luo, Yujun Li, and Cheng Chen. Finding second-order stationary points in nonconvex-strongly-concave minimax optimization. In *NeurIPS*, 2022.

[36] Matthew Mackay, Paul Vicol, Jonathan Lorraine, David Duvenaud, and Roger Grosse. Self-tuning networks: Bilevel optimization of hyperparameters using structured best-response functions. In *ICLR*, 2019.

[37] Arkadij Semenovič Nemirovskij and David Borisovich Yudin. Problem complexity and method efficiency in optimization. 1983.

[38] 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.

[39] Aravind Rajeswaran, Chelsea Finn, Sham M. Kakade, and Sergey Levine. Meta-learning with implicit gradients. In *NeurIPS*, volume 32, 2019.

[40] Han Shen, Pin-Yu Chen, Payel Das, and Tianyi Chen. Seal: Safety-enhanced aligned llm fine-tuning via bilevel data selection. In *ICLR*, 2025.

[41] Han Shen, Quan Xiao, and Tianyi Chen. On penalty-based bilevel gradient descent method. *Mathematical Programming*, pages 1–51, 2025.

[42] Quan Xiao and Tianyi Chen. Unlocking global optimality in bilevel optimization: A pilot study. In *ICLR*, 2025.

[43] Quan Xiao, Songtao Lu, and Tianyi Chen. An generalized alternating optimization method for bilevel problems under the polyak-łojasiewicz condition. In *NeurIPS*, 2023.

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

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

[46] Junjie Yang, Kaiyi Ji, and Yingbin Liang. Provably faster algorithms for bilevel optimization. In *NeurIPS*, 2021.

[47] Yifan Yang, Peiyao Xiao, and Kaiyi Ji. Achieving $\mathcal{O}(\epsilon^{-1.5})$ complexity in hessian/jacobian-free stochastic bilevel optimization. In *NeurIPS*, 2023.

Appendix A. Notations for Tensors

We follow the notation of tensors used by Kolda and Bader [27]. For two p-way tensors $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_p}$ and $\mathcal{Y} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_p}$, their inner product $z = \langle \mathcal{X}, \mathcal{Y} \rangle$ is defined as

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \cdots \sum_{i_q=1}^{n_p} \mathcal{X}_{i_1,i_2,\cdots,i_p} \mathcal{Y}_{i_1,i_2,\cdots,i_p}.$$

For two tensors $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots n_p}$ and $\mathcal{Y} \in \mathbb{R}^{m_1 \times m_2 \cdots \times m_q}$, their outer product $\mathcal{Z} = \mathcal{X} \otimes \mathcal{Y}$ is a tensor $\mathcal{Z} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_p \times m_1 \times m_2 \times \cdots \times m_q}$ whose elements are defined as

$$(\mathcal{X} \otimes \mathcal{Y})_{i_1,i_2,\cdots,i_p,j_1,j_2,\cdots,j_q} = \mathcal{X}_{i_1,i_2,\cdots,i_p} \mathcal{Y}_{i_1,i_2,\cdots,i_p}.$$

The operator norm of a tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_p}$ is defined as

$$\|\mathcal{X}\| = \sup_{\|\boldsymbol{u}_i\|=1, i=1,\cdots,p} \langle \mathcal{X}, \boldsymbol{u}_1 \otimes \boldsymbol{u}_2 \otimes \cdots \otimes \boldsymbol{u}_p \rangle.$$

Equipped with the notion of norm, we say a mapping $\mathcal{T} : \mathbb{R} \to \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_p}$ is D-bounded if

$$\|\mathcal{T}(\boldsymbol{x})\| \leq D, \quad \forall \boldsymbol{x} \in \mathbb{R}.$$

We say $\mathcal{T}$ is C-Lipschitz continuous if

$$\|\mathcal{T}(\boldsymbol{x}) - \mathcal{T}(\boldsymbol{y})\| \leq C\|\boldsymbol{x} - \boldsymbol{y}\|, \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}.$$

Appendix B. Comparison to Previous Works

All our above assumptions align with [11] except for the additional Assumption 2.5. Since we are not the first work to demonstrate that additional assumptions can lead to acceleration in bilevel optimization, we first give a detailed discussion on other assumptions made in related works to see our differences before we show our improved upper bound.

Stochastic Hessian assumption. Ghadimi and Wang [18], Ji et al. [22] assumes the access to a stochastic Hessian estimator $\boldsymbol{H}(\boldsymbol{x}, \boldsymbol{y})$ such that

$$\mathbb{E}\boldsymbol{H}(\boldsymbol{x}, \boldsymbol{y}) = \nabla^2 g(\boldsymbol{x}, \boldsymbol{y}), \quad \mathbb{E}\|\boldsymbol{H}(\boldsymbol{x}, \boldsymbol{y}) - \nabla^2 g(\boldsymbol{x}, \boldsymbol{y})\| \leq \sigma^2. \tag{6}$$

Under this assumption, in conjunction with Assumption 2.2, 2.3, and 2.4, Ghadimi and Wang [18] proposed the BSA method and showed that it can find an ϵ stationary point of $\varphi(\boldsymbol{x})$ with $\tilde{\mathcal{O}}(\epsilon^{-6})$ stochastic gradient oracles and $\tilde{\mathcal{O}}(\epsilon^{-4})$ stochastic HVP oracles. Later, Ji et al. [22] proposed the stocBiO method which only requires $\tilde{\mathcal{O}}(\epsilon^{-4})$ stochastic gradient and HVP oracles. Compared to them, we consider the setting where the algorithms only have access to stochastic gradient estimators, and we make no assumptions on the stochastic Hessians.

Mean-squared smoothness assumption. Besides Assumption 2.1, 2.2, 2.3, 2.4 and the stochastic Hessian assumption (6), Khanduri et al. [26], Yang et al. [46, 47] further assumes that the stochastic estimators to gradients and Hessians are mean-squared smooth:

$$\begin{aligned}\mathbb{E}\|F(\boldsymbol{x},\boldsymbol{y}) - F(\boldsymbol{x}',\boldsymbol{y}')\|^2 &\le \bar{L}_1^2\|(\boldsymbol{x},\boldsymbol{y}) - (\boldsymbol{x}',\boldsymbol{y}')\|^2, \\ \mathbb{E}\|G(\boldsymbol{x},\boldsymbol{y}) - G(\boldsymbol{x}',\boldsymbol{y}')\|^2 &\le \bar{L}_1^2\|(\boldsymbol{x},\boldsymbol{y}) - (\boldsymbol{x}',\boldsymbol{y}')\|^2, \\ \mathbb{E}\|\boldsymbol{H}(\boldsymbol{x},\boldsymbol{y}) - \boldsymbol{H}(\boldsymbol{x}',\boldsymbol{y}')\|^2 &\le \bar{L}_2^2\|(\boldsymbol{x},\boldsymbol{y}) - (\boldsymbol{x}',\boldsymbol{y}')\|^2.\end{aligned} \tag{7}$$

Under this additional assumption, they proposed faster stochastic methods with upper complexity bound of $\tilde{\mathcal{O}}(\epsilon^{-3})$ via variance reduction [13, 16]. However, variance reduction are typically ineffective in practice [15] since the mean-squared smoothness constants $\bar{L}_1$ and $\bar{L}_2$ can be arbitrarily worse than the smoothness constants L_1 and L_2. In this paper, we only consider the setting without mean-squared smoothness assumptions and study a different acceleration mechanism from variance reduction.

Jointly high-order smoothness assumption. Huang et al. [21] introduced a second-order smoothness assumption similar to but stronger than Assumption 2.5 when $p = 2$. Specifically, they assumed the second-order smoothness jointly in $(\boldsymbol{x},\boldsymbol{y})$ instead of $\boldsymbol{y}$ only:

$$\begin{aligned}&\nabla^2 f(\boldsymbol{x},\boldsymbol{y}) \text{ is } L_2\text{-Lipschitz jointly in } (\boldsymbol{x},\boldsymbol{y}); \\ &\nabla^3 g(\boldsymbol{x},\boldsymbol{y}) \text{ is } L_3\text{-Lipschitz jointly in } (\boldsymbol{x},\boldsymbol{y}).\end{aligned} \tag{8}$$

The jointly second-order smoothness (8) ensures that the hyper-objective $\varphi(\boldsymbol{x})$ has Lipschitz continuous Hessians, which further allows the application of known techniques in minimizing second-order smooth objectives. Huang et al. [21] applied the technique from [1, 24, 25, 44] to show that an HVP-based method can find a second-order stationary point in $\tilde{\mathcal{O}}(\epsilon^{-2})$ complexity under the deterministic setting, and in $\tilde{\mathcal{O}}(\epsilon^{-4})$ under the stochastic Hessian assumption (6). Yang et al. [45] applied the technique from [31] to accelerate the complexity HVP-based method to $\tilde{\mathcal{O}}(\epsilon^{-1.75})$ in the deterministic setting. Chen et al. [11] also proposed a fully first-order method to achieve the same $\tilde{\mathcal{O}}(\epsilon^{-1.75})$ complexity. Compared to these works, our work demonstrates a unique acceleration mechanism in stochastic bilevel optimization that only comes from the high-order smoothness in $\boldsymbol{y}$.

Appendix C. Hyper-Gradient Approximation via Finite Difference

Recall our notations that

$$\begin{aligned} g_\nu(\boldsymbol{x},\boldsymbol{y}) &:= \nu f(\boldsymbol{x},\boldsymbol{y}) + g(\boldsymbol{x},\boldsymbol{y}), \\ \boldsymbol{y}_\nu^*(\boldsymbol{x}) &:= \arg\min_{\boldsymbol{y}\in\mathbb{R}^{d_y}} g_\nu(\boldsymbol{x},\boldsymbol{y}), \\ \ell_\nu(\boldsymbol{x}) &:= \min_{\boldsymbol{y}\in\mathbb{R}^{d_y}} g_\nu(\boldsymbol{x},\boldsymbol{y}),\end{aligned}$$

where g_ν is the perturbed lower-lever problem with $\boldsymbol{y}_\nu^*(\boldsymbol{x})$ and $\ell_\nu(\boldsymbol{x})$ being its optimal solution and optimal value, respectively. Since the constraint $\boldsymbol{y} = \arg\min_{\boldsymbol{z}\in\mathbb{R}^{d_z}} g(\boldsymbol{x},\boldsymbol{z})$ is equivalent to requiring $g(\boldsymbol{x},\boldsymbol{y}) \le \min_{\boldsymbol{z}\in\mathbb{R}^{d_y}} g(\boldsymbol{x},\boldsymbol{z})$, it can be shown [11, Lemma B.3] that $\frac{\partial}{\partial\nu}\ell_\nu(\boldsymbol{x})|_{\nu=0} =$

$\varphi(\boldsymbol{x})$ holds under Assumption 2.3 and 2.4. Furthermore, Kwon et al. [28] showed that the partial derivatives with respect to $\boldsymbol{x}$ and ν are commutable, which leads to

$$\frac{\partial^2}{\partial \nu \partial \boldsymbol{x}} \ell_\nu(\boldsymbol{x})|_{\nu=0} = \frac{\partial^2}{\partial \boldsymbol{x} \partial \nu} \ell_\nu(\boldsymbol{x})|_{\nu=0} = \nabla \varphi(\boldsymbol{x}). \tag{9}$$

Let $\nu = 1/\lambda$ in Eq. (3). Then the fully first-order hyper-gradient estimator [11, 28] is exactly using forward difference to approximate $\nabla \varphi(\boldsymbol{x})$, that is,

$$\frac{\frac{\partial}{\partial \boldsymbol{x}} \ell_\nu(\boldsymbol{x}) - \frac{\partial}{\partial \boldsymbol{x}} \ell_0(\boldsymbol{x})}{\nu} \approx \frac{\partial^2}{\partial \nu \partial \boldsymbol{x}} \ell_\nu(\boldsymbol{x})|_{\nu=0} = \nabla \varphi(\boldsymbol{x}). \tag{10}$$

However, the forward difference is not the only way to approximate a derivative. Essentially, it falls into a general class of pth-order finite difference [4] that can guarantee an $\mathcal{O}(\nu^p)$ approximation error. We restate this known result in the following lemma and also provide a self-contained proof for completeness.

Lemma C.1 *Assume the unitary function $\psi : \mathbb{R} \to \mathbb{R}$ has C-Lipschitz continuous pth-order derivative. If p is even, there exist pth-order central difference coefficients $\{\alpha_j\}_{j=-p/2}^{p/2}$ such that*

$$\left| \frac{1}{\nu} \sum_{j=-p/2}^{p/2} \alpha_j \psi(j\nu) - \psi'(0) \right| = \mathcal{O}(C\nu^p),$$

where $\alpha_0 = 0$ and $\alpha_j = \alpha_{-j}$ for all $j = 1, \cdots, p/2$. If p is odd, there exist pth-order forward difference coefficients $\{\beta_j\}_{j=0}^{p}$ such that

$$\left| \frac{1}{\nu} \sum_{j=-p/2}^{p/2} \beta_j \psi(j\nu) - \psi'(0) \right| = \mathcal{O}(C\nu^p),$$

Proof If $\psi^{(p)}(\nu)$ is C-Lipschitz continuous in ν, then by Taylor's theorem we have

$$\psi(\nu) = \psi(0) + \sum_{k=1}^{p} \frac{(j\nu)^k}{k!} \psi^{(k)}(0) + \mathcal{O}\left(C\nu^{p+1}\right). \tag{11}$$

If p is even, we choose the generalized central difference. If p is odd, we choose the generalized forward difference. Our choices underpin the following proof. Below, we analyze the case when p is even or odd separately.

If p is even. For the coefficients $\{\alpha_j\}_{j=-p/2}^{p/2}$, we set

$$\alpha_j = \alpha_{-j}, \quad \forall j = 0, 1, \cdots, p/2.$$

Then, summing up Eq. (11) with coefficients α_j gives

$$\frac{1}{\nu} \sum_{j=-p/2}^{j=p/2} \alpha_j \psi(j\nu) = \underbrace{2 \sum_{j=1}^{p/2} \alpha_j \sum_{k=1,3,\cdots}^{p/2-1} \frac{j^k \nu^{k-1}}{k!} \psi^{(k)}(0)}_{(*)} + \mathcal{O}\left(C\nu^p\right).$$

To let term (*) be equivalent to $\psi'(0)$, we let $\{\alpha_j\}_{j=1}^{p/2}$ satisfy the following equations:

$$2\sum_{j=1}^{p/2} \alpha_j j^k = \mathbf{1}_{k=1}, \quad \forall k = 1, 3, \cdots, p/2-1,$$

which is equivalent to let $\{j\alpha_j\}_{j=1}^{p/2}$ satisfy the following linear equation

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1^2 & 2^2 & 3^2 & \cdots & (p/2)^2 \\ 1^4 & 2^4 & 3^4 & \cdots & (p/2)^4 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1^{p/2-2} & 2^{p/2-2} & 3^{p/2-2} & \cdots & (p/2)^{p/2-2} \end{pmatrix} \begin{pmatrix} \alpha_1 \\ 2\alpha_2 \\ 3\alpha_3 \\ \vdots \\ (p/2)\alpha_{p/2} \end{pmatrix} = \begin{pmatrix} 1/2 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Since the coefficient matrix in the above linear equation is a Vandermonde matrix, we know this equation has a unique solution, which gives the value of $\{\alpha_j\}_{j=1}^{p/2}$.

If p is odd. For the coefficients $\{\beta_j\}_{j=0}^{p}$, we first let them satisfy the constraint $\sum_{j=0}^{p} \beta_j = 0$. Then, summing up Eq. (11) with coefficients β_j gives

$$\frac{1}{\nu}\sum_{j=0}^{j=p} \beta_j \psi(j\nu) = \underbrace{\sum_{j=0}^{p} \beta_j \sum_{k=1}^{p} \frac{j^k \nu^{k-1}}{k!} \psi^{(k)}(0)}_{(*)} + \mathcal{O}\left(C\nu^p\right).$$

To let term (*) be equivalent to $\psi'(0)$, we let $\{\beta_j\}_{j=0}^{p}$ satisfy the following equations:

$$\sum_{j=1}^{p} \beta_j j^k = \mathbf{1}_{k=1}, \quad \forall k = 1, 2, \cdots, p,$$

which is equivalent to let $\{j\beta\}_{j=1}^{p}$ satisfy the following linear equation

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 3 & \cdots & p \\ 1^2 & 2^2 & 3^2 & \cdots & p^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1^{p-1} & 2^{p-1} & 3^{p-1} & \cdots & p^{p-1} \end{pmatrix} \begin{pmatrix} \beta_1 \\ 2\beta_2 \\ 3\beta_3 \\ \vdots \\ p\beta_p \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

As before, the coefficient matrix in the above linear equation is also a Vandermonde matrix. Therefore, we know this equation has a unique solution, which gives the value of $\{\beta_j\}_{j=1}^{p}$ and the coefficient β_0 can be calculated by $\beta_0 = 1 - \sum_{j=1}^{p} \beta_j$. ■

When $p = 1$, we have $\beta_0 = -1$, $\beta_1 = 1$, and we obtain the forward difference estimator $\psi(\nu) - \psi(0)/\nu$; When $p = 2$ we have $\alpha_{-1} = -1/2$, $\alpha_1 = 1/2$ and we obtain the central difference estimator $(\psi(\nu) - \psi(-\nu))/(2\nu)$. Lemma C.1 tells us that in general we can always construct a finite difference estimator $\mathcal{O}(\nu^p)$ error with p points with for even p or $p+1$ points for odd p under the given smoothness conditions. This leads to the hyper-gradient estimator used in Algorithm 1.

Remark C.1 *A subtlety to use Lemma C.1 for hyper-gradient estimation is that it only applies to a unitary function while $\frac{\partial}{\partial \boldsymbol{x}}\ell_\nu(\boldsymbol{x})$ is a vector-valued function in ν. However, the approximation error still holds for the whole vector under the Euclidean norm if we apply the lemma on each dimension and note that the finite difference coefficients are the same for all dimensions.*

Appendix D. Proof of Lemma 3.1

The proof relies on the high-dimensional version of the Faà di Bruno formula. To formally state the result, we define the following notions. For a mapping $\mathcal{T}:\mathbb{R}^m \to \mathbb{R}^{n_1\times\cdots\times n_q}$, we define its kth-order directional derivative evaluated at $\boldsymbol{z}\in\mathbb{R}^m$ along the direction $(\boldsymbol{u}_1,\cdots,\boldsymbol{u}_k)$ as

$$\nabla^k_{\boldsymbol{u}_1,\cdots,\boldsymbol{u}_k}\mathcal{T}_{|\boldsymbol{z}} = \nabla^k\mathcal{T}_{|\boldsymbol{z}}(\boldsymbol{u}_1,\cdots,\boldsymbol{u}_k).$$

We let the symmetric products of $\boldsymbol{u}_1,\cdots,\boldsymbol{u}_k$ as

$$\boldsymbol{u}_1\vee\boldsymbol{u}_2\vee\cdots\vee\boldsymbol{u}_k = \frac{1}{k!}\sum_{\pi\in\mathrm{Perm}(k)}\boldsymbol{u}_{\pi(1)}\otimes\boldsymbol{u}_{\pi(2)}\otimes\cdots\otimes\boldsymbol{u}_{\pi(k)},$$

where $\mathrm{Perm(k)}$ denotes the set of permutations of $\{1,2,\cdots,k\}$. Also, we define the set of all (unordered) partitions of a set A into k pairwise disjoint non-empty sets as

$$\mathcal{P}(A,k) = \left\{\boldsymbol{P}=(P_1,\cdots,P_k)\subseteq\mathcal{B}(A)\mid A=\cup_{j=1}^k P_j;\ \emptyset\notin\boldsymbol{P};\ P_i\cap P_j=\emptyset,\forall i<j\right\},$$

where $\mathcal{B}(A)$ is the power set of A, *i.e.*, the set of all subsets of A. We also abbreviate $\mathcal{P}(\{1:q\},k)$ as $\mathcal{P}(q,k)$. Using the above notions, we have the following result.

Lemma D.1 ([32, Proposition 3.1]) *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two mappings. If $\mathcal{T}_1$ and $\mathcal{T}_2$ are k-times differentiable at the point $\boldsymbol{z}$ and $\mathcal{T}_1(\boldsymbol{z})$, respectively, then the composite mapping $\mathcal{T}_2\circ\mathcal{T}_1$ is k-times differentiable at the point $\boldsymbol{z}$ and we have*

$$\nabla^q(\mathcal{T}_2\circ\mathcal{T}_1)_{|\boldsymbol{z}}(\vee_{i=1}^q\boldsymbol{u}_i) = \sum_{\substack{1\le k\le q,\\ \boldsymbol{P}\in\mathcal{P}(q,k)}}\nabla^k\mathcal{T}_{2|\mathcal{T}_1(\boldsymbol{z})}\left(\nabla^{|P_1|}\mathcal{T}_{1|\boldsymbol{z}}(\vee_{i\in P_1}\boldsymbol{u}_i),\cdots\nabla^{|P_k|}\mathcal{T}_{1|\boldsymbol{z}}(\vee_{i\in P_k}\boldsymbol{u}_i)\right).$$

Recall Danskin's theorem that $\frac{\partial}{\partial\boldsymbol{x}}\ell_\nu(\boldsymbol{x}) = \frac{\partial}{\partial\boldsymbol{x}}g_\nu(\boldsymbol{x},\boldsymbol{y}^*_\nu(\boldsymbol{x}))$. We can apply Lemma D.1 with $\mathcal{T}_1=\boldsymbol{y}^*_\nu(\boldsymbol{x})$ and $\mathcal{T}_1=\frac{\partial}{\partial\boldsymbol{x}}g_\nu(\boldsymbol{x},\boldsymbol{y})$ to obtain that

$$\frac{\partial^{q+1}}{\partial\nu^q\partial\boldsymbol{x}}\ell_\nu(\boldsymbol{x}) = \sum_{\substack{1\le k\le q,\\ \boldsymbol{P}\in\mathcal{P}(q,k)}}\frac{\partial^{k+1}}{\partial\boldsymbol{y}^k\partial\boldsymbol{x}}g_\nu(\boldsymbol{x},\boldsymbol{y}^*_\nu(\boldsymbol{x}))\left(\frac{\partial^{|P_1|}}{\partial\nu^{|P_1|}}\boldsymbol{y}^*_\nu(\boldsymbol{x}),\cdots,\frac{\partial^{|P_k|}}{\partial\nu^{|P_k|}}\boldsymbol{y}^*_\nu(\boldsymbol{x})\right). \tag{12}$$

Symmetrically, using the first-order optimality condition $\frac{\partial}{\partial\boldsymbol{y}}g_\nu(\boldsymbol{x},\boldsymbol{y}^*_\nu(\boldsymbol{x}))=0$ and where the first identity uses the Lemma D.1 with $\mathcal{T}_1=\boldsymbol{y}^*_\nu(\boldsymbol{x})$ and $\mathcal{T}_1=\frac{\partial}{\partial\boldsymbol{y}}g_\nu(\boldsymbol{x},\boldsymbol{y})$ yields that

$$0 = \sum_{\substack{1\le k\le q,\\ \boldsymbol{P}\in\mathcal{P}(q,k)}}\frac{\partial^{k+1}}{\partial\boldsymbol{y}^{k+1}}g_\nu(\boldsymbol{x},\boldsymbol{y}^*_\nu(\boldsymbol{x}))\left(\frac{\partial^{|P_1|}}{\partial\nu^{|P_1|}}\boldsymbol{y}^*_\nu(\boldsymbol{x}),\cdots,\frac{\partial^{|P_k|}}{\partial\nu^{|P_k|}}\boldsymbol{y}^*_\nu(\boldsymbol{x})\right). \tag{13}$$

Since $\mathcal{P}(q,1)$ contains only one element, the above identity implies that

$$\begin{aligned}
&\frac{\partial^q}{\partial \nu^q} \boldsymbol{y}_\nu^*(\boldsymbol{x}) = -\left(\nabla_{yy}^2 g_\nu(\boldsymbol{x}, \boldsymbol{y}_\nu^*(\boldsymbol{x}))\right)^{-1} \sum_{\substack{2 \le k \le q, \\ \boldsymbol{P} \in \mathcal{P}(q,k)}} \boldsymbol{w}_{k,\boldsymbol{P}}, \\
&\text{where } \boldsymbol{w}_{k,\boldsymbol{P}} = \frac{\partial^{k+1}}{\partial \boldsymbol{y}^{k+1}} g_\nu(\boldsymbol{x}, \boldsymbol{y}_\nu^*(\boldsymbol{x})) \left(\frac{\partial^{|P_1|}}{\partial \nu^{|P_1|}} \boldsymbol{y}_\nu^*(\boldsymbol{x}), \cdots, \frac{\partial^{|P_k|}}{\partial \nu^{|P_k|}} \boldsymbol{y}_\nu^*(\boldsymbol{x}) \right).
\end{aligned} \tag{14}$$

Based on Eq. (14), we can prove by induction that $\frac{\partial^q}{\partial \nu^q} \boldsymbol{y}_\nu^*(\boldsymbol{x})$ is $\mathcal{O}(\kappa^{2q+1})$-Lipschitz continuous in ν for all $q = 0, \cdots, p$. The induction base for $q = 0, 1$ is already proved by Chen et al. [11].

Lemma D.2 (Chen et al. [11, Lemma B.2 and B.5]) *Let $\nu \in (0, 1/(2\kappa)]$. Under Assumption 2.3 and 2.4, $\boldsymbol{y}_\nu^*(\boldsymbol{x})$ and $\frac{\partial}{\partial \nu} \boldsymbol{y}_\nu^*(\boldsymbol{x})$ is $\mathcal{O}(\kappa)$- and $\mathcal{O}(\kappa^3)$-Lipschitz continuous in ν, respectively.*

Since Eq. (14) also involves $(\nabla_{yy}^2 g_\nu(\boldsymbol{x}, \boldsymbol{y}_\nu^*(\boldsymbol{x})))^{-1}$, we also need the following lemma that gives its boundedness and Lipschitz continuity constants.

Lemma D.3 (Chen et al. [11, Lemma B.1 and Eq. 18]) *Let $\nu \in (0, 1/(2\kappa)]$. Under Assumption 2.3 and 2.4, $(\nabla_{yy} g_\nu(\boldsymbol{x}, \boldsymbol{y}_\nu^*(\boldsymbol{x})))^{-1}$ is $2/\mu$-bounded and $\mathcal{O}(\kappa^2/\mu)$-Lipschitz continuous in ν.*

In the remaining proofs, we will use Eq. (14) prove by induction that $\frac{\partial^q}{\partial \nu^q} \boldsymbol{y}_\nu^*(\boldsymbol{x})$ is $\mathcal{O}(\kappa^{2q+1})$-Lipschitz continuous in ν, then we can easily use Eq. (12) to show that $\frac{\partial^{q+1}}{\partial \nu^q \partial \boldsymbol{x}} \ell_\nu(\boldsymbol{x})$ is $\mathcal{O}(\kappa^{2q+1} \bar{L})$-Lipschitz continuous in ν for all $q = 0, \cdots, p$. Note that the computational graph of either $\frac{\partial^q}{\partial \nu^q} \boldsymbol{y}_\nu^*(\boldsymbol{x}))$ or $\frac{\partial^{q+1}}{\partial \nu^q \partial \boldsymbol{x}} \ell_\nu(\boldsymbol{x})$ in Eq. (12) or (14) defines a tree, where the root is output, the leaves are inputs, and the other nodes are the intermediate results in the computation. We can analyze the Lipschitz continuities of all the nodes from bottom to top using the following lemma.

Lemma D.4 (Luo et al. [35, Lemma 12]) *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two tensor-to-tensor mappings. If $\mathcal{T}_1$ is D_1-bounded and C_1-Lipschitz continuous, $\mathcal{T}_2$ is D_2-bounded and C_2-Lipschitz continuous, then the product mapping $\mathcal{T}_1 \times \mathcal{T}_2$ is $D_1 D_2$-bounded and $(C_1 D_2 + C_2 D_1)$-Lipschitz continuous.*

Now, let us restate Lemma 3.1 and then prove it.

Lemma 3.1 *Let $\nu \in (0, 1/(2\kappa)]$. For any instance in the pth-order smooth bilevel problem class $\mathcal{F}^{nc\text{-}sc}(L_0, \cdots, L_{p+1}, \mu, \Delta)$ in Definition 2.2, $\frac{\partial^{p+1}}{\partial \nu^p \partial \boldsymbol{x}} \ell_\nu(\boldsymbol{x})$ is $\mathcal{O}(\kappa^{2p+1} \bar{L})$-Lipschitz continuous in ν.*

Proof Now, we formally begin to prove by induction that $\frac{\partial^q}{\partial \nu^q} \boldsymbol{y}_\nu^*(\boldsymbol{x})$ is $\mathcal{O}(\kappa^{2q+1})$-Lipschitz continuous in ν for all $q = 0, \cdots, p$. Recall that the induction base follows Lemma D.2. In the following, we use the induction hypothesis that $\frac{\partial^k}{\partial \nu^k} \boldsymbol{y}_\nu^*(\boldsymbol{x}))$ is $\mathcal{O}(\kappa^{2k+1})$-Lipschitz continuous in ν for all $k = 0, \cdots, q-1$ to prove that $\frac{\partial^q}{\partial \nu^q} \boldsymbol{y}_\nu^*(\boldsymbol{x}))$ is $\mathcal{O}(\kappa^{2q+1})$-Lipschitz continuous in ν. We know that $\frac{\partial^{k+1}}{\partial \boldsymbol{y}^{k+1}} g_\nu(\boldsymbol{x}, \boldsymbol{y}_\nu^*(\boldsymbol{x}))$ is $\mathcal{O}(\bar{L})$-bounded and $\mathcal{O}(\kappa \bar{L})$-Lipschitz continuous in ν. Therefore, we can use Lemma D.4 to conclude that each $\boldsymbol{w}_{k,\boldsymbol{P}}$ is $\mathcal{O}(\kappa^{\sum_{j=1}^k (2|P_j|-1)} \bar{L}) = \mathcal{O}(\kappa^{2q-k} \bar{L})$-bounded and $\mathcal{O}(\bar{L} \cdot \kappa^{2q-k+2} + \kappa \bar{L} \cdot \kappa^{2q-k}) = \mathcal{O}(\kappa^{2q-k+2} \bar{L})$-Lipschitz continuous in ν. It further implies that the summation $\boldsymbol{w} := \sum_{2 \le k \le q, \boldsymbol{P} \in \mathcal{P}(q,k)} \boldsymbol{w}_{k,\boldsymbol{P}}$ is $\mathcal{O}(\kappa^{2q-2} \bar{L})$-bounded and $\mathcal{O}(\kappa^{2q} \bar{L})$-Lipschitz continuous in ν. Then, we can recall Lemma D.3 that $(\nabla_{yy} g_\nu(\boldsymbol{x}, \boldsymbol{y}_\nu^*(\boldsymbol{x})))^{-1}$ is $2/\mu$-bounded and

$\mathcal{O}(\kappa^2/\mu)$-Lipschitz continuous in ν, and use Eq. (14) to finish the induction that $\frac{\partial^q}{\partial \nu^q} \boldsymbol{y}_\nu^*(\boldsymbol{x}) = -\left(\nabla_{yy}^2 g_\nu(\boldsymbol{x}, \boldsymbol{y}_\nu^*(\boldsymbol{x}))\right)^{-1} \boldsymbol{w}$ is $\mathcal{O}(\kappa^{2q+1})$-Lipschitz continuous in ν for all $q = 0, \cdots, p$. Finally, by analogy with the similarity of Eq. (12) and (14), we can follow the same analysis to show that $\frac{\partial^{q+1}}{\partial \nu^q \partial \boldsymbol{x}} \ell_\nu(\boldsymbol{x})$ is $\mathcal{O}(\kappa^{2q+1} \bar{L})$-Lipschitz continuous in ν for all $= 0, \cdots, p$. ∎

Remark D.1 (Tighter bounds for $p = 2$) *Note that the variables $\boldsymbol{x}$ and ν play equal roles in our analysis. Therefore, our result in $p = 2$ essentially implies that $\frac{\partial^3}{\partial \nu \partial \boldsymbol{x}^2} \ell_\nu(\boldsymbol{x})$ is $\mathcal{O}(\kappa^5 \bar{L})$-Lipschitz continuous in ν around zero, which tightens the $\mathcal{O}(\kappa^6 \bar{L})$ bound of Hessian convergence in [11, Lemma 5.1a] and is of independent interest. The main insight is to avoid the direct calculation of $\nabla^2 \varphi(\boldsymbol{x}) = \frac{\partial^3}{\partial \nu \partial \boldsymbol{x}^2} \ell_\nu(\boldsymbol{x})|_{\nu=0}$ which involves third-order derivatives and makes the analysis more complex, but instead always to analyze it through the limiting point $\lim_{\nu \to 0_+} \frac{\partial^3}{\partial \nu \partial \boldsymbol{x}^2} \ell_\nu(\boldsymbol{x})$.*

Appendix E. Proof of Theorem 3.1

Algorithm 2 F^{2}SA-p $(\boldsymbol{x}_0, \boldsymbol{y}_0)$, odd p

```
$\boldsymbol{y}_0^j = \boldsymbol{y}_0, \ \forall j \in \mathbb{N}$
for $t = 0, 1, \cdots, T-1$
    Sample random i.i.d indexes $\{(\xi_j^y, \zeta_j^y)\}_{j=0}^p$ and $\{(\xi_i^x, \zeta_i^x)\}_{i=1}^S$.
    for $j = 0, \cdots, p$
        $\boldsymbol{y}_t^{j,0} = \boldsymbol{y}_t^j$
        for $k = 0, 1, \cdots, K-1$
            $\boldsymbol{y}_t^{j,k+1} = \boldsymbol{y}_t^{j,k} - \eta_y \left( j\nu F_y(\boldsymbol{x}_t, \boldsymbol{y}_t^{j,k}; \xi_j^y) + G_y(\boldsymbol{x}_t, \boldsymbol{y}_t^{j,k}; \zeta_j^y) \right)$
        end for
        $\boldsymbol{y}_{t+1}^j = \boldsymbol{y}_t^{j,K}$
    end for
    Let $\{\beta_j\}_{j=0}^p$ be the $p$th-order forward difference coefficients defined in Lemma C.1.
    $\Phi_t = \frac{1}{S} \sum_{i=1}^S \sum_{j=0}^p \beta_j \left( j F_x(\boldsymbol{x}_t, \boldsymbol{y}_{t+1}^j; \xi_i^x) + \frac{G_x(\boldsymbol{x}_t, \boldsymbol{y}_{t+1}^j; \zeta_i^x)}{\nu} \right)$
    $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta_x \Phi_t / \|\Phi_t\|$
end for
```

In the main text, we only present the algorithm when p is even. The algorithm when p is odd follows a similar design, which is presented in Algorithm 2 for completeness. Our algorithms consist of a double loop, where the outer loop performs normalized SGD (NSGD) and the inner loop performs SGD. Before we give the formal proof, we first recall the convergence result for (N)SGD.

Lemma E.1 (Cutkosky and Mehta [12, Lemma 2]) *Consider the NSGD update $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta F_t/\|F_t\|$ to optimize a function $f : \mathbb{R}^d \to \mathbb{R}$ with L-Lipschitz continuous gradients. We have*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\boldsymbol{x}_t))\| \le \frac{3(f(\boldsymbol{x}_0) - \inf_{\boldsymbol{x}\in\mathbb{R}^d} f(\boldsymbol{x}))}{\eta T} + \frac{3L\eta}{2} + \frac{8}{T}\sum_{t=0}^{T-1}\mathbb{E}\|F_t - \nabla f(\boldsymbol{x}_t)\|.$$

Lemma E.2 (Kwon et al. [29, Lemma C.1]) *Consider the SGD update $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta F_t/\|F_t\|$ to optimize a μ-strongly convex function $f : \mathbb{R}^d \to \mathbb{R}$ with L-Lipschitz continuous gradients. Let $\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}\in\mathbb{R}^d} f(\boldsymbol{x})$ be the unique minimizer to f. Suppose F_t is an unbiased estimator to $\nabla f(\boldsymbol{x}_t)$ with variance bounded by σ^2. Setting $\eta < 2/(\mu + L)$, we have*

$$\mathbb{E}\|\boldsymbol{x}_t - \boldsymbol{x}^*\|^2 \le (1-\mu\eta)^t\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|^2 + \frac{\eta\sigma^2}{\mu}.$$

The following two lemmas are also useful in the analysis.

Lemma E.3 (Chen et al. [11, Lemma 4.1]) *Under Assumption 2.3, and 2.4, the hyper-objective $\varphi(\boldsymbol{x}) = f(\boldsymbol{x}, \boldsymbol{y}^*(\boldsymbol{x}))$ is differentiable and has $L_\varphi = \mathcal{O}(\bar{L}\kappa^3)$-Lipschitz continuous gradients.*

Lemma E.4 (Chen et al. [11, Lemma B.6]) *Let $\nu \in (-1/\kappa, 1/\kappa)$. Under Assumption 2.3, and 2.4, the optimal (perturbed) lower-level solution mapping $\boldsymbol{y}_\nu^*(\boldsymbol{x}) = \arg\min_{\boldsymbol{y}\in\mathbb{R}^{d_y}} \ell_v(\boldsymbol{x}, \boldsymbol{y})$ is 4κ-Lipschitz continuous in $\boldsymbol{x}$.*

Now, we first restate Theorem 3.1 and then prove it.

Theorem 3.1 (Main theorem) *For any instance in the pth-order smooth bilevel problem class $\mathcal{F}^{nc\text{-}sc}(L_0, \cdots, L_{p+1}, \mu, \Delta)$ as per Definition 2.2, set the hyper-parameters as*

$$\begin{gathered}\nu \asymp \min\left\{\frac{R}{\kappa}, \left(\frac{\epsilon}{\bar{L}\kappa^{2p+1}}\right)^{1/p}\right\}, \quad \eta_x \asymp \frac{\epsilon}{L_1\kappa^3}, \quad \eta_y \asymp \frac{\nu^2\epsilon^2}{L_1\kappa\sigma^2}, \\ S \asymp \frac{\sigma^2}{\nu^2\epsilon^2}, \quad K \asymp \frac{\kappa^2\sigma^2}{\nu^2\epsilon^2}\log\left(\frac{RL_1\kappa}{\nu\epsilon}\right), \quad T \asymp \frac{\Delta}{\eta_x\epsilon},\end{gathered} \tag{5}$$

where $R = \|\boldsymbol{y}_0 - \boldsymbol{y}^*(\boldsymbol{x}_0)\|$. Run Algorithm 1 if p is even or Algorithm 2 if p is odd. Then we can provably find an ϵ-stationary point of $\varphi(\boldsymbol{x})$ with the total SFO calls upper bounded by

$$pT(S+K) = \mathcal{O}\left(\frac{p\Delta L_1\bar{L}^{2/p}\sigma^2\kappa^{9+2/p}}{\epsilon^{4+2/p}}\log\left(\frac{RL_1\bar{L}\kappa}{\epsilon}\right)\right).$$

Proof We separately consider the complexity for the outer loop and the inner loop.

Outer Loop. According to Lemma E.3, the hyper-objective $\varphi(\boldsymbol{x})$ has $L_\varphi = \mathcal{O}(\bar{L}\kappa^3)$-Lipschitz continuous gradients. If we can guarantee the condition

$$\mathbb{E}\|\Phi_t - \nabla\varphi(\boldsymbol{x}_t)\| \le \frac{\epsilon}{32}, \quad t = 0, \cdots, T-1, \tag{15}$$

then we can further set $\eta_x = \epsilon/6L_\varphi$ and apply Lemma E.1 to conclude that the algorithm can provably find an ϵ-stationary point of $\varphi(\boldsymbol{x})$ in $T = \lceil 6\Delta/\epsilon\eta_x\rceil = \mathcal{O}(\Delta L_1\kappa^3\epsilon^{-2})$ outer iterations.

Inner Loop. From the above analysis, the remaining goal is to show that the inner loop always returns Φ_t satisfying Eq. (15), which requires $\mathbb{E}\|\Phi_t - \nabla\varphi(\boldsymbol{x}_t)\| = \mathcal{O}(\epsilon)$ for all $t = 0, \cdots, T-1$. Note that the setting of mini-batch size $S = \Omega\left({}^{\sigma^2}\!/_{\nu^2\epsilon^2}\right)$ ensures that

$$\begin{cases} \mathbb{E}\left\|\Phi_t - \sum_{j=-p/2}^{p/2} \alpha_j \left(j\nabla_x f(\boldsymbol{x}_t, \boldsymbol{y}_{t+1}^j) + \dfrac{\nabla_x g(\boldsymbol{x}_t, \boldsymbol{y}_{t+1}^j)}{\nu}\right)\right\| = \mathcal{O}(\epsilon), & p \text{ is even;} \\ \mathbb{E}\left\|\Phi_t - \sum_{j=0}^{p} \beta_j \left(j\nabla_x f(\boldsymbol{x}_t, \boldsymbol{y}_{t+1}^j) + \dfrac{\nabla_x g(\boldsymbol{x}_t, \boldsymbol{y}_{t+1}^j)}{\nu}\right)\right\| = \mathcal{O}(\epsilon), & p \text{ is odd.} \end{cases}$$

By Lemma 3.1 and Lemma C.1, setting $\nu = \mathcal{O}\left(({}^{\epsilon}\!/_{\bar{L}\kappa^{2p+1}})^{1/p}\right)$ can ensure that

$$\begin{cases} \left\|\nabla\varphi(\boldsymbol{x}_t) - \sum_{j=-p/2}^{p/2} \alpha_j \left(j\nabla_x f(\boldsymbol{x}_t, \boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_t)) + \dfrac{\nabla_x g(\boldsymbol{x}_t, \boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_t))}{\nu}\right)\right\| = \mathcal{O}(\epsilon), & p \text{ is even;} \\ \left\|\nabla\varphi(\boldsymbol{x}_t) - \sum_{j=0}^{p} \beta_j \left(j\nabla_x f(\boldsymbol{x}_t, \boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_t)) + \dfrac{\nabla_x g(\boldsymbol{x}_t, \boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_t))}{\nu}\right)\right\| = \mathcal{O}(\epsilon), & p \text{ is odd.} \end{cases}$$

Therefore, a sufficient condition of $\mathbb{E}\|\Phi_t - \nabla\varphi(\boldsymbol{x}_t)\| = \mathcal{O}(\epsilon)$ is

$$\begin{cases} \|\boldsymbol{y}_{t+1}^j - \boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_t)\| = \mathcal{O}(\nu\epsilon/L_1), \ \ \forall j = -p/2, \cdots, p/2, & p \text{ is even;} \\ \|\boldsymbol{y}_{t+1}^j - \boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_t)\| = \mathcal{O}(\nu\epsilon/L_1), \ \ \forall j = 0, \cdots, p, & p \text{ is odd.} \end{cases} \tag{16}$$

Our next goal is to show that our parameter setting fulfills Eq. (16). Note that for $\nu = \mathcal{O}(1/\kappa)$, the (perturbed) lower-level problem $g_{j\nu}(\boldsymbol{x}, \boldsymbol{y})$ is $\Omega(\mu)$-strongly convex in $\boldsymbol{y}$ and has $\mathcal{O}(L_1)$-Lipschitz continuous gradients jointly in $(\boldsymbol{x}, \boldsymbol{y})$. Therefore, if we set $\eta_y \lesssim 1/L_1$, then we can apply Lemma E.2 on the lower-level problem $g_{j\nu}(\boldsymbol{x}, \boldsymbol{y})$ to conclude that for ant j, we have

$$\mathbb{E}\|\boldsymbol{y}_{t+1} - \boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_t)\|^2 \leq (1 - \mu\eta_y)^K \|\boldsymbol{y}_t - \boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_t)\|^2 + \mathcal{O}(\eta_y\sigma^2/\mu).$$

Comparing it with Eq. (16), we can set $\eta_y = \mathcal{O}\left({}^{\nu^2\epsilon^2}\!/_{L_1\kappa\sigma^2}\right)$ to ensure that for ant j, we have

$$\mathbb{E}\|\boldsymbol{y}_{t+1} - \boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_t)\| \leq (1 - \mu\eta_y)^K \|\boldsymbol{y}_t - \boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_t)\| + \mathcal{O}(\nu\epsilon/L_1).$$

Further, we can use Lemma E.4 and the triangle inequality to obtain that for ant j, we have

$$\mathbb{E}\|\boldsymbol{y}_{t+1} - \boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_t)\| \leq (1 - \mu\eta_y)^K (\|\boldsymbol{y}_t - \boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_{t-1})\| + 4\kappa\|\boldsymbol{x}_t - \boldsymbol{x}_{t-1}\|) + \mathcal{O}(\nu\epsilon/L_1). \tag{17}$$

The recursion (17) implies our setting of K can ensure that Eq. (16) holds for all $t = 0, \cdots, T-1$. We give an induction-based proof. To let the induction base holds for $t = 1$, it suffices to set $K = \Omega\left({}^{\log(RL_1/\nu\epsilon)}\!/_{\mu\eta_y}\right) = \Omega\left({}^{\log(RL_1/\nu\epsilon)\kappa^2\sigma^2}\!/_{\nu^2\epsilon^2}\right)$, where $\|\boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_0) - \boldsymbol{y}^*(\boldsymbol{x}_0)\|^2 = \mathcal{O}(R)$ is due to the setting of $\nu = \mathcal{O}\left({}^{R}\!/_{\kappa}\right)$ and the fact that $\boldsymbol{y}_\nu^*(\boldsymbol{x})$ is κ-Lipschitz in ν by Lemma D.2. Next, assume that we have already guaranteed Eq. (16) holds for iteration t, we prove that our setting of K implies Eq. (16) holds for iteration $t+1$. Note that the NSGD update in $\boldsymbol{x}$ means that $\|\boldsymbol{x}_t - \boldsymbol{x}_{t-1}\| = \eta_x = \mathcal{O}\left({}^{\epsilon}\!/_{6L_1\kappa^3}\right)$. Therefore, Eq. (17) in conjunction with the induction hypothesis indicates that

$$\mathbb{E}\|\boldsymbol{y}_{t+1} - \boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_t)\| \lesssim (1 - \mu\eta_y)^K \left(\frac{\nu\epsilon}{L_1} + \frac{\epsilon}{L_1\kappa^2}\right) + \frac{\nu\epsilon}{L_1}.$$

Therefore, we know that to let Eq. (16) holds for iteration $t+1$, it suffices to let $K = \Omega\left({}^{\log(1/\nu\kappa^2)}\!/_{\mu\eta_y}\right) = \Omega\left({}^{\log(1/\nu\kappa^2)\kappa^2\sigma^2}\!/_{\nu^2\epsilon^2}\right)$. This finishes the induction.

Total Complexity. According to the above analysis, we set $\nu \asymp (\epsilon/\bar{L}\kappa^{2p+1})^{1/p}$, $S \asymp \sigma^2/\nu^2\epsilon^2$, $T \asymp \Delta L_1 \kappa^3 \epsilon^{-2}$, and $K \asymp \log(RL_1\kappa/\nu\epsilon)\kappa^2\sigma^2/\nu^2\epsilon^2$ to ensure that the algorithm provably find an ϵ-stationary point of $\varphi(\boldsymbol{x})$. Since $S \lesssim K$, the total complexity of the algorithm is

$$pT(S+K) = \mathcal{O}(pTK) = \mathcal{O}\left(p \cdot \frac{\Delta L_1 \kappa^3}{\epsilon^2} \cdot \frac{\kappa^2\sigma^2}{\nu^2\epsilon^2} \log\left(\frac{RL_1\kappa}{\nu\epsilon}\right)\right)$$
$$= \mathcal{O}\left(\frac{p\Delta L_1 \bar{L}^{2/p}\sigma^2\kappa^{9+2/p}}{\epsilon^{4+2/p}} \log\left(\frac{RL_1\kappa}{\nu\epsilon}\right)\right).$$

Remark E.1 (Comparison of results for odd p and even p.) *Note that by Lemma C.1 when p is odd, we need to use $p+1$ points to construct the estimator, which means the algorithm needs to solve $p+1$ lower-level problems in each iteration to achieve an $\mathcal{O}(\nu^p)$ error guarantee. In contrast, when p is even, p points are enough since the pth-order central difference estimator satisfies that $\alpha_0 = 0$. It suggests that even when p is odd, the algorithm designed for odd p may still be better. For instance, the F^{2}SA-2 may always be a better choice than F^{2}SA since its benefits almost come for free: (1) it still only needs to solve 2 lower-level problems as the F^{2}SA method, which means the per-iteration complexity remains the same. (2) Although the improved complexity of F^{2}SA-2 relies on the second-order smooth condition, without such a condition, its error guarantee in hyper-gradient estimation only degenerates to a first-order one, which means it is at least as good as F^{2}SA.*

■

Appendix F. An $\Omega(\epsilon^{-4})$ Lower Bound

In this section, we prove an $\Omega(\epsilon^{-4})$ lower bound for stochastic bilevel optimization via a reduction to single-level optimization. Our lower bound holds for any randomized algorithms $\mathtt{A}$, which consists of a sequence of measurable mappings $\{\mathtt{A}_t\}_{t=1}^T$ that is defined recursively by

$$(\boldsymbol{x}_{t+1}, \boldsymbol{y}_{t+1}) = \mathtt{A}_t\left(r, F(\boldsymbol{x}_0, \boldsymbol{y}_0), G(\boldsymbol{x}_0, \boldsymbol{y}_0)), \cdots, F(\boldsymbol{x}_t, \boldsymbol{y}_t), G(\boldsymbol{x}_t, \boldsymbol{y}_t))\right), \quad t \in \mathbb{N}_+, \tag{18}$$

where r is a random seed drawn at the beginning to produce the queries, and F, G are the stochastic gradient estimators that satisfy Assumption 2.1. Without loss of generality, we assume that $(\boldsymbol{x}_0, \boldsymbol{y}_0) = (\boldsymbol{0}, \boldsymbol{0})$. Otherwise, we can prove the same lower bound by shifting the functions.

The construction. We construct a separable bilevel instance such that the upper-level function $f(\boldsymbol{x}, \boldsymbol{y}) \equiv f_{\boldsymbol{U}}(\boldsymbol{x})$ and its stochastic gradient align with the hard instance in [3], while the lower-level function is the simple quadratic $g(\boldsymbol{x}, y) \equiv g(y) = \mu y^2/2$ with deterministic gradients. Specially, we follow Arjevani et al. [3] to uniformly draw matrix $\boldsymbol{U} \in \mathbb{R}^{d \times T}$ from the class of all column orthogonal matrices such that $\boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_T$ and define the upper- and lower-level functions $f_{\boldsymbol{U}} : \mathbb{R}^d \to \mathbb{R}$, $g : \mathbb{R} \to \mathbb{R}$ as

$$f_{\boldsymbol{U}}(\boldsymbol{x}) = \frac{L_1\beta^2}{\bar{L}_1} f^{\mathrm{nc}}(\rho(\boldsymbol{U}^\top \boldsymbol{x}/\beta)) + \frac{L_1\lambda}{2\bar{L}_1}\|\boldsymbol{x}\|^2, \quad g(y) = \frac{\mu}{2}y^2, \tag{19}$$

where $\bar{L}_1 = 155$, $\beta = 4\bar{L}_1\epsilon/L_1$, $\rho : \mathbb{R}^T \to \mathbb{R}^T$ is $\rho(\boldsymbol{x}) = \boldsymbol{x}/\sqrt{1 + \|\boldsymbol{x}\|^2/R^2}$, $R = 230\sqrt{T}$, $\lambda = 1/5$, and $f_T : \mathbb{R}^T \to \mathbb{R}$ is the nonconvex hard instance introduced by Carmon et al. [6]:

$$f^{\mathrm{nc}}(x) := -\Psi(1)\Psi(x_1) + \sum_{i=2}^{T}[\Phi(-x_{i-1})\Phi(-x_i) - \Psi(x_{i-1})\Phi(x_i)].$$

In the above, the component functions $\Psi, \Phi : \mathbb{R} \to \mathbb{R}$ are defined as

$$\Psi(t) = \begin{cases} 0, & t \leq 1/2, \\ \exp(1 - 1/(2t-1)^2), & t < 1/2 \end{cases} \quad \text{and} \quad \Phi(t) = \sqrt{\mathrm{e}} \int_{-\infty}^{t} \exp(-t^2/2)\mathrm{d}t.$$

For our hard instance in Eq. (19), we define the stochastic gradient estimator $F_{\boldsymbol{U}}$ and G as

$$F_{\boldsymbol{U}}(\boldsymbol{x}) = \frac{L_1}{\bar{L}_1}\left(\beta(\nabla\rho(\boldsymbol{x}))^\top \boldsymbol{U} F_T(\boldsymbol{U}^\top\rho(\boldsymbol{x})) + \lambda\boldsymbol{x}\right) \quad \text{and} \quad G(y) = \mu y. \tag{20}$$

In the above, $F_T : \mathbb{R}^T \to \mathbb{R}^T$ is the stochastic gradient estimator of ∇f^{nc} defined by

$$[F_T(\boldsymbol{x})]_i = \nabla_i f^{\mathrm{nc}}(\boldsymbol{x})\left(1 + \mathbf{1}_{i > \mathrm{prog}_{1/4}(x)}(\xi/\gamma - 1)\right), \quad \xi \sim \mathrm{Bernoulli}(\gamma),$$

where $\mathrm{prog}_\alpha(x) = \max\{i \geq 0 \mid |x_i| > \alpha\}$ and $\gamma = \min\{(46\epsilon)^2/\sigma^2, 1\}$.

For this separable bilevel instance, we can show that for any randomized algorithm defined in Eq. (18) that uses oracles $(F_{\boldsymbol{U}}, G)$, the progress in $\boldsymbol{x}$ can be simulated by another randomized algorithm that only uses $F_{\boldsymbol{U}}$, meaning that the single-level lower bound [3] also holds.

Theorem F.1 (Lower bound) *There exist numerical constants $c > 0$ such that for all $\Delta > 0$, $L_1, L_2, \cdots, L_{p+1} > 0$ and $\epsilon \leq c\sqrt{L_1\Delta}$, there exists a distribution over the function class $\mathcal{F}^{nc\text{-}sc}(L_0, L_2, \cdots, L_{p+1}, \mu, \Delta)$ and the stochastic gradient estimators satisfying Assumption 2.1, such that any randomized algorithm $\mathtt{A}$ defined as Eq. (18) can not find an ϵ-stationary point of $\varphi(\boldsymbol{x}) = f(\boldsymbol{x}, \boldsymbol{y}^*(\boldsymbol{x}))$ in less than $\Omega(\Delta L_1\sigma^2\epsilon^{-4})$ SFO calls.*

Proof For any randomized algorithm $\mathtt{A}$ defined as Eq. (18) running it on our hard instance, we show that it can be simulated by another randomized algorithm running on the variable $\boldsymbol{x}$ such that the lower bound in [3] can be applied. Since $G(y) = \mu y$ is a deterministic mapping we know that any randomized algorithm $\mathtt{A}$ induces a sequence of measurable mappings $\{\mathtt{A}'_t\}_{t\in\mathbb{N}}$ such that

$$(\boldsymbol{x}_t, y_t) = \mathtt{A}'_t(\xi, F(\boldsymbol{x}_0), \cdots, F(\boldsymbol{x}_{t-1}), y_0, \cdots, y_{t-1}).$$

Expanding the recursion for y_t shows that the above equation induces another sequence of measurable mappings $\{\mathtt{A}''_t\}_{t\in\mathbb{N}}$ such that

$$(\boldsymbol{x}_t, y_t) = \mathtt{A}''_t(\xi, F(\boldsymbol{x}_0), \cdots, F(\boldsymbol{x}_{t-1})).$$

It means that the iterate $\boldsymbol{x}_t$ can be simulated via a measurable mapping from $(r, F(\boldsymbol{x}_0), \cdots, F(\boldsymbol{x}_{t-1}))$, for which [3, Theorem 3] shows that the function $f_{\boldsymbol{U}} : \mathbb{R}^T \to \mathbb{R}$ and its associated stochastic first-order oracle $\boldsymbol{F}_{\boldsymbol{U}} : \mathbb{R}^T \to \mathbb{R}^T$ gives the $\Omega(\Delta L_1\sigma^2\epsilon^{-4})$ lower bound as required. ■

The analysis is simple using our fully separable construction $f(\boldsymbol{x}, y) = f_{\boldsymbol{U}}(\boldsymbol{x})$ and $g(y) = \mu y^2/2$. But we are a bit surprised that our straightforward construction is not used in prior works such as [14]. Below, we give a detailed discussion on the constructions in other works.

Comparison to other bilevel lower bounds. Dagréou et al. [14] proved lower bounds for finite-sum bilevel optimization via a similar reduction to single-level optimization. However, the direct extension of their construction in the fully stochastic setting gives $f(\boldsymbol{x}, \boldsymbol{y}) = f_{\boldsymbol{U}}(\boldsymbol{y})$ and $g(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} - \boldsymbol{y})^2$, where the high-order derivatives of $f(\boldsymbol{x}, \boldsymbol{y})$ not $\mathcal{O}(1)$-Lipschitz in $\boldsymbol{y}$ and thus violates our assumptions. Kwon et al. [29] also proved an $\Omega(\epsilon^{-4})$ lower bound for stochastic bilevel optimization. However, their construction $f(\boldsymbol{x}, y) = y$ and $g(\boldsymbol{x}, y) = (f_{\boldsymbol{U}}(\boldsymbol{x}) - y)^2$ violate the first-order smoothness of $g(\boldsymbol{x}, y)$ in $\boldsymbol{x}$ when y is far way from $f_{\boldsymbol{U}}(\boldsymbol{x})$. In this work, we use a fully separable construction to avoid all the aforementioned issues in other works.

Appendix G. Experiments

In this section, we conduct numerical experiments to verify our theory. We consider the “learn-to-regularize” problem on the “20 Newsgroup” dataset, which is a very standard benchmark in bilevel optimization [11, 20, 22, 33]. In this task, we aim at learning the optimal regularizer for each parameter of a model. We formulate this task into the following bilevel optimization problem:

$$\min_{\boldsymbol{x} \in \mathbb{R}^p} \ell^{\mathrm{val}}(\boldsymbol{y}), \quad \text{s.t.} \quad \boldsymbol{y} \in \arg\min_{\boldsymbol{y} \in \mathbb{R}^{q \times p}} \ell^{\mathrm{tr}}(\boldsymbol{y}) + \|\boldsymbol{W}_{\boldsymbol{x}} \boldsymbol{y}\|^2,$$

where $\boldsymbol{x}$ parameterizes the regularization matrix via $\boldsymbol{W}_{\boldsymbol{x}} = \mathrm{diag}(\exp(\boldsymbol{x}))$, $\boldsymbol{y}$ parameterizes a linear model that maps $p = 130,107$ features to $q = 20$ classes, while ℓ^{val} and ℓ^{tr} denote the validation and training loss, respectively. Using the logistic loss function, it is clear that the objective is arbitrarily smooth. The whole dataset contains 18,000 samples. We compare our proposed method F^{2}SA-p with both the previous best fully first-order method F^{2}SA and other Hessian-vector-product-based methods stocBiO [22], MRBO and VRBO [46]. We tune p in $\{1, 2, \cdots, 5\}$ and find that $p = 2$ is the optimal choice. One possible reason is that the instance of both $p = 1, 2$ only requires solving two lower-level problems at each iteration, but the instance of $p \geq 3$ requires solving more than three lower-level problems and may not be concretely efficient. We regard F^{2}SA-2 as an important instantiation of F^{2}SA-p and present its concrete procedure in Algorithm 3. For all the algorithms, we search the optimal hyperparameters (including η_x, η_y, ν) in a logarithmic scale with base 10 and present the experiment results in Figure 1, where we also include a line “w/o Reg” that means the baseline without tuning any regularization. It can be observed that: (1) all the hessian-vector-product-based methods are worse than fully first-order methods; (2) the variance reduction technique in VRBO/MRBO is ineffective and may even harm the performance, which also aligns with the findings in [33]; (3) our method F^{2}SA-2 significantly outperforms all the other algorithms. Our preliminary experiment results on the standard benchmark show the potential of F^{2}SA-2 on large-scale bilevel problems.

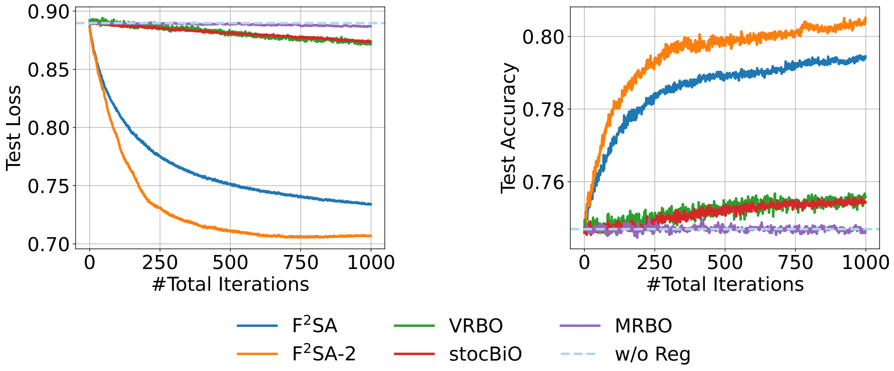

Figure 1: Performances of different algorithms when learning the optimal regularization.

Algorithm 3 F^{2}SA-2 $(\boldsymbol{x}_0, \boldsymbol{y}_0)$

```
$\boldsymbol{z}_0 = \boldsymbol{y}_0$
for $t = 0, 1, \cdots, T-1$
    Sample random i.i.d indexes $(\xi^y, \zeta^y)$, $(\xi^z, \zeta^z)$, and $\{(\xi_i^x, \zeta_i^x)\}_{i=1}^S$.
    $\boldsymbol{y}_t^0 = \boldsymbol{y}_t, \ \boldsymbol{z}_t^0 = \boldsymbol{z}_t$
    for $k = 0, 1, \cdots, K-1$
        $\boldsymbol{y}_t^{k+1} = \boldsymbol{y}_t^k - \eta_y \left(\nu F_y(\boldsymbol{x}_t, \boldsymbol{y}_t^k; \xi^y) + G_y(\boldsymbol{x}_t, \boldsymbol{y}_t^k; \zeta^y)\right)$
        $\boldsymbol{z}_t^{k+1} = \boldsymbol{z}_t^k - \eta_y \left(-\nu F_y(\boldsymbol{x}_t, \boldsymbol{z}_t^t; \xi^z) + G_y(\boldsymbol{x}_t, \boldsymbol{z}_t^k; \zeta^z)\right)$
    end for
    $\boldsymbol{y}_{t+1} = \boldsymbol{y}_t^K, \ \boldsymbol{z}_{t+1} = \boldsymbol{z}_t^K$
    $\Phi_t = \frac{1}{2}\sum_{i=1}^S \left(F_x(\boldsymbol{x}_t, \boldsymbol{y}_{t+1}; \xi_i^x) + F_x(\boldsymbol{x}_t, \boldsymbol{z}_{t+1}; \xi_i^x) + \frac{G_x(\boldsymbol{x}_t, \boldsymbol{y}_{t+1}; \zeta_i^x) - G_x(\boldsymbol{x}_t, \boldsymbol{z}_{t+1}; \zeta_i^x)}{\nu}\right)$
    $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta_x \Phi_t / \|\Phi_t\|$
end for
```