Cyclic Block Coordinate Descent With Variance Reduction for Composite Nonconvex Optimization

Xufeng Cai [1] **Chaobing Song** [1] **Stephen J. Wright** [1] **Jelena Diakonikolas** [1]

Abstract

Nonconvex optimization is central in solving many machine learning problems, in which block-wise structure is commonly encountered. In this work, we propose cyclic block coordinate methods for nonconvex optimization problems with non-asymptotic gradient norm guarantees. Our convergence analysis is based on a gradient Lipschitz condition with respect to a Mahalanobis norm, inspired by a recent progress on cyclic block coordinate methods. In deterministic settings, our convergence guarantee matches the guarantee of (full-gradient) gradient descent, but with the gradient Lipschitz constant being defined w.r.t. a Mahalanobis norm. In stochastic settings, we use recursive variance reduction to decrease the per-iteration cost and match the arithmetic operation complexity of current optimal stochastic full-gradient methods, with a unified analysis for both finite-sum and infinite-sum cases. We prove a faster linear convergence result when a Polyak-Łojasiewicz (PŁ) condition holds. To our knowledge, this work is the first to provide non-asymptotic convergence guarantees — variance-reduced or not — for a cyclic block coordinate method in general composite (smooth + non-smooth) nonconvex settings. Our experimental results demonstrate the efficacy of the proposed cyclic scheme in training deep neural nets.

1. Introduction

Exploiting structural information in machine learning (ML) problems is key to enabling optimization at extreme scale. Important examples of such structure are block separability, giving rise to block coordinate methods, and finite/infinite sum structure, giving rise to stochastic, possibly variance-reduced optimization methods. In this work, we explore both these types of structure to develop novel optimization methods with fast convergence.

We focus on nonconvex optimization problems of the form

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} F(\boldsymbol{x}) = f(\boldsymbol{x}) + r(\boldsymbol{x}), \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^d$ can be partitioned into m disjoint blocks $\boldsymbol{x} = (\boldsymbol{x}^1, \ldots, \boldsymbol{x}^m)$ with $\boldsymbol{x}^j \in \mathbb{R}^{d_j}$ for $j \in [m]$ and $\sum_{j=1}^m d_j = d$; $f(\boldsymbol{x})$ is a smooth nonconvex function; $r(\boldsymbol{x}) = \sum_{j=1}^m r^j(\boldsymbol{x}^j)$ is block separable, extended-valued, closed convex function such that each $r^j(\cdot)$ (and thus the separable sum r) admits an efficiently computable proximal operator. We consider in particular the finite-sum variant of (1):

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} F(\boldsymbol{x}) = f(\boldsymbol{x}) + r(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^n f_i(\boldsymbol{x}) + r(\boldsymbol{x}), \tag{2}$$

in which $f(\boldsymbol{x})$ is nonconvex and smooth and n is usually very large. Without loss of generality, due to the central limit theorem, we use $n = +\infty$ in (2) to refer to the following stochastic (infinite-sum) optimization setting:

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} F(\boldsymbol{x}) = \mathbb{E}_{\xi\sim\mathcal{D}}[f(\boldsymbol{x};\xi)] + r(\boldsymbol{x}), \tag{3}$$

where ξ is a random variable from an unknown distribution $\mathcal{D}$. Problems of the form (2) and (3) commonly arise in machine learning, especially in (regularized, empirical, or population) risk minimization.

1.1. Motivation and Related Work

Both block coordinate and variance-reduced stochastic gradient methods are prevalent in machine learning, due to their effectiveness in handling large problem instances; see e.g., Gorbunov et al. (2020); Wright (2015); Allen-Zhu et al. (2016); Nesterov (2012); Allen-Zhu (2017); Johnson & Zhang (2013); Diakonikolas & Orecchia (2018); Nakamura et al. (2021); Li et al. (2021); Beck & Tetruashvili (2013); Hong et al. (2017); Xu & Yin (2015); Chen & Gu (2016) and references therein.

Block coordinate methods can be classified into three main categories according to the order in which blocks of coor-

[1]Department of Computer Sciences, University of Wisconsin-Madison, Madison, WI, USA. Correspondence to: Xufeng Cai <xcai74@wisc.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

dinates are selected: (i) greedy, or Gauss-Southwell methods (Nutini et al., 2015), which in each iteration selects the block of coordinates that lead to the highest progress in minimizing the objective function; (ii) randomized block coordinate methods, which select blocks of coordinates at random (with replacement), according to some pre-defined probability distribution (Nesterov, 2012); and (iii) cyclic block coordinate methods, which update the coordinate blocks in a cyclic order (Beck & Tetruashvili, 2013). (A combination of (ii) and (iii) known as random-permutations methods uses a cyclic approach but randomly reshuffles the order in which the blocks are updated at the start of each cycle.) Greedy methods can be quite effective in practice when their selection rule can be implemented efficiently, but they are applicable only to very specialized problems. Thus, most of the focus has been on randomized and cyclic methods.

From a theoretical standpoint, randomized methods have received much more attention than cyclic methods. The reason is that the randomly selected block of gradient coordinates can be related to the full gradient by taking the expectation, which allows their analysis to be reduced to the analysis of standard first-order methods; see Nesterov (2012); Nesterov & Stich (2017); Allen-Zhu et al. (2016); Diakonikolas & Orecchia (2018). By contrast, cyclic methods are much more challenging to analyze, as it is unclear how to relate the partial gradient to the full one. Obtaining non-asymptotic convergence guarantees for such methods was initially considered nearly impossible (Nesterov, 2012). Despite much of the progress on the theoretical front (Beck & Tetruashvili, 2013; Saha & Tewari, 2013; Gurbuzbalaban et al., 2017; Lee & Wright, 2019; Wright & Lee, 2020; Li et al., 2017; Sun & Ye, 2021), most of the literature addressing cyclic methods deals with convex (often quadratic) objective functions and provides convergence guarantees that are typically worse by a factor polynomial in the dimension d than the equivalent guarantees for randomized methods. For nonconvex objectives, there are few existing guarantees, and these require additional assumptions such as multiconvexity (i.e., that the function is convex over a coordinate block when other blocks of coordinates remain fixed) and the Kurdyka-Łojasiewicz (KŁ) property, or else provide convergence guarantees that are only asymptotic (Xu & Yin, 2013; 2015; 2017; Zeng et al., 2014). On the other hand, the recent work by Song & Diakonikolas (2021) avoids the explicit dependence on the dimension by introducing a novel Lipschitz condition that holds w.r.t. a Mahalanobis norm. This condition is the inspiration for the methods proposed in our work, but the techniques in Song & Diakonikolas (2021) cannot be applied directly to the current context of composite nonconvex problems, as they address monotone variational inequalities. An entirely separate analysis framework is required, and is presented here.

From the implementation viewpoint, randomized methods require generating pseudo-random numbers from a predefined probability distribution to determine which coordinate block should be selected in each iteration. This operation may dominate the arithmetic cost when the coordinate update is cheap. Cyclic methods are simple, intuitive, and more efficient for implementation, and often demonstrate better empirical performance than the randomized methods (Beck & Tetruashvili, 2013; Chow et al., 2017; Sun & Ye, 2021). They are thus the default algorithms for many software packages such as SparseNet (Mazumder et al., 2011) and GLMNet (Friedman et al., 2010) in high-dimensional computational statistics and have found wide applications in areas such as variational inference (Blei et al., 2017; Plummer et al., 2020), non-negative matrix factorization (Vandaele et al., 2016), k-means clustering (Nie et al., 2021), and phase retrieval (Zeng & So, 2020).

More recent literature has also sought to combine the favorable properties of stochastic optimization methods (such as SGD) with block coordinate updates; see, e.g., Xu & Yin (2015); Nakamura et al. (2021); Fu et al. (2020); Chen & Gu (2016); Lei & Shanbhag (2020); Wang et al. (2016)), which address nonconvex problems of the form (2) and (3). Compared with traditional stochastic gradient methods, which simultaneously update all variables using Gauss-Jacobi-style iterations, block-coordinate variants of stochastic gradient update the variables sequentially with Gauss-Seidel-style iterations, thus usually taking fewer iterations to converge (see e.g., Xu & Yin (2015)). One common approach to further improve sample complexity in stochastic optimization is to use variance reduction, which for block coordinate methods in nonconvex settings has been done in Chen & Gu (2016); Chauhan et al. (2017); Lei & Shanbhag (2020). However, to the best of our knowledge, non-asymptotic convergence results have only been established for randomized methods with variance reduction (Chen & Gu, 2016; Lei & Shanbhag, 2020). We are not aware of work that incorporates variance reduction techniques with cyclic methods in nonconvex settings. Even in the *convex* setting, the only work we are aware of that combines variance reduction with a cyclic method is Song & Diakonikolas (2021), but this paper utilizes SVRG-style variance reduction, whose applicability in nonconvex settings is unclear.

1.2. Contributions

Our main contributions can be summarized as follows.

Proximal Cyclic Block Coordinate Descent (P-CCD). We provide a non-asymptotic convergence analysis for the standard P-CCD method in deterministic nonconvex settings, based on a Lipschitz condition w.r.t. a Mahalanobis norm, inspired by the recent work by Song & Diakonikolas (2021). However, the techniques are completely disjoint and their results (for monotone variational inequalities) nei-

ther imply ours (for nonconvex minimization), nor the other way around. Our Lipschitz condition, which implies block (coordinate) smoothness, is more general. The comparison between the new Lipschitz condition and the standard (Euclidean-norm) Lipschitz condition is discussed in Section 2. We show that P-CCD has the same sublinear convergence rate as full gradient methods, and achieves linear convergence under a PŁ condition. To the best of our knowledge, these are the first such results for a cyclic method in the composite nonconvex setting (1), where standard tools such as monotonicity (convex inequalities) used in the earlier paper cannot be used to establish convergence.

Variance-Reduced P-CCD. We propose a stochastic gradient variant of P-CCD with recursive variance reduction for solving nonconvex problems of the form (2) and (3). The recursive variance reduction technique of Li et al. (2021) was used prior to our work only in the full-gradient setting, and the extension to the cyclic block coordinate setting requires addressing nontrivial technical obstacles such as establishing a new potential function and controlling additional error terms arisen from the cyclic update rule. We prove its non-asymptotic convergence using an analysis that unifies the finite-sum and infinite-sum settings, which also matches the arithmetic operation complexity of optimal stochastic full-gradient methods for nonconvex minimization. A faster, linear convergence rate is attained under a PŁ condition. To our knowledge, our work is the first to incorporate variance reduction into cyclic methods in nonconvex settings while providing non-asymptotic convergence guarantees.

Numerical Experiments. We apply our proposed cyclic algorithms to train LeNet on the CIFAR-10 dataset, and compare them with SGD and the PAGE algorithm (Li et al., 2021). Our preliminary results demonstrate that the cyclic methods converge faster with better generalization than full gradient methods when using large batch sizes, thus shedding light on the possibility of remedying the drawbacks of large-batch methods (Keskar et al., 2017).

1.3. Further Related Work

Both block coordinate methods and variance reduction techniques in stochastic optimization have been subjects of much research. For conciseness, we review only the additional literature that is most closely related to our work.

Block Coordinate Descent. Block coordinate methods have been widely used in both convex and nonconvex applications such as feature selection in high-dimensional computational statistics (Wu & Lange, 2008; Friedman et al., 2010; Mazumder et al., 2011) and empirical risk minimization in machine learning (Nesterov, 2012; Lin et al., 2015; Allen-Zhu et al., 2016; Alacaoglu et al., 2017; Diakonikolas & Orecchia, 2018; Xu & Yin, 2015). The convergence of block coordinate methods has been extensively studied for various settings, see e.g., Grippof & Sciandrone (1999); Tseng (2001); Razaviyayn et al. (2013); Xu & Yin (2015); Song & Diakonikolas (2021) and references therein. In nonconvex settings, asymptotic convergence of block coordinate methods was established in Chen et al. (2021); Xu & Yin (2017). In terms of non-asymptotic convergence guarantees, Chen & Gu (2016) provides such a result for a randomized method under a sparsity constraint and restricted strong convexity, while Xu & Yin (2017; 2013) provides results for cyclic methods under the KŁ property. For a stochastic gradient variant of a cyclic method, *asymptotic* convergence was analyzed by Xu & Yin (2015).

Variance Reduction. To address the issue of the constant variance of the (minibatch) gradient estimator, several variance reduction methods have been proposed. SAG (Schmidt et al., 2017) was the first stochastic gradient method with a linear convergence rate for strongly convex finite-sum problems, and was based on a biased gradient estimator. Johnson & Zhang (2013) and Defazio et al. (2014) improved SAG by proposing unbiased estimators of SVRG-type and SAGA-type, respectively. These estimators were further enhanced with Nesterov acceleration (Allen-Zhu, 2017; Song et al., 2020) and applied to nonconvex finite-sum/infinite-sum problems (Reddi et al., 2016; Lei et al., 2017). For nonconvex stochastic (infinite-sum) problems, the recursive variance reduction estimators SARAH (Nguyen et al., 2017) and SPIDER (Fang et al., 2018; Zhou et al., 2018a;b) were proposed to attain the optimal oracle complexity of $\mathcal{O}(1/\epsilon^3)$ for finding an ϵ-approximate stationary point. PAGE (Li et al., 2021) and STORM (Cutkosky & Orabona, 2019) further simplified SARAH and SPIDER by reducing the number of loops and avoiding large minibatches.

2. Preliminaries

We consider a real d-dimensional Euclidean space $(\mathbb{R}^d, \|\cdot\|)$, where $\|\cdot\| = \sqrt{\langle\cdot,\cdot\rangle}$ is induced by the (standard) inner product associated with the space and d is finite. For any given positive integer m, we use $[m]$ to denote the set $\{1, 2, \ldots, m\}$. We assume that we are given a positive integer $m \leq d$ and a partition of the coordinates $[d]$ into nonempty sets $\mathcal{S}^1, \mathcal{S}^2, \ldots, \mathcal{S}^m$. We let $\boldsymbol{x}^j$ denote the subvector of $\boldsymbol{x}$ indexed by the coordinates contained in $\mathcal{S}^j$ and let $d^j := |\mathcal{S}^j|$ denote the size of the set $\mathcal{S}^j$, for $j \in [m]$. To simplify the notation, we assume that the partition into sets $\mathcal{S}^1, \mathcal{S}^2, \ldots, \mathcal{S}^m$ is ordered, in the sense that for $1 \leq j < j' \leq m$, $\max_{i\in\mathcal{S}^j} i < \min_{i'\in\mathcal{S}^{j'}} i'$. This assumption is without loss of generality, as our results are invariant to permutations of the coordinates. Given a matrix $\boldsymbol{A}$, we let $\|\boldsymbol{A}\| := \sup\{\boldsymbol{A}\boldsymbol{x} : \boldsymbol{x} \in \mathbb{R}^d, \|\boldsymbol{x}\| \leq 1\}$ denote the standard operator norm. For a positive definite symmetric matrix $\boldsymbol{A}$, $\|\cdot\|_{\boldsymbol{A}}$ denotes the Mahalanobis norm defined by $\|\boldsymbol{x}\|_{\boldsymbol{A}} = \sqrt{\langle \boldsymbol{A}\boldsymbol{x}, \boldsymbol{x}\rangle}$. We use $\boldsymbol{I}_d$ to denote the identity

matrix of size $d \times d$; when the context is clear, we omit the subscript. For a sequence of positive semidefinite $d \times d$ matrices $\{\boldsymbol{Q}^j\}_{j=1}^m$, we define $\widehat{\boldsymbol{Q}}^j$ by

$$(\widehat{\boldsymbol{Q}}^j)_{t,k} = \begin{cases} (\boldsymbol{Q}^j)_{t,k}, & \text{if } \min\{t,k\} > \sum_{\ell=1}^{j-1} d^\ell, \\ 0, & \text{otherwise,} \end{cases}$$

which corresponds to the matrix $\boldsymbol{Q}^j$ with first $j-1$ blocks of rows and columns set to zero. Similarly, we define $\widetilde{\boldsymbol{Q}}^j$ by

$$(\widetilde{\boldsymbol{Q}}^j)_{t,k} = \begin{cases} (\boldsymbol{Q}^j)_{t,k}, & \text{if } \max\{t,k\} \le \sum_{\ell=1}^{j-1} d^\ell, \\ 0, & \text{otherwise.} \end{cases}$$

In other words, $\widetilde{\boldsymbol{Q}}^j$ corresponds to $\boldsymbol{Q}^j$ with all but its first $j-1$ blocks of rows and columns set to zero.

We use $\nabla^j f(\boldsymbol{x})$ to denote the subvector of the gradient $\nabla f(\boldsymbol{x})$ indexed by the elements of $\mathcal{S}^j$. For a block-separable convex function $r(\boldsymbol{x}) = \sum_{j=1}^m r^j(\boldsymbol{x}^j)$, we use $r'(\boldsymbol{x})$ and $r^{j,'}(\boldsymbol{x}^j)$ to denote the elements in the subdifferential sets $\partial r(\boldsymbol{x})$ and $\partial r^j(\boldsymbol{x}^j)$ for $j \in [m]$, respectively.

Throughout the paper, we make use of the following assumptions. The first assumption is standard and rules out degenerate problem instances.

Assumption 2.1. $F(\boldsymbol{x})$ is bounded below and $\boldsymbol{x}^*$ is a global minimum of F.

The following two assumptions are the gradient Lipschitz conditions used in the analysis of our algorithms. These conditions are not standard, due to the choice of weighted norms $\|\cdot\|_{\boldsymbol{\Lambda}_j}, \|\cdot\|_{\boldsymbol{\Lambda}_j^{-1}}$, and $\|\cdot\|_{\boldsymbol{Q}^j}$. Assumption 2.3 is inspired by a similar Lipschitz condition introduced by Song & Diakonikolas (2021), the main difference with that paper being to use a more general norm $\|\cdot\|_{\boldsymbol{\Lambda}_j^{-1}}$ for the gradients.

Assumption 2.2. For all $\boldsymbol{x}$ and $\boldsymbol{y}$ that differ only in the j^{th} block, where $j \in [m]$, $f(\cdot)$ satisfies the following:

$$\|\nabla^j f(\boldsymbol{x}) - \nabla^j f(\boldsymbol{y})\|_{\boldsymbol{\Lambda}_j^{-1}} \le \|\boldsymbol{x}^j - \boldsymbol{y}^j\|_{\boldsymbol{\Lambda}_j}, \tag{4}$$

where $\boldsymbol{\Lambda}_j \in \mathbb{R}^{d_j \times d_j}$ is a positive definite diagonal matrix.

Observe that when $\boldsymbol{\Lambda}_j = L_j \boldsymbol{I}_{d^j}$, Assumption 2.2 becomes the standard block Lipschitz condition (Nesterov, 2012).

Assumption 2.3. There exist symmetric positive semidefinite $d \times d$ matrices $\boldsymbol{Q}^j$, $1 \le j \le m$, such that each $\nabla^j f(\cdot)$ is 1-Lipschitz continuous w.r.t. the seminorm $\|\cdot\|_{\boldsymbol{Q}^j}$. That is, $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, we have

$$\|\nabla^j f(\boldsymbol{x}) - \nabla^j f(\boldsymbol{y})\|^2_{\boldsymbol{\Lambda}_j^{-1}} \le \|\boldsymbol{x} - \boldsymbol{y}\|^2_{\boldsymbol{Q}^j}. \tag{5}$$

We remark that matrices $\boldsymbol{Q}^j$ do *not* need to be known to the algorithm. Observe that if f is L-smooth w.r.t. the Euclidean norm, i.e., if $\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\| \le L\|\boldsymbol{x} - \boldsymbol{y}\|$, then Assumption 2.3 can be satisfied with $\boldsymbol{\Lambda}_j = L\boldsymbol{I}_{d^j}$ and $\boldsymbol{Q}^j = L\boldsymbol{I}_d$ for $j \in [m]$. Indeed, in this case we have, $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, $\|\nabla^j f(\boldsymbol{x}) - \nabla^j f(\boldsymbol{y})\|^2_{\boldsymbol{\Lambda}_j^{-1}} \le \frac{1}{L}\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\|^2 \le L\|\boldsymbol{x} - \boldsymbol{y}\|^2 = \|\boldsymbol{x} - \boldsymbol{y}\|^2_{\boldsymbol{Q}^j}$. However, the more general matrices $\boldsymbol{\Lambda}_j$ and $\boldsymbol{Q}^j$ in Assumptions 2.2 and 2.3 provide more flexibility in exploiting the problem geometry, as shown and tested in our numerical experiments in Section 5. For further discussion and comparison to Euclidean Lipschitz constants, see Song & Diakonikolas (2021).

In the following, we let $\boldsymbol{\Lambda}$ be the diagonal matrix composed of positive diagonal blocks $\boldsymbol{\Lambda}_j$ for $j \in [m]$, i.e., $\boldsymbol{\Lambda} = \operatorname{diag}(\boldsymbol{\Lambda}_1, \boldsymbol{\Lambda}_2, \ldots, \boldsymbol{\Lambda}_m)$. We further provide the appropriate assumption about the PŁ-condition (Polyak, 1963; Kurdyka, 1998) w.r.t. the norm $\|\cdot\|_{\boldsymbol{\Lambda}}$ as follows, which is standard in nonconvex optimization (Attouch et al., 2010; Bolte et al., 2014; Frankel et al., 2015; Karimi et al., 2016; Li et al., 2021). This assumption is used only when proving linear convergence of our algorithms, not throughout the paper. The constant in this assumption need not be known.

Assumption 2.4. We say that F satisfies the PŁ condition w.r.t. $\|\cdot\|_{\boldsymbol{\Lambda}}$ with parameter $\mu > 0$, if for all $\boldsymbol{x} \in \mathbb{R}^d$,

$$\operatorname{dist}^2(\partial F(\boldsymbol{x}), \boldsymbol{0}) \ge 2\mu\big(F(\boldsymbol{x}) - F(\boldsymbol{x}^*)\big), \tag{6}$$

with $\operatorname{dist}^2(\partial F(\boldsymbol{x}), \boldsymbol{0}) := \inf_{r'(\boldsymbol{x}) \in \partial r(\boldsymbol{x})} \|\nabla f(\boldsymbol{x}) + r'(\boldsymbol{x})\|^2_{\boldsymbol{\Lambda}^{-1}}$.

Stochastic Settings. In the stochastic setting of our problem, we consider the finite sum nonconvex optimization problem described by (2). Our analysis also handles the case of stochastic optimization problems of the form (3) by taking $n \to \infty$. To avoid using separate notation for the two settings (finite and infinite sum), we state the assumptions and the results for problems (2) and treat (3) as the limiting case of (2) when $n \to \infty$.

Assumption 2.5. For any $\boldsymbol{x} \in \mathbb{R}^d$,

$$\mathbb{E}_i\big[\|\nabla f_i(\boldsymbol{x}) - \nabla f(\boldsymbol{x})\|^2_{\boldsymbol{\Lambda}^{-1}}\big] \le \sigma^2, \tag{7}$$

where i is drawn uniformly at random from $[n]$.

In the following, we assume a "two-point" oracle in which one can query the stochastic gradient at two points with the same random seed $i \in [n]$. Such an oracle assumption has been used extensively in prior work on variance reduction; see e.g., Fang et al. (2018); Zhou et al. (2018b); Wang et al. (2019); Cutkosky & Orabona (2019); Li et al. (2021).

Assumption 2.6. For all $\boldsymbol{x}$ and $\boldsymbol{y}$ that only differ in the j^{th} block for $j \in [m]$,

$$\mathbb{E}_i\big[\|\nabla^j f_i(\boldsymbol{x}) - \nabla^j f_i(\boldsymbol{y})\|_{\boldsymbol{\Lambda}_j^{-1}}\big] \le \|\boldsymbol{x}^j - \boldsymbol{y}^j\|_{\boldsymbol{\Lambda}_j}, \tag{8}$$

where i is drawn uniformly at random from $[n]$ and $\boldsymbol{\Lambda}_j \in \mathbb{R}^{d_j \times d_j}$ is a positive definite diagonal matrix.

Assumption 2.6 implies Assumption 2.2, due to the finite-sum assumption and uniform sampling. For simplicity, we use the same matrix $\mathbf{\Lambda}_j$ for both smoothness conditions.

Assumption 2.7. There exist positive semidefinite matrices $\boldsymbol{Q}^j$, $1 \le j \le m$ such that each $\nabla^j f$ is expected 1-Lipschitz continuous w.r.t. the seminorm $\|\cdot\|_{\boldsymbol{Q}^j}$, i.e., $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$,

$$\mathbb{E}_i\big[\|\nabla^j f_i(\boldsymbol{x}) - \nabla^j f_i(\boldsymbol{y})\|^2_{\mathbf{\Lambda}_j^{-1}}\big] \le \|\boldsymbol{x} - \boldsymbol{y}\|^2_{\boldsymbol{Q}^j}, \tag{9}$$

where i is drawn uniformly at random from $[n]$.

Similarly, Assumption 2.7 implies Assumption 2.3, so we use the same matrix $\boldsymbol{Q}^j$ for both cases. Finally, we introduce a useful result on variance bound from Zheng & Kwok (2016) for our later convergence analysis, with the proof provided in Appendix A for completeness.

Lemma 2.8. *Let $\mathcal{B}$ be the set of $|\mathcal{B}| = b$ samples from $[n]$, drawn without replacement and uniformly at random. Then, $\forall \boldsymbol{x} \in \mathbb{R}^d$ and $j \in [m]$,*

$$\begin{aligned}&\mathbb{E}_{\mathcal{B}}\Big[\Big\|\frac{1}{b}\sum_{i\in\mathcal{B}} \nabla^j f_i(\boldsymbol{x}) - \nabla^j f(\boldsymbol{x})\Big\|^2\Big] \\ &= \frac{n-b}{b(n-1)}\mathbb{E}_i\big[\|\nabla^j f_i(\boldsymbol{x}) - \nabla^j f(\boldsymbol{x})\|^2\big].\end{aligned} \tag{10}$$

3. P-CCD

As a warmup, in this section we provide a novel analysis of the standard Proximal Cyclic Block Coordinate Descent (P-CCD) algorithm (Algorithm 1) for the deterministic setting, adapted to our choice of block norms $\|\cdot\|_{\mathbf{\Lambda}_j}$. P-CCD cycles through the m blocks of variables, updating one block at a time. When $m = 1$, P-CCD is the standard proximal gradient method, while when $m = d$, P-CCD is a proximal version of cyclic coordinate descent.

Algorithm 1 Proximal Cyclic Block Coordinate Descent (P-CCD)

Input: $m, K, \boldsymbol{x}_0, \mathbf{\Lambda}_1, \mathbf{\Lambda}_2, \ldots, \mathbf{\Lambda}_m$
for $k = 1$ to K **do**
  for $j = 1$ to m **do**
    $\boldsymbol{x}_{k-1,j} = (\boldsymbol{x}_k^1, \ldots, \boldsymbol{x}_k^{j-1}, \boldsymbol{x}_{k-1}^j, \ldots, \boldsymbol{x}_{k-1}^m)$
    $\boldsymbol{x}_k^j = \arg\min_{\boldsymbol{x}^j \in \mathbb{R}^{d_j}} \Big\{\langle \nabla^j f(\boldsymbol{x}_{k-1,j}), \boldsymbol{x}^j\rangle + \frac{1}{2}\|\boldsymbol{x}^j - \boldsymbol{x}_{k-1}^j\|^2_{\mathbf{\Lambda}_j} + r^j(\boldsymbol{x}^j)\Big\}$
  end for
end for
return $\arg\min_{\boldsymbol{x}_k} \|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|_{\mathbf{\Lambda}}$ $(k \in [K])$

To analyze the convergence of Algorithm 1, we first define

$$\hat{L} := \Big\|\mathbf{\Lambda}^{-1/2}\Big(\sum_{j=1}^m \hat{\boldsymbol{Q}}^j\Big)\mathbf{\Lambda}^{-1/2}\Big\|,$$

to simplify the notation. This constant appears in the analysis but is not used by the algorithm.

The analysis is built on two key lemmas. Lemma 3.1 bounds the norm of the gradient by the distance between successive iterates, using the generalized Lipschitz condition w.r.t. a Mahalanobis norm, as stated in Assumption 2.3. Lemma 3.2 then bounds the sum of successive squared distances between iterates by the initial optimality gap, similar to a result that is typically proved for the (full-gradient) proximal method. Jointly, these two lemmas lead to a guarantee of a proximal method, but with the generalized Lipschitz constant $\hat{L}$ replacing the traditional full-gradient Lipschitz constant encountered in full-gradient methods.

Lemma 3.1. *Under Assumption 2.3, the iterates $\{\boldsymbol{x}_k\}$ generated by Algorithm 1 satisfy*

$$\operatorname{dist}^2(\partial F(\boldsymbol{x}_k), \mathbf{0}) \le 2(\hat{L} + 1)\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|^2_{\mathbf{\Lambda}}.$$

To bound $\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|^2_{\mathbf{\Lambda}}$, we prove the following descent lemma induced by the block-wise smoothness in Assumption 2.2 and by telescoping cyclically over the blocks.

Lemma 3.2. *Under Assumptions 2.2 and 2.3, the iterates $\{\boldsymbol{x}_k\}$ generated by Algorithm 1 satisfy*

$$\sum_{i=1}^k \|\boldsymbol{x}_i - \boldsymbol{x}_{i-1}\|^2_{\mathbf{\Lambda}} \le 2(F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)). \tag{11}$$

Proofs of Lemmas 3.1 and 3.2 are deferred to Appendix B. The next result describes the convergence of Algorithm 1.

Theorem 3.3. *Under Assumptions 2.2 and 2.3, let $\boldsymbol{x}^*$ be a global minimizer of (1) and $\{\boldsymbol{x}_k\}$ be the iterates generated by Algorithm 1. Then after K (outer-loop) iterations, we have*

$$\min_{k\in[K]} \operatorname{dist}^2(\partial F(\boldsymbol{x}_k), \mathbf{0}) \le \frac{4(\hat{L} + 1)(F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*))}{K}.$$

Proof. By combining Lemmas 3.1 and 3.2, we have

$$\begin{aligned}\sum_{k=1}^K \operatorname{dist}^2(\partial F(\boldsymbol{x}), \mathbf{0}) &\le 2(\hat{L} + 1)\sum_{k=1}^K \|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|^2_{\mathbf{\Lambda}} \\ &\le 4(\hat{L} + 1)(F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)).\end{aligned}$$

It remains to use that for all $k \in [K]$, we have $\operatorname{dist}^2(\partial F(\boldsymbol{x}_k), \mathbf{0}) \ge \min_{k'\in[K]} \operatorname{dist}^2(\partial F(\boldsymbol{x}_{k'}), \mathbf{0})$. □

In comparison with guarantees with respect to Euclidean Lipschitz constants, this guarantee is never worse than by a factor m. In the case in which f is L-smooth $\mathbf{\Lambda} = \boldsymbol{Q}^j = L\boldsymbol{I}_d$ (the worst case), we have $\inf_{r'(\mathbf{x})\in\partial r(\mathbf{x})}\|\nabla f(\mathbf{x}) + r'(\mathbf{x})\|^2 = L\operatorname{dist}^2(\partial F(\mathbf{x}), \mathbf{0})$ and

$\hat{L} = \frac{1}{L}\|\sum_{j=1}^m \hat{\boldsymbol{Q}}^j\| \le \frac{1}{L}\|\sum_{j=1}^m \boldsymbol{Q}^j\| \le m$, while in practice possibly $\|\sum_{j=1}^m \hat{\boldsymbol{Q}}^j\| \ll \|\sum_{j=1}^m \boldsymbol{Q}^j\|$ and $\hat{L} \ll m$ (see discussions in e.g., Song & Diakonikolas (2021)). The same points hold for guarantees in Section 4 as well. Note that the linear dependence on m cannot be improved in the worst case for standard P-CCD, even on smooth convex problems (Sun & Ye, 2021; Kamri et al., 2022).

If F further satisfies the PŁ condition of Assumption 2.4, Algorithm 1 can achieve a faster, linear convergence rate. We summarize this result in Corollary 3.4 below, deferring the proof to Appendix B.

Corollary 3.4. *Suppose that the conditions of Theorem 3.3 hold and that F further satisfies Assumption 2.4. Then we have after K iterations of Algorithm 1 that*

$$F(\boldsymbol{x}_K) - F(\boldsymbol{x}^*) \le \Big(\frac{2(\hat{L}+1)}{2(\hat{L}+1)+\mu}\Big)^K (F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)).$$

The main bottleneck in implementing P-CCD is in finding appropriate matrices $\boldsymbol{\Lambda}_j$ that satisfy Assumption 2.2. The simplest approach is to use $\boldsymbol{\Lambda}_j = L_j \boldsymbol{I}_{d_j}$ and estimate L_j adaptively using the standard backtracking line search. This procedure can be implemented efficiently, as the analysis requires Assumption 2.2 to hold only between successive iterates. The use of more general diagonal matrices $\boldsymbol{\Lambda}_j$ is a form of block preconditioning, which is frequently used to heuristically improve the performance of full-gradient methods. In our neural net training experiments, for example, we use spectral normalization (see Section 5 for more details).

4. Variance Reduced P-CCD

We now consider nonconvex optimization problems of the form (2). When n is finite, (2) is a finite-sum problem and we can compute the full gradient of $F(\boldsymbol{x})$ with $O(nd)$ cost. Without loss of generality, we use $n = +\infty$ to denote the general stochastic optimization setting as in (3), where the full gradient can no longer be computed in finite time. For both settings, Algorithm 2 describes VR-CCD, the Variance-Reduced Cyclic block Coordinate Descent algorithm, which combines Algorithm 1 with recursive variance reduction of PAGE type (Li et al., 2021) to reduce the per-iteration cost and improve the overall complexity.

Instead of computing the block-wise gradient at each inner iteration as P-CCD (Algorithm 1), VR-CCD maintains and updates a recursive gradient estimator g_{k-1}^j of PAGE type for each block gradient j at outer iteration k (i.e., g_{k-1}^j estimates $\nabla^j f(\boldsymbol{x}_{k-1,j})$). By the definition of g_{k-1}^j in Line 6 of Algorithm 2, it uses a minibatch estimate $\frac{1}{b}\sum_{i\in\mathcal{B}} \nabla^j f_i(\boldsymbol{x}_{k-1,j})$ with probability p, where $|\mathcal{B}| = b$. With probability $1-p$, the estimate $\boldsymbol{g}_{k-2}^j + \frac{1}{b'}\sum_{i\in\mathcal{B}'}(\nabla^j f_i(\boldsymbol{x}_{k-1,j}) - \nabla^j f_i(\boldsymbol{x}_{k-2,j}))$ reuses the previous j^{th} block gradient estimator g_{k-2}^j, and forms an approximation of the gradient difference $\nabla^j f(\boldsymbol{x}_{k-1,j}) - \nabla^j f(\boldsymbol{x}_{k-2,j})$ based on the minibatch $\mathcal{B}'$ where $|\mathcal{B}'| = b'$. When $p = 1$, the PAGE estimator reduces to vanilla minibatch SGD. To lower the computational cost, it is common to take $b' \ll b$ and $p \ll 1$. The estimator g_{k-1}^j is then incorporated into the Lipschitz gradient surrogate function in Line 7 to compute the new iterate $\boldsymbol{x}_k^j$. The variance of PAGE estimator w.r.t. block coordinates can be bounded recursively as in Lemma 4.1 below, using the minibatch variance bound results in Lemma 2.8. The proof appears in Appendix C.

To simplify the notation, we use the following definitions in the statements and proofs for this section, for $k \ge 0$:

$$\begin{aligned}
\tilde{L} &:= \big\|\boldsymbol{\Lambda}^{-1/2}\big(\sum_{j=1}^m \tilde{\boldsymbol{Q}}^j\big)\boldsymbol{\Lambda}^{-1/2}\big\|,\\
u_k &:= \sum_{j=1}^m \|\boldsymbol{g}_{k-1}^j - \nabla^j f(\boldsymbol{x}_{k-1,j})\|^2_{\boldsymbol{\Lambda}_j^{-1}},\\
v_k &:= \|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|^2_{\boldsymbol{\Lambda}},\\
s_k &:= \operatorname{dist}^2(\partial F(\boldsymbol{x}_k), \boldsymbol{0}).
\end{aligned}$$

Lemma 4.1. *Suppose Assumptions 2.5–2.7 hold, then the variance $\mathbb{E}[u_k]$ of the gradient estimators $\{\boldsymbol{g}_{k-1}^j\}_{j=1}^m$ at iteration k of Algorithm 2 is bounded by:*

$$\begin{aligned}
\mathbb{E}[u_k] \le \frac{2p(n-b)\sigma^2}{b(n-1)} + 2\Big(\frac{p(n-b)}{b(n-1)} + \frac{1-p}{b'}\Big)\tilde{L}\mathbb{E}[v_k]\\
+ (1-p)\mathbb{E}[u_{k-1}] + \frac{2(1-p)\hat{L}}{b'}\mathbb{E}[v_{k-1}].
\end{aligned}$$

The general strategy to analyze the convergence of VR-CCD can be summarized as follows. Let

$$\Phi_k = a_k\mathbb{E}[F(\boldsymbol{x}_k)] + b_k\mathbb{E}[u_k] + c_k\mathbb{E}[v_k]$$

be a potential function, where $\{a_k\}_{k\ge1}, \{b_k\}_{k\ge1}, \{c_k\}_{k\ge1}$ are non-negative sequences to be specified later in the analysis. Our goal is to show that

$$\mathbb{E}[s_k] \le \Phi_{k-1} - \Phi_k + \mathcal{E}_k, \tag{12}$$

where $\mathcal{E}_k$ are error terms arising from the noise of the estimator. Then, by telescoping (12) and controlling the error sequence $\mathcal{E}_k$, we obtain the gradient norm guarantee of Algorithm 2. First, we make use of the following descent lemma that utilizes block-wise smoothness from Assumption 2.6. Its proof is deferred to Appendix C.

Lemma 4.2. *Let Assumption 2.6 hold and let $r^{j,'}(\boldsymbol{x}_k^j) \in \partial r^j(\boldsymbol{x}_k^j)$ be such that $\boldsymbol{x}_k^j = \boldsymbol{x}_{k-1}^j - \eta\boldsymbol{\Lambda}_j^{-1}(\boldsymbol{g}_{k-1}^j + r^{j,'}(\boldsymbol{x}_k^j))$.*

Algorithm 2 Variance-Reduced Cyclic Block Coordinate Descent (VR-CCD)

Input: m, η, K, p, b, b', $\mathbf{\Lambda}_1, \mathbf{\Lambda}_2, \ldots, \mathbf{\Lambda}_m$, $\boldsymbol{x}_0 = \boldsymbol{x}_{-1} = \boldsymbol{x}_{-1,1} = \cdots = \boldsymbol{x}_{-1,m+1}$.
$\boldsymbol{g}_{-1} = \frac{1}{b}\sum_{i\in\mathcal{B}} \nabla f_i(\boldsymbol{x}_0)$.
for $k = 1$ to K **do**
  for $j = 1$ to m **do**
    $\boldsymbol{x}_{k-1,j} = (\boldsymbol{x}_k^1, \ldots, \boldsymbol{x}_k^{j-1}, \boldsymbol{x}_{k-1}^j, \ldots \boldsymbol{x}_{k-1}^m)$
    $\boldsymbol{g}_{k-1}^j = \begin{cases} \frac{1}{b}\sum_{i\in\mathcal{B}} \nabla^j f_i(\boldsymbol{x}_{k-1,j}), \text{ with probability } p \\ \boldsymbol{g}_{k-2}^j + \frac{1}{b'}\sum_{i\in\mathcal{B}'}(\nabla^j f_i(\boldsymbol{x}_{k-1,j}) - \nabla^j f_i(\boldsymbol{x}_{k-2,j})), \text{ with probability } 1-p \end{cases}$
    $\boldsymbol{x}_k^j = \arg\min_{\boldsymbol{x}^j\in\mathbb{R}^d} \left\{ \langle \boldsymbol{g}_{k-1}^j, \boldsymbol{x}^j\rangle + r^j(\boldsymbol{x}^j) + \frac{1}{2\eta}\|\boldsymbol{x}^j - \boldsymbol{x}_{k-1}^j\|_{\mathbf{\Lambda}_j}^2 \right\}$
  end for
end for
return $\hat{\boldsymbol{x}}_K$ uniformly drawn from $\{\boldsymbol{x}_k\}_{k\in[K]}$

Then the iterates of Algorithm 2 satisfy

$$F(\boldsymbol{x}_k) \le F(\boldsymbol{x}_{k-1}) - \frac{1-\eta}{2\eta}v_k + \frac{\eta}{2}u_k - \frac{\eta}{2}\sum_{j=1}^m \|\nabla^j f(\boldsymbol{x}_{k-1,j}) + r^{j,'}(\boldsymbol{x}_k^j)\|_{\mathbf{\Lambda}_j^{-1}}^2. \tag{13}$$

In the statement of Lemma 4.2, there must exist $r^{j,'}(\boldsymbol{x}_k^j) \in \partial r^j(\boldsymbol{x}_k^j)$ such that $\boldsymbol{x}_k^j = \boldsymbol{x}_{k-1}^j - \eta\mathbf{\Lambda}_j^{-1}(\boldsymbol{g}_{k-1}^j + r^{j,'}(\boldsymbol{x}_k^j))$, due to the first-order optimality condition of the minimization problem that defines $\boldsymbol{x}_k^j$.

We further bound the gradient norms of intermediate iterates within a cycle in Inequality (13), i.e., $\sum_{j=1}^m \|\nabla^j f(\boldsymbol{x}_{k-1,j}) + r^{j,'}(\boldsymbol{x}_k^j)\|_{\mathbf{\Lambda}_j^{-1}}^2$, using smoothness from Assumption 2.7.

Lemma 4.3. *Let Assumption 2.7 hold and let $r^{j,'}(\boldsymbol{x}_k^j) \in \partial r^j(\boldsymbol{x}_k^j)$ be such that $\boldsymbol{x}_k^j = \boldsymbol{x}_{k-1}^j - \eta\mathbf{\Lambda}_j^{-1}(\boldsymbol{g}_{k-1}^j + r^{j,'}(\boldsymbol{x}_k^j))$. Then for Algorithm 2 we have*

$$s_k \le 2\hat{L}v_k + 2\sum_{j=1}^m \|\nabla^j f(\boldsymbol{x}_{k-1,j}) + r^{j,'}(\boldsymbol{x}_k)\|_{\mathbf{\Lambda}_j^{-1}}^2. \tag{14}$$

By combining Lemma 4.2 and 4.3 and using the recursive variance bound of the estimator from Lemma 4.1, we are ready to prove a bound on iteration complexity for VR-CCD in Theorem 4.4. The proof is in Appendix C.

Theorem 4.4. *Suppose that Assumptions 2.2–2.3 and 2.5–2.7 hold. Let $\boldsymbol{x}^*$ be a global minimizer of (1) and $\{\boldsymbol{x}_k\}$ be the iterates generated by Algorithm 2. Then, we have*

$$\mathbb{E}\big[\mathrm{dist}^2(\partial F(\hat{\boldsymbol{x}}_K), \mathbf{0})\big] \le \frac{4\Delta_0}{\eta K} + \frac{2(1-p)(n-b)\sigma^2}{pb(n-1)K} + \frac{4(n-b)\sigma^2}{b(n-1)}, \tag{15}$$

where $\Delta_0 = F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)$, and $0 < \eta \le \frac{-1+\sqrt{1+4c_0}}{2c_0}$ with $c_0 = \frac{2(1-p)\hat{L}}{pb'} + \hat{L} + 2\big(\frac{p(n-b)}{b(n-1)} + \frac{1-p}{b'}\big)\frac{\tilde{L}}{p}$.

Note that Theorem 4.4 is generic for both finite-sum and infinite-sum cases. We summarize its implications for both problems in the following corollaries, for specific parameters of Algorithm 2. In the remaining results of this section, we assume that the assumptions of Theorem 4.4 hold. The proofs are provided in Appendix C for completeness.

Corollary 4.5 (Finite-sum)**.** *Choosing $b = n$, $b' = \sqrt{n}$, and $p = \frac{b'}{b+b'}$, and setting $K = \frac{4\Delta_0}{\epsilon^2\eta}$, we have $\mathbb{E}\big[\mathrm{dist}^2(\partial F(\hat{\boldsymbol{x}}_K), \mathbf{0})\big] \le \epsilon^2$ with $\mathcal{O}\big(nd + \frac{\Delta_0 d\sqrt{n(\hat{L}+\tilde{L})}}{\epsilon^2}\big)$ arithmetic operations, where $\Delta_0 = F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)$.*

Corollary 4.6 (Infinite-sum)**.** *Choosing $b = \lceil\frac{12\sigma^2}{\epsilon^2}\rceil$, $b' = \sqrt{b}$, and $p = \frac{b'}{b+b'}$, and setting $K = \frac{12\Delta_0}{\epsilon^2\eta} + \frac{1}{2p}$, we have $\mathbb{E}\big[\mathrm{dist}^2(\partial F(\hat{\boldsymbol{x}}_K), \mathbf{0})\big] \le \epsilon^2$ with $\mathcal{O}\big(bd + \frac{\Delta_0 d\sqrt{b(\hat{L}+\tilde{L})}}{\epsilon^2}\big)$ arithmetic operations, where $\Delta_0 = F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)$.*

Under the PŁ condition, a faster convergence rate can be proved for VR-CCD, as we show next.

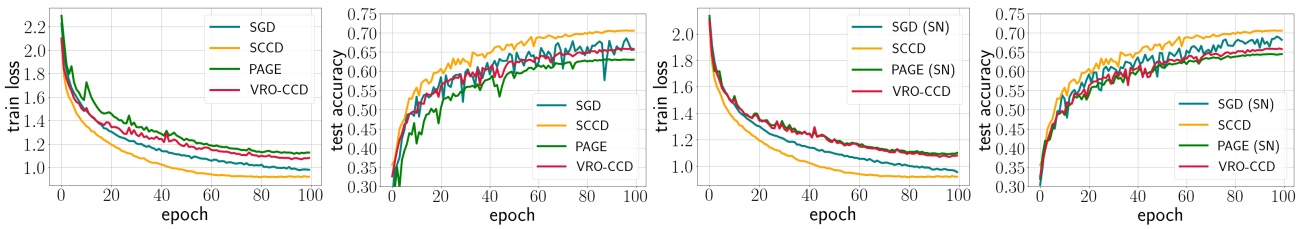
Corollary 4.7. *Suppose that F further satisfies Assumption 2.4, then we have for Algorithm 2,*

$$\mathbb{E}[F(\boldsymbol{x}_K) - F(\boldsymbol{x}^*)] \le \big(1 + \frac{\eta\mu}{2}\big)^{-K}\Big(\Delta_0 + \frac{\sigma^2\eta(1-p)(n-b)}{pb(n-1)}\Big) + \frac{4(n-b)\sigma^2}{b\mu(n-1)},$$

where $0 < \eta \le \min\big\{\frac{p}{\mu(1-p)}, \frac{-1+\sqrt{1+4c_0}}{2c_0}\big\}$ with $c_0 = \hat{L} + \frac{4\hat{L}}{pb'} + \frac{4\tilde{L}}{p}\big(\frac{p(n-b)}{b(n-1)} + \frac{1-p}{b'}\big)$, and $\Delta_0 = F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)$.

In the following, we further provide arithmetic operation complexity results under PŁ condition with specifying the parameters of the algorithm for both finite-sum and infinite-sum problems.

Corollary 4.8 (Finite-sum)**.** *Choosing $b = n$, $b' = \sqrt{n}$, and $p = \frac{b'}{b+b'}$, and setting $K = (1 + \frac{2}{\eta\mu})\log(\frac{\Delta_0}{\epsilon})$, we have $\mathbb{E}[F(\boldsymbol{x}_K) - F(\boldsymbol{x}^*)] \le \epsilon$ with $\mathcal{O}\Big(nd + \big(\frac{\sqrt{n(\hat{L}+\tilde{L})}}{\mu} +$*



(a) Train Loss (b) Test Accuracy (c) Train Loss (SN) (d) Test Accuracy (SN)

Figure 1. Comparison of SGD, SCCD, PAGE and VRO-CCD on training LeNet on CIFAR-10.

$n\big)d\log(\frac{\Delta_0}{\epsilon})\Big)$ *arithmetic operations, where* $\Delta_0 = F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)$.

Corollary 4.9 (Infinite-sum). *Choosing* $b = \lceil \frac{12\sigma^2}{\mu\epsilon} \rceil$, $b' = \sqrt{b}$, *and* $p = \frac{b'}{b+b'}$, *and setting* $K = (1 + \frac{2}{\eta\mu})\log(\frac{3\Delta_0}{\epsilon})$, *we have* $\mathbb{E}[F(\boldsymbol{x}_K) - F(\boldsymbol{x}^*)] \leq \epsilon$ *with* $\mathcal{O}\Big(bd + \big(\frac{\sqrt{b(\hat{L}+\tilde{L})}}{\mu} + b\big)d\log(\frac{\Delta_0}{\epsilon})\Big)$ *arithmetic operations, where* $\Delta_0 = F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)$.

A few remarks are in order here. In addition to requiring matrices $\boldsymbol{\Lambda}_j$, VR-CCD also requires constants $\hat{L}$ and $\tilde{L}$ to set the learning rate, but these constants are often not readily available in practice. Instead, one can tune the value of the learning rate η, as is frequently done in practice for other optimization methods. Additionally, our methods require a fresh sample for each block in the cyclic update — an undesirable feature, because the sample complexity increases with the number of blocks. However, this requirement appears only to be an artifact of the analysis. In practice, we can implement a variant of VR-CCD (VRO-CCD) which re-uses the same sample for all the blocks. Interestingly, this variant shows even better empirical performance than VR-CCD (see Fig. 4 in Appendix D). Analysis of this variant is beyond the scope of this paper and is an interesting direction for future research.

5. Numerical Experiments and Discussion

We now describe numerical experiments on training neural networks to evaluate the cyclic block coordinate update scheme. In particular, we train LeNet (LeCun et al., 1998) with weight decay for the image classification task on CIFAR-10 (Krizhevsky et al., 2009) for our experiments. Details of the architecture are provided in Appendix D. We implement VRO-CCD and its special case SCCD (obtained by setting $p = 1$), and compare them with SGD and PAGE (Li et al., 2021). Note that VRO-CCD and PAGE, SCCD and SGD differ only in whether the variables are cyclically updated or not, so provide a fair comparison to justify the efficacy of the cyclic scheme. We implement all the algorithms using PyTorch (Paszke et al., 2019), and run the experiments on Google Colab standard GPU backend.[1]

Table 1. Runtime (seconds) per iteration and per epoch.

ALGORITHM	RUNTIME (ITER)	RUNTIME (EPOCH)
SGD	0.172	16.9 ± 0.9
SCCD	0.185	18.1 ± 0.6
PAGE	0.017	20.7 ± 0.9
VRO-CCD	0.041	49.5 ± 2.3

For all algorithms, we set the mini-batch size b to be 512 and the weight decay parameter to be 0.0005. We repeat the experiments 3 times with 100 epochs and average the results. For the learning rate, we use the cosine learning rate scheduler (Loshchilov & Hutter, 2017), which is tuned separately for each method via a grid search. For VRO-CCD and PAGE methods, we set $b' = \sqrt{b}$ and $p = \frac{b'}{b+b'}$ according to the theoretical results. For VRO-CCD and SCCD, we split the neural network parameters with each layer as a block ($m = 5$), and estimate the $\boldsymbol{\Lambda}_j$ of fully connected layers by spectral norms of the weights using spectral normalization (SN) (Miyato et al., 2018). VRO-CCD and SCCD are also compared with SGD and PAGE with the same spectral normalization. We report and plot the train loss and test accuracy against the epoch numbers in Fig. 1, where one epoch corresponds to the number of arithmetic operations in one data pass, in the order $\mathcal{O}(Nd)$ where N is the size of the training set. We summarize the runtime of each algorithm per iteration[2] and per epoch in Table 1, which is averaged by the results of 100 epochs.

From Fig. 1, we observe that (i) SCCD and VRO-CCD with cyclic scheme exhibit faster convergence with better generalization than SGD and PAGE, respectively, in Figures 1(a) and 1(b); (ii) The edge of cyclic scheme is still

[1]Our code is available at https://github.com/zephyr-cai/NonconvexCCD.

[2]One iteration for cyclic methods refers to one cycle of block coordinate updates. Since PAGE switches between large and small batches, we only summarize the mean runtime per iteration here.

noticeable comparing to SGD and PAGE with the same spectral normalization in Figures 1(c) and 1(d). All of these validate the efficacy of the cyclic update scheme. Note that in this experiment SGD and SCCD can admit larger stepsize, thus showing faster convergence (see Fig. 5 in Appendix D for further numerical comparison with same stepsizes).

When using small batches, VRO-CCD is around 2.4 times slower than PAGE as in Table 1, because the cyclic update becomes the major computational bottleneck in each iteration. However, for large batches, SCCD sacrifices only a marginal 7.5% runtime per iteration in comparison with SGD. SCCD also converges fastest in terms of wall-clock time (see Fig. 3 in Appendix D, as one may worry about the accumulation of marginal time increase over epochs). Our conclusion is that the cyclic scheme can be an efficient and effective alternative for large-batch methods.

Acknowledgements

XC and CS acknowledge support from NSF DMS 2023239. SW acknowledges support from NSF Awards DMS 2023239 and CCF 2224213, DOE via subcontract 8F-30039 from Argonne National Laboratory, and AFOSR Award FA9550-21-1-0084. JD acknowledges support from the U.S. Office of Naval Research under contract number N000142212348 and from NSF Award CCF 2007757.

References

Alacaoglu, A., Tran Dinh, Q., Fercoq, O., and Cevher, V. Smooth primal-dual coordinate descent algorithms for nonsmooth convex optimization. In *Proc. NeurIPS'17*, 2017.

Allen-Zhu, Z. Katyusha: The first direct acceleration of stochastic gradient methods. *The Journal of Machine Learning Research*, 18(1):8194–8244, 2017.

Allen-Zhu, Z., Qu, Z., Richtárik, P., and Yuan, Y. Even faster accelerated coordinate descent using non-uniform sampling. In *Proc. ICML'16*, 2016.

Attouch, H., Bolte, J., Redont, P., and Soubeyran, A. Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the kurdyka-łojasiewicz inequality. *Mathematics of Operations Research*, 35(2):438–457, 2010.

Beck, A. and Tetruashvili, L. On the convergence of block coordinate descent type methods. *SIAM Journal on Optimization*, 23(4):2037–2060, 2013.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

Bolte, J., Sabach, S., and Teboulle, M. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, 146(1-2):459–494, 2014.

Chauhan, V. K., Dahiya, K., and Sharma, A. Mini-batch block-coordinate based stochastic average adjusted gradient methods to solve big data problems. In *Proc. ACML'16*, 2017.

Chen, J. and Gu, Q. Accelerated stochastic block coordinate gradient descent for sparsity constrained nonconvex optimization. In *Proc. UAI'16*, 2016.

Chen, Z., Li, Y., and Lu, J. On the global convergence of randomized coordinate gradient descent for non-convex optimization. *arXiv preprint arXiv:2101.01323*, 2021.

Chow, Y. T., Wu, T., and Yin, W. Cyclic coordinate-update algorithms for fixed-point problems: Analysis and applications. *SIAM Journal on Scientific Computing*, 39(4): A1280–A1300, 2017.

Cutkosky, A. and Orabona, F. Momentum-based variance reduction in non-convex SGD. In *Proc. NeurIPS'19*, 2019.

Defazio, A., Bach, F., and Lacoste-Julien, S. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Proc. NeurIPS'14*, 2014.

Diakonikolas, J. and Orecchia, L. Alternating randomized block coordinate descent. In *Proc. ICML'18*, 2018.

Fang, C., Li, C. J., Lin, Z., and Zhang, T. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. In *Proc. NeurIPS'18*, 2018.

Frankel, P., Garrigos, G., and Peypouquet, J. Splitting methods with variable metric for Kurdyka–Łojasiewicz functions and general convergence rates. *Journal of Optimization Theory and Applications*, 165:874–900, 2015.

Friedman, J., Hastie, T., and Tibshirani, R. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1, 2010.

Fu, X., Ibrahim, S., Wai, H.-T., Gao, C., and Huang, K. Block-randomized stochastic proximal gradient for low-rank tensor factorization. *IEEE Transactions on Signal Processing*, 68:2170–2185, 2020.

Gorbunov, E., Hanzely, F., and Richtárik, P. A unified theory of sgd: Variance reduction, sampling, quantization and coordinate descent. In *Proc. AISTATS'20*, 2020.

Grippof, L. and Sciandrone, M. Globally convergent block-coordinate techniques for unconstrained optimization. *Optimization Methods and Software*, 10(4):587–637, 1999.

Gurbuzbalaban, M., Ozdaglar, A., Parrilo, P. A., and Vanli, N. When cyclic coordinate descent outperforms randomized coordinate descent. In *Proc. NeurIPS'17*, 2017.

Hong, M., Wang, X., Razaviyayn, M., and Luo, Z.-Q. Iteration complexity analysis of block coordinate descent methods. *Mathematical Programming*, 163(1):85–114, 2017.

Johnson, R. and Zhang, T. Accelerating stochastic gradient descent using predictive variance reduction. In *Proc. NeurIPS'13*, 2013.

Kamri, Y., Hendrickx, J. M., and Glineur, F. On the worst-case analysis of cyclic coordinate-wise algorithms on smooth convex functions. *arXiv preprint arXiv:2211.17018*, 2022.

Karimi, H., Nutini, J., and Schmidt, M. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Proc. ECML PKDD'2016*, 2016.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. In *Proc. ICLR'17*, 2017.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Kurdyka, K. On gradients of functions definable in o-minimal structures. *Annales de l'institut Fourier*, 48 (3):769–783, 1998.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lee, C.-P. and Wright, S. J. Random permutations fix a worst case for cyclic coordinate descent. *IMA Journal of Numerical Analysis*, 39(3):1246–1275, 2019.

Lei, J. and Shanbhag, U. V. Asynchronous variance-reduced block schemes for composite non-convex stochastic optimization: block-specific steplengths and adapted batch-sizes. *Optimization Methods and Software*, pp. 1–31, 2020.

Lei, L., Ju, C., Chen, J., and Jordan, M. I. Nonconvex finite-sum optimization via SCSG methods. In *Proc. NeurIPS'17*, 2017.

Li, X., Zhao, T., Arora, R., Liu, H., and Hong, M. On faster convergence of cyclic block coordinate descent-type methods for strongly convex minimization. *The Journal of Machine Learning Research*, 18(1):6741–6764, 2017.

Li, Z., Bao, H., Zhang, X., and Richtárik, P. PAGE: A simple and optimal probabilistic gradient estimator for nonconvex optimization. In *Proc. ICML21*, 2021.

Lin, Q., Lu, Z., and Xiao, L. An accelerated randomized proximal coordinate gradient method and its application to regularized empirical risk minimization. *SIAM Journal on Optimization*, 25(4):2244–2273, 2015.

Loshchilov, I. and Hutter, F. SGDR: stochastic gradient descent with warm restarts. In *Proc. ICLR'17,*, 2017.

Mazumder, R., Friedman, J. H., and Hastie, T. Sparsenet: Coordinate descent with nonconvex penalties. *Journal of the American Statistical Association*, 106(495):1125–1138, 2011.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. In *Proc. ICLR'18*, 2018.

Nakamura, K., Soatto, S., and Hong, B.-W. Block-cyclic stochastic coordinate descent for deep neural networks. *Neural Networks*, 139:348–357, 2021.

Nesterov, Y. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

Nesterov, Y. and Stich, S. U. Efficiency of the accelerated coordinate descent method on structured optimization problems. *SIAM Journal on Optimization*, 27(1):110–123, 2017.

Nguyen, L. M., Liu, J., Scheinberg, K., and Takáč, M. Sarah: A novel method for machine learning problems using stochastic recursive gradient. In *Proc. ICML'17*, 2017.

Nie, F., Xue, J., Wu, D., Wang, R., Li, H., and Li, X. Coordinate descent method for k-means. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(5):2371–2385, 2021.

Nutini, J., Schmidt, M., Laradji, I., Friedlander, M., and Koepke, H. Coordinate descent converges faster with the Gauss-Southwell rule than random selection. In *Proc. ICML'15*, 2015.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. In *Proc. NeurIPS'19*, 2019.

Plummer, S., Pati, D., and Bhattacharya, A. Dynamics of coordinate ascent variational inference: A case study in 2d ising models. *Entropy*, 22(11):1263, 2020.

Polyak, B. Gradient methods for the minimisation of functionals. *Ussr Computational Mathematics and Mathematical Physics*, 3:864–878, 1963.

Razaviyayn, M., Hong, M., and Luo, Z.-Q. A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM Journal on Optimization*, 23(2):1126–1153, 2013.

Reddi, S. J., Hefny, A., Sra, S., Poczos, B., and Smola, A. Stochastic variance reduction for nonconvex optimization. In *Proc. ICML'16*, 2016.

Saha, A. and Tewari, A. On the nonasymptotic convergence of cyclic coordinate descent methods. *SIAM Journal on Optimization*, 23(1):576–601, 2013.

Schmidt, M., Le Roux, N., and Bach, F. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1):83–112, 2017.

Song, C. and Diakonikolas, J. Fast cyclic coordinate dual averaging with extrapolation for generalized variational inequalities. *arXiv preprint arXiv:2102.13244*, 2021.

Song, C., Jiang, Y., and Ma, Y. Variance reduction via accelerated dual averaging for finite-sum optimization. In *Proc. NeurIPS'20*, 2020.

Sun, R. and Ye, Y. Worst-case complexity of cyclic coordinate descent: $O(n^2)$ gap with randomized version. *Mathematical Programming*, 185(1):487–520, 2021.

Tseng, P. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109(3):475–494, 2001.

Vandaele, A., Gillis, N., Lei, Q., Zhong, K., and Dhillon, I. Efficient and non-convex coordinate descent for symmetric nonnegative matrix factorization. *IEEE Transactions on Signal Processing*, 64(21):5571–5584, 2016.

Wang, W., Wan, H., and Chang, K.-H. Randomized block coordinate descendant strong for large-scale stochastic optimization. In *Proc. WSC'16*, 2016.

Wang, Z., Ji, K., Zhou, Y., Liang, Y., and Tarokh, V. Spiderboost and momentum: Faster variance reduction algorithms. In *Proc. NeurIPS'19*, 2019.

Wright, S. and Lee, C.-p. Analyzing random permutations for cyclic coordinate descent. *Mathematics of Computation*, 89(325):2217–2248, 2020.

Wright, S. J. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

Wu, T. T. and Lange, K. Coordinate descent algorithms for lasso penalized regression. *The Annals of Applied Statistics*, 2(1):224–244, 2008.

Xu, Y. and Yin, W. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal on Imaging Sciences*, 6(3):1758–1789, 2013.

Xu, Y. and Yin, W. Block stochastic gradient iteration for convex and nonconvex optimization. *SIAM Journal on Optimization*, 25(3):1686–1716, 2015.

Xu, Y. and Yin, W. A globally convergent algorithm for nonconvex optimization based on block coordinate update. *Journal of Scientific Computing*, 72(2):700–734, 2017.

Zeng, J., Peng, Z., Lin, S., and Xu, Z. A cyclic coordinate descent algorithm for lq regularization. *arXiv preprint arXiv:1408.0578*, 2014.

Zeng, W.-J. and So, H.-C. Coordinate descent algorithms for phase retrieval. *Signal Processing*, 169:107418, 2020.

Zheng, S. and Kwok, J. T. Fast-and-light stochastic admm. In *Proc. IJCAI'16*, 2016.

Zhou, D., Xu, P., and Gu, Q. Finding local minima via stochastic nested variance reduction. *arXiv preprint arXiv:1806.08782*, 2018a.

Zhou, D., Xu, P., and Gu, Q. Stochastic nested variance reduction for nonconvex optimization. In *Proc. NeurIPS'18*, 2018b.

A. Omitted Proofs from Section 2

Lemma 2.8. *Let $\mathcal{B}$ be the set of $|\mathcal{B}| = b$ samples from $[n]$, drawn without replacement and uniformly at random. Then, $\forall \boldsymbol{x} \in \mathbb{R}^d$ and $j \in [m]$,*

$$\begin{aligned} &\mathbb{E}_{\mathcal{B}}\Big[\big\|\frac{1}{b}\sum_{i\in\mathcal{B}}\nabla^j f_i(\boldsymbol{x}) - \nabla^j f(\boldsymbol{x})\big\|^2\Big] \\ &= \frac{n-b}{b(n-1)}\mathbb{E}_i\big[\|\nabla^j f_i(\boldsymbol{x}) - \nabla^j f(\boldsymbol{x})\|^2\big]. \end{aligned} \tag{10}$$

Proof. Observe first by expanding the square that

$$\begin{aligned} &\mathbb{E}_{\mathcal{B}}\Big[\big\|\frac{1}{b}\sum_{i\in\mathcal{B}}\nabla^j f_i(\boldsymbol{x}) - \nabla^j f(\boldsymbol{x})\big\|^2\Big] \\ &= \frac{1}{b^2}\mathbb{E}_{\mathcal{B}}\Big[\sum_{i,i'\in\mathcal{B}}\big\langle\nabla^j f_i(\boldsymbol{x}) - \nabla^j f(\boldsymbol{x}), \nabla^j f_{i'}(\boldsymbol{x}) - \nabla^j f(\boldsymbol{x})\big\rangle\Big] \\ &= \frac{1}{b^2}\mathbb{E}_{\mathcal{B}}\Big[\sum_{i,i'\in\mathcal{B},i\neq i'}\big\langle\nabla^j f_i(\boldsymbol{x}) - \nabla^j f(\boldsymbol{x}), \nabla^j f_{i'}(\boldsymbol{x}) - \nabla^j f(\boldsymbol{x})\big\rangle\Big] + \frac{1}{b}\mathbb{E}_i\Big[\|\nabla^j f_i(\boldsymbol{x}) - \nabla^j f(\boldsymbol{x})\|^2\Big]. \end{aligned}$$

Since the batch $\mathcal{B}$ is drawn independently and uniformly from $[n]$, we know the probability that each pair (i, i') with $i \neq i'$ belongs to $\mathcal{B}$ can be given as $\frac{b(b-1)}{n(n-1)}$. Further, by the linearity of expectation, we have

$$\begin{aligned} &\mathbb{E}_{\mathcal{B}}\Big[\sum_{i,i'\in\mathcal{B},i\neq i'}\big\langle\nabla^j f_i(\boldsymbol{x}) - \nabla^j f(\boldsymbol{x}), \nabla^j f_{i'}(\boldsymbol{x}) - \nabla^j f(\boldsymbol{x})\big\rangle\Big] \\ &= \mathbb{E}_{\mathcal{B}}\Big[\sum_{i,i'\in[n],i\neq i'}\mathbb{1}_{i,i'\in\mathcal{B}}\big\langle\nabla^j f_i(\boldsymbol{x}) - \nabla^j f(\boldsymbol{x}), \nabla^j f_{i'}(\boldsymbol{x}) - \nabla^j f(\boldsymbol{x})\big\rangle\Big] \\ &= \sum_{i,i'\in[n],i\neq i'}\mathbb{E}_{\mathcal{B}}\Big[\mathbb{1}_{i,i'\in\mathcal{B}}\big\langle\nabla^j f_i(\boldsymbol{x}) - \nabla^j f(\boldsymbol{x}), \nabla^j f_{i'}(\boldsymbol{x}) - \nabla^j f(\boldsymbol{x})\big\rangle\Big] \\ &= \frac{b(b-1)}{n(n-1)}\sum_{i,i'\in[n],i\neq i'}\big\langle\nabla^j f_i(\boldsymbol{x}) - \nabla^j f(\boldsymbol{x}), \nabla^j f_{i'}(\boldsymbol{x}) - \nabla^j f(\boldsymbol{x})\big\rangle, \end{aligned}$$

where $\mathbb{1}$ is the indicator function with $\mathbb{1}_{i,i'\in\mathcal{B}} = 1$ if both $i, i' \in \mathcal{B}$ otherwise 0. Then we obtain

$$\begin{aligned} &\mathbb{E}_{\mathcal{B}}\Big[\big\|\frac{1}{b}\sum_{i\in\mathcal{B}}\nabla^j f_i(\boldsymbol{x}) - \nabla^j f(\boldsymbol{x})\big\|^2\Big] \\ &= \frac{b-1}{bn(n-1)}\sum_{i,i'\in[n],i\neq i'}\big\langle\nabla^j f_i(\boldsymbol{x}) - \nabla^j f(\boldsymbol{x}), \nabla^j f_{i'}(\boldsymbol{x}) - \nabla^j f(\boldsymbol{x})\big\rangle + \frac{1}{b}\mathbb{E}_i\Big[\|\nabla^j f_i(\boldsymbol{x}) - \nabla^j f(\boldsymbol{x})\|^2\Big] \\ &= \frac{b-1}{bn(n-1)}\sum_{i,i'\in[n]}\big\langle\nabla^j f_i(\boldsymbol{x}) - \nabla^j f(\boldsymbol{x}), \nabla^j f_{i'}(\boldsymbol{x}) - \nabla^j f(\boldsymbol{x})\big\rangle + \Big(\frac{1}{b} - \frac{b-1}{b(n-1)}\Big)\mathbb{E}_i\Big[\|\nabla^j f_i(\boldsymbol{x}) - \nabla^j f(\boldsymbol{x})\|^2\Big] \\ &\overset{(i)}{=} \frac{n-b}{b(n-1)}\mathbb{E}_i\Big[\|\nabla^j f_i(\boldsymbol{x}) - \nabla^j f(\boldsymbol{x})\|^2\Big], \end{aligned}$$

where (i) is due to the finite-sum structure of f, i.e. $f = \frac{1}{n}\sum_{i=1}^n f_i$. □

Remark A.1. The proof for Lemma 2.8 does not involve the specification of the norm, thus applying to $\|\cdot\|_{\boldsymbol{\Lambda}_j^{-1}}$ in the paper.

B. Omitted Proofs from Section 3

Lemma 3.1. *Under Assumption 2.3, the iterates $\{\boldsymbol{x}_k\}$ generated by Algorithm 1 satisfy*

$$\operatorname{dist}^2(\partial F(\boldsymbol{x}_k), \mathbf{0}) \leq 2(\hat{L}+1)\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|_{\boldsymbol{\Lambda}}^2.$$

Proof. By the definition of $\boldsymbol{x}_k^j$ (Step 5 in Algorithm 1), for each $j \in [m]$, we have that there exists $r^{j,'}(\boldsymbol{x}_k^j) \in \partial r^j(\boldsymbol{x}_k^j)$ such that $\boldsymbol{x}_k^j = \boldsymbol{x}_{k-1}^j - \boldsymbol{\Lambda}_j^{-1}(\nabla^j f(\boldsymbol{x}_{k-1}, j) + r^{j,'}(\boldsymbol{x}_k^j))$. Observe that, due to the block separability of r, the vector r' obtained by concatenating such vectors $r^{j,'}$, $j \in [m]$, is a subgradient vector for r. Hence, using block separability of r and the definitions of the matrix $\boldsymbol{\Lambda}$ and the norm $\|\cdot\|_{\boldsymbol{\Lambda}}$, we have

$$\begin{aligned}\|\nabla f(\boldsymbol{x}_k) + r'(\boldsymbol{x}_k)\|_{\boldsymbol{\Lambda}^{-1}}^2 &= \sum_{j=1}^m \|\nabla^j f(\boldsymbol{x}_k) + r^{j,'}(\boldsymbol{x}_k^j)\|_{\boldsymbol{\Lambda}_j^{-1}}^2 \\ &\le 2\sum_{j=1}^m \big(\|\nabla^j f(\boldsymbol{x}_k) - \nabla^j f(\boldsymbol{x}_{k-1,j})\|_{\boldsymbol{\Lambda}_j^{-1}}^2 + \|\nabla^j f(\boldsymbol{x}_{k-1,j}) + r^{j,'}(\boldsymbol{x}_k^j)\|_{\boldsymbol{\Lambda}_j^{-1}}^2\big) \\ &= 2\sum_{j=1}^m \big(\|\nabla^j f(\boldsymbol{x}_k) - \nabla^j f(\boldsymbol{x}_{k-1,j})\|_{\boldsymbol{\Lambda}_j^{-1}}^2 + \|\boldsymbol{\Lambda}_j(\boldsymbol{x}_k^j - \boldsymbol{x}_{k-1}^j)\|_{\boldsymbol{\Lambda}_j^{-1}}^2\big), \end{aligned} \tag{16}$$

where the inequality comes from adding and subtracting $\nabla^j f(\boldsymbol{x}_{k-1,j})$ inside the norm terms and using Young's inequality, while the last equality is by $\boldsymbol{x}_k^j = \boldsymbol{x}_{k-1}^j - \boldsymbol{\Lambda}_j^{-1}(\nabla^j f(\boldsymbol{x}_{k-1}, j) + r^{j,'}(\boldsymbol{x}_k^j))$.

By Assumption 2.3, we have

$$\|\nabla^j f(\boldsymbol{x}_k) - \nabla^j f(\boldsymbol{x}_{k-1,j})\|_{\boldsymbol{\Lambda}_j^{-1}}^2 \le (\boldsymbol{x}_k - \boldsymbol{x}_{k-1,j})^T \boldsymbol{Q}^j (\boldsymbol{x}_k - \boldsymbol{x}_{k-1,j}), \tag{17}$$

while using the definition of the norm $\|\cdot\|_{\boldsymbol{\Lambda}_j^{-1}}$, we have

$$\|\boldsymbol{\Lambda}_j(\boldsymbol{x}_k^j - \boldsymbol{x}_{k-1}^j)\|_{\boldsymbol{\Lambda}_j^{-1}}^2 = \|\boldsymbol{x}_k^j - \boldsymbol{x}_{k-1}^j\|_{\boldsymbol{\Lambda}_j}^2. \tag{18}$$

Observe further that $\boldsymbol{x}_{k-1,j}$ has the same elements as $\boldsymbol{x}_k$ in the first $j-1$ coordinates. Meanwhile, for $t \ge j$, let $(\boldsymbol{x}_{k-1,j})_t, (\boldsymbol{x}_{k-1})_t$ denote the t^{th} blocks of $\boldsymbol{x}_{k-1,j}$ and $\boldsymbol{x}_{k-1}$ respectively. Then we have $(\boldsymbol{x}_{k-1,j})_t = (\boldsymbol{x}_{k-1})_t$, thus $(\boldsymbol{x}_{k-1,j})_t - (\boldsymbol{x}_k)_t = (\boldsymbol{x}_{k-1})_t - (\boldsymbol{x}_k)_t$. Thus, based on the definition of $\hat{\boldsymbol{Q}}^j$, we have

$$(\boldsymbol{x}_k - \boldsymbol{x}_{k-1,j})^T \boldsymbol{Q}^j (\boldsymbol{x}_k - \boldsymbol{x}_{k-1,j}) = (\boldsymbol{x}_k - \boldsymbol{x}_{k-1})^T \hat{\boldsymbol{Q}}^j (\boldsymbol{x}_k - \boldsymbol{x}_{k-1}). \tag{19}$$

As a result, combining (16)–(19), we have

$$\begin{aligned}&\|\nabla f(\boldsymbol{x}_k) + r'(\boldsymbol{x}_k)\|_{\boldsymbol{\Lambda}^{-1}}^2 \\ &\le 2\sum_{j=1}^m \Big((\boldsymbol{x}_k - \boldsymbol{x}_{k-1})^T \hat{\boldsymbol{Q}}^j (\boldsymbol{x}_k - \boldsymbol{x}_{k-1}) + \|\boldsymbol{x}_k^j - \boldsymbol{x}_{k-1}^j\|_{\boldsymbol{\Lambda}_j}^2\Big) \\ &= 2(\boldsymbol{\Lambda}^{1/2}(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}))^T \boldsymbol{\Lambda}^{-1/2}\Big(\sum_{j=1}^m \hat{\boldsymbol{Q}}^j\Big)\boldsymbol{\Lambda}^{-1/2}(\boldsymbol{\Lambda}^{1/2}(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})) + 2\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|_{\boldsymbol{\Lambda}}^2 \\ &\le 2\Big\|\boldsymbol{\Lambda}^{-1/2}\Big(\sum_{j=1}^m \hat{\boldsymbol{Q}}^j\Big)\boldsymbol{\Lambda}^{-1/2}\Big\|\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|_{\boldsymbol{\Lambda}}^2 + 2\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|_{\boldsymbol{\Lambda}}^2 \\ &= 2\Big(\Big\|\boldsymbol{\Lambda}^{-1/2}\Big(\sum_{j=1}^m \hat{\boldsymbol{Q}}^j\Big)\boldsymbol{\Lambda}^{-1/2}\Big\| + 1\Big)\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|_{\boldsymbol{\Lambda}}^2. \end{aligned}$$

To complete the proof, it remains to note that, by definition, $\hat{L} = \Big\|\boldsymbol{\Lambda}^{-1/2}\Big(\sum_{j=1}^m \hat{\boldsymbol{Q}}^j\Big)\boldsymbol{\Lambda}^{-1/2}\Big\|$ and observe that $\operatorname{dist}^2(\partial F(\boldsymbol{x}_k), \boldsymbol{0}) \le \|\nabla f(\boldsymbol{x}_k) + r'(\boldsymbol{x}_k)\|_{\boldsymbol{\Lambda}^{-1}}$, for any $r'(\boldsymbol{x}_k) \in \partial r(\boldsymbol{x}_k)$. □

Lemma 3.2. *Under Assumptions 2.2 and 2.3, the iterates $\{\boldsymbol{x}_k\}$ generated by Algorithm 1 satisfy*

$$\sum_{i=1}^k \|\boldsymbol{x}_i - \boldsymbol{x}_{i-1}\|_{\boldsymbol{\Lambda}}^2 \le 2(F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)). \tag{11}$$

Proof. Let $r^{j,'}(\boldsymbol{x}_k^j) \in \partial r^j(\boldsymbol{x}_k^j)$ be such that $\boldsymbol{x}_k^j = \boldsymbol{x}_{k-1}^j - \boldsymbol{\Lambda}_j^{-1}(\nabla^j f(\boldsymbol{x}_{k-1}, j) + r^{j,'}(\boldsymbol{x}_k^j))$. By Assumption 2.2 and the definition of $\boldsymbol{x}_{k-1,j+1}$, we have

$$\begin{aligned}
F(\boldsymbol{x}_{k-1,j+1}) &= f(\boldsymbol{x}_{k-1,j+1}) + r(\boldsymbol{x}_{k-1,j+1}) \\
&\leq f(\boldsymbol{x}_{k-1,j}) + \langle \nabla^j f(\boldsymbol{x}_{k-1,j}), \boldsymbol{x}_k^j - \boldsymbol{x}_{k-1}^j \rangle + \frac{1}{2}\|\boldsymbol{x}_k^j - \boldsymbol{x}_{k-1}^j\|_{\boldsymbol{\Lambda}_j}^2 \\
&\quad + \sum_{i=1}^{j-1} r^i(\boldsymbol{x}_k^i) + r^j(\boldsymbol{x}_k^j) + \sum_{i=j+1}^{m} r^i(\boldsymbol{x}_{k-1}^i) \\
&\leq f(\boldsymbol{x}_{k-1,j}) + r(\boldsymbol{x}_{k-1,j}) + \langle \nabla^j f(\boldsymbol{x}_{k-1,j}) + r^{j,'}(\boldsymbol{x}_k^j), \boldsymbol{x}_k^j - \boldsymbol{x}_{k-1}^j \rangle + \frac{1}{2}\|\boldsymbol{x}_k^j - \boldsymbol{x}_{k-1}^j\|_{\boldsymbol{\Lambda}_j}^2 \\
&= F(\boldsymbol{x}_{k-1,j}) - \frac{1}{2}\|\boldsymbol{x}_k^j - \boldsymbol{x}_{k-1}^j\|_{\boldsymbol{\Lambda}_j}^2, \qquad (20)
\end{aligned}$$

where the second inequality is by the definition of a subgradient and last equality is by $\boldsymbol{x}_k^j = \boldsymbol{x}_{k-1}^j - \boldsymbol{\Lambda}_j^{-1}(\nabla^j f(\boldsymbol{x}_{k-1}, j) + r^{j,'}(\boldsymbol{x}_k^j))$. Applying (20) recursively over $j = 1, 2, \ldots, m$, we have

$$\begin{aligned}
F(\boldsymbol{x}_k) = F(\boldsymbol{x}_{k-1,m+1}) &\leq F(\boldsymbol{x}_{k-1,1}) - \sum_{j=1}^{m} \frac{1}{2}\|\boldsymbol{x}_k^j - \boldsymbol{x}_{k-1}^j\|_{\boldsymbol{\Lambda}_j}^2 \\
&= F(\boldsymbol{x}_{k-1}) - \frac{1}{2}\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|_{\boldsymbol{\Lambda}}^2. \qquad (21)
\end{aligned}$$

where the last equality is by $\boldsymbol{x}_{k-1,1} = \boldsymbol{x}_{k-1}$ and the definition of $\boldsymbol{\Lambda}$.

Telescoping (21) from $i = 1$ to k, we have

$$F(\boldsymbol{x}_k) \leq F(\boldsymbol{x}_0) - \sum_{i=1}^{k} \frac{1}{2}\|\boldsymbol{x}_i - \boldsymbol{x}_{i-1}\|_{\boldsymbol{\Lambda}}^2. \qquad (22)$$

It remains to use that $F(\boldsymbol{x}_k) \geq F(\boldsymbol{x}^*)$, by the definition of $\boldsymbol{x}^*$. □

Corollary 3.4. *Suppose that the conditions of Theorem 3.3 hold and that F further satisfies Assumption 2.4. Then we have after K iterations of Algorithm 1 that*

$$F(\boldsymbol{x}_K) - F(\boldsymbol{x}^*) \leq \Big(\frac{2(\hat{L}+1)}{2(\hat{L}+1)+\mu}\Big)^K (F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)).$$

Proof. Combining Lemma 3.1 and Inequality (21), we have

$$\begin{aligned}
\operatorname{dist}^2(\partial F(\boldsymbol{x}_k), \mathbf{0}) &\leq 2(\hat{L}+1)\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|_{\boldsymbol{\Lambda}}^2 \\
&\leq 4(\hat{L}+1)(F(\boldsymbol{x}_{k-1}) - F(\boldsymbol{x}_k)).
\end{aligned}$$

Using Assumption 2.4 and the last inequality, we have

$$2\mu(F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*)) \leq 4(\hat{L}+1)(F(\boldsymbol{x}_{k-1}) - F(\boldsymbol{x}_k)),$$

which leads to

$$F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*) \leq \frac{2(\hat{L}+1)}{2(\hat{L}+1)+\mu}(F(\boldsymbol{x}_{k-1}) - F(\boldsymbol{x}^*)).$$

Applying the last inequality recursively from K down to 1, we obtain

$$F(\boldsymbol{x}_K) - F(\boldsymbol{x}^*) \leq \Big(\frac{2(\hat{L}+1)}{2(\hat{L}+1)+\mu}\Big)^K (F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)),$$

which completes the proof. □

C. Omitted Proofs from Section 4

To prove Theorem 4.4, we first prove the following auxiliary lemma.

Lemma 4.1. *Suppose Assumptions 2.5–2.7 hold, then the variance $\mathbb{E}[u_k]$ of the gradient estimators $\{\boldsymbol{g}_{k-1}^j\}_{j=1}^m$ at iteration k of Algorithm 2 is bounded by:*

$$\mathbb{E}[u_k] \leq \frac{2p(n-b)\sigma^2}{b(n-1)} + 2\Big(\frac{p(n-b)}{b(n-1)} + \frac{1-p}{b'}\Big)\tilde{L}\mathbb{E}[v_k] + (1-p)\mathbb{E}[u_{k-1}] + \frac{2(1-p)\hat{L}}{b'}\mathbb{E}[v_{k-1}].$$

Proof. Let $\mathcal{F}_{k,j-1}$ denote the natural filtration, containing all algorithm randomness up to and including outer iteration k and inner iteration $j-1$. By the definition of $\boldsymbol{g}_{k-1}^j$, we have

$$\begin{aligned}
&\mathbb{E}[\|\boldsymbol{g}_{k-1}^j - \nabla^j f(\boldsymbol{x}_{k-1,j})\|^2_{\boldsymbol{\Lambda}_j^{-1}}|\mathcal{F}_{k,j-1}] \\
&= p\mathbb{E}\Big[\big\|\frac{1}{b}\sum_{i=1}^{b}\nabla^j f_i(\boldsymbol{x}_{k-1,j}) - \nabla^j f(\boldsymbol{x}_{k-1,j})\big\|^2_{\boldsymbol{\Lambda}_j^{-1}}|\mathcal{F}_{k,j-1}\Big] \\
&\quad + (1-p)\mathbb{E}\Big[\big\|\boldsymbol{g}_{k-2}^j + \frac{1}{b'}\sum_{i=1}^{b'}(\nabla^j f_i(\boldsymbol{x}_{k-1,j}) - \nabla^j f_i(\boldsymbol{x}_{k-2,j})) - \nabla^j f(\boldsymbol{x}_{k-1,j})\big\|^2_{\boldsymbol{\Lambda}_j^{-1}}|\mathcal{F}_{k,j-1}\Big] \\
&= p\mathbb{E}\Big[\big\|\frac{1}{b}\sum_{i=1}^{b}\nabla^j f_i(\boldsymbol{x}_{k-1,j}) - \nabla^j f(\boldsymbol{x}_{k-1,j})\big\|^2_{\boldsymbol{\Lambda}_j^{-1}}|\mathcal{F}_{k,j-1}\Big] + (1-p)\|\boldsymbol{g}_{k-2}^j - \nabla^j f(\boldsymbol{x}_{k-2,j})\|^2_{\boldsymbol{\Lambda}_j^{-1}} \\
&\quad + (1-p)\mathbb{E}\Big[\big\|\frac{1}{b'}\sum_{i=1}^{b'}(\nabla^j f_i(\boldsymbol{x}_{k-1,j}) - \nabla^j f_i(\boldsymbol{x}_{k-2,j})) - \nabla^j f(\boldsymbol{x}_{k-1,j}) + \nabla^j f(\boldsymbol{x}_{k-2,j})\big\|^2_{\boldsymbol{\Lambda}_j^{-1}}|\mathcal{F}_{k,j-1}\Big],
\end{aligned}$$

where the last equality uses that $\boldsymbol{g}_{k-2}^j - \nabla^j f(\boldsymbol{x}_{k-2,j})$ is measurable w.r.t. $\mathcal{F}_{k,j-1}$ and $\mathbb{E}[\frac{1}{b'}\sum_{i=1}^{b'}(\nabla^j f_i(\boldsymbol{x}_{k-1,j}) - \nabla^j f_i(\boldsymbol{x}_{k-2,j})) - \nabla^j f(\boldsymbol{x}_{k-1,j}) + \nabla^j f(\boldsymbol{x}_{k-2,j})|\mathcal{F}_{k,j-1}] = 0$. Using that for any random variable X, $\mathbb{E}[(X - \mathbb{E}[X])^2] \leq \mathbb{E}[X^2]$ and proceeding as in Lemma 2.8 with respect to $\|\cdot\|_{\boldsymbol{\Lambda}_j^{-1}}$, we further have

$$\begin{aligned}
&\mathbb{E}[\|\boldsymbol{g}_{k-1}^j - \nabla^j f(\boldsymbol{x}_{k-1,j})\|^2_{\boldsymbol{\Lambda}_j^{-1}}|\mathcal{F}_{k,j-1}] \\
&\leq p\mathbb{E}\Big[\big\|\frac{1}{b}\sum_{i=1}^{b}\nabla^j f_i(\boldsymbol{x}_{k-1,j}) - \nabla^j f(\boldsymbol{x}_{k-1,j})\big\|^2_{\boldsymbol{\Lambda}_j^{-1}}|\mathcal{F}_{k,j-1}\Big] + (1-p)\|\boldsymbol{g}_{k-2}^j - \nabla^j f(\boldsymbol{x}_{k-2,j})\|^2_{\boldsymbol{\Lambda}_j^{-1}} \\
&\quad + (1-p)\mathbb{E}\Big[\big\|\frac{1}{b'}\sum_{i=1}^{b'}(\nabla^j f_i(\boldsymbol{x}_{k-1,j}) - \nabla^j f_i(\boldsymbol{x}_{k-2,j}))\|^2_{\boldsymbol{\Lambda}_j^{-1}}|\mathcal{F}_{k,j-1}\Big] \\
&\leq \frac{p(n-b)}{b(n-1)}\mathbb{E}_i\big[\|\nabla^j f_i(\boldsymbol{x}_{k-1,j}) - \nabla^j f(\boldsymbol{x}_{k-1,j})\big\|^2_{\boldsymbol{\Lambda}_j^{-1}}\big] + (1-p)\big\|\boldsymbol{g}_{k-2}^j - \nabla^j f(\boldsymbol{x}_{k-2,j})\|^2_{\boldsymbol{\Lambda}_j^{-1}} \\
&\quad + \frac{1-p}{b'}\mathbb{E}_i[\|\nabla^j f_i(\boldsymbol{x}_{k-1,j}) - \nabla^j f_i(\boldsymbol{x}_{k-2,j})\|^2_{\boldsymbol{\Lambda}_j^{-1}}].
\end{aligned} \tag{23}$$

We now proceed to simplify Eq. (23), by simplifying the first and the last term appearing in it. For the last term, we have, using our smoothness assumption,

$$\begin{aligned}
&\mathbb{E}_i\Big[\big\|\nabla^j f_i(\boldsymbol{x}_{k-1,j}) - \nabla^j f_i(\boldsymbol{x}_{k-2,j})\big\|^2_{\boldsymbol{\Lambda}_j^{-1}}\Big] \\
&\leq 2\mathbb{E}_i\Big[\big\|\nabla^j f_i(\boldsymbol{x}_{k-1,j}) - \nabla^j f_i(\boldsymbol{x}_{k-1})\big\|^2_{\boldsymbol{\Lambda}_j^{-1}}\Big] + 2\mathbb{E}_i\Big[\big\|\nabla^j f_i(\boldsymbol{x}_{k-1}) - \nabla^j f_i(\boldsymbol{x}_{k-2,j})\big\|^2_{\boldsymbol{\Lambda}_j^{-1}}\Big] \\
&\leq 2(\boldsymbol{x}_{k-1,j} - \boldsymbol{x}_{k-1})^T\boldsymbol{Q}^j(\boldsymbol{x}_{k-1,j} - \boldsymbol{x}_{k-1}) + 2(\boldsymbol{x}_{k-1} - \boldsymbol{x}_{k-2,j})^T\boldsymbol{Q}^j(\boldsymbol{x}_{k-1} - \boldsymbol{x}_{k-2,j}) \\
&= 2(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})^T\widetilde{\boldsymbol{Q}}^j(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}) + 2(\boldsymbol{x}_{k-1} - \boldsymbol{x}_{k-2})^T\widehat{\boldsymbol{Q}}^j(\boldsymbol{x}_{k-1} - \boldsymbol{x}_{k-2}),
\end{aligned} \tag{24}$$

where the first inequality comes from adding and subtracting $\nabla^j f_i(\boldsymbol{x}_{k-1})$ and using Young's inequality, the second inequality is by Assumption 2.7, and the last equality is by the definitions of $\boldsymbol{x}_{k-1,j}$ and matrices $\widehat{\boldsymbol{Q}}^j$ and $\widetilde{\boldsymbol{Q}}^j$. Summing Eq. (24) from $j=1$ to m, it follows that

$$\begin{aligned}
&\sum_{j=1}^m \mathbb{E}_i\Big[\big\|\nabla^j f_i(\boldsymbol{x}_{k-1,j}) - \nabla^j f_i(\boldsymbol{x}_{k-2,j})\big\|^2_{\boldsymbol{\Lambda}_j^{-1}}\Big] \\
&\le 2(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})^T\Big(\sum_{j=1}^m \widetilde{\boldsymbol{Q}}^j\Big)(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}) + 2(\boldsymbol{x}_{k-1} - \boldsymbol{x}_{k-2})^T\Big(\sum_{j=1}^m \widehat{\boldsymbol{Q}}^j\Big)(\boldsymbol{x}_{k-1} - \boldsymbol{x}_{k-2}) \\
&\le 2\Big\|\boldsymbol{\Lambda}^{-1/2}\Big(\sum_{j=1}^m \widetilde{\boldsymbol{Q}}^j\Big)\boldsymbol{\Lambda}^{-1/2}\Big\|\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|^2_{\boldsymbol{\Lambda}} + 2\Big\|\boldsymbol{\Lambda}^{-1/2}\Big(\sum_{j=1}^m \widehat{\boldsymbol{Q}}^j\Big)\boldsymbol{\Lambda}^{-1/2}\Big\|\|\boldsymbol{x}_{k-1} - \boldsymbol{x}_{k-2}\|^2_{\boldsymbol{\Lambda}} \\
&= 2\tilde{L}\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|^2_{\boldsymbol{\Lambda}} + 2\hat{L}\|\boldsymbol{x}_{k-1} - \boldsymbol{x}_{k-2}\|^2_{\boldsymbol{\Lambda}}.
\end{aligned}$$

Taking expectation with all the randomness in the algorithm on both sides and applying the tower property of expectation, we have

$$\mathbb{E}\Big[\sum_{j=1}^m \big\|\nabla^j f_i(\boldsymbol{x}_{k-1,j}) - \nabla^j f_i(\boldsymbol{x}_{k-2,j})\big\|^2_{\boldsymbol{\Lambda}_j^{-1}}\Big] \le 2\tilde{L}\mathbb{E}\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|^2_{\boldsymbol{\Lambda}} + 2\hat{L}\mathbb{E}\|\boldsymbol{x}_{k-1} - \boldsymbol{x}_{k-2}\|^2_{\boldsymbol{\Lambda}}. \tag{25}$$

On the other hand, to simplify the first term, we add and subtract $\nabla^j f_i(\boldsymbol{x}_{k-1}) + \nabla^j f(\boldsymbol{x}_{k-1})$ and apply Young's inequality to get

$$\begin{aligned}
&\mathbb{E}_i\Big[\big\|\nabla^j f_i(\boldsymbol{x}_{k-1,j}) - \nabla^j f(\boldsymbol{x}_{k-1,j})\big\|^2_{\boldsymbol{\Lambda}_j^{-1}}\Big] \\
&= \mathbb{E}_i\Big[\|\nabla^j f_i(\boldsymbol{x}_{k-1,j}) - \nabla^j f_i(\boldsymbol{x}_{k-1}) + \nabla^j f(\boldsymbol{x}_{k-1}) - \nabla^j f(\boldsymbol{x}_{k-1,j}) + \nabla^j f_i(\boldsymbol{x}_{k-1}) - \nabla^j f(\boldsymbol{x}_{k-1})\|^2_{\boldsymbol{\Lambda}_j^{-1}}\Big] \\
&\le 2\mathbb{E}_i\Big[\|\nabla^j f_i(\boldsymbol{x}_{k-1,j}) - \nabla^j f_i(\boldsymbol{x}_{k-1}) - \big(\nabla^j f(\boldsymbol{x}_{k-1,j}) - \nabla^j f(\boldsymbol{x}_{k-1})\big)\|^2_{\boldsymbol{\Lambda}_j^{-1}} + \|\nabla^j f_i(\boldsymbol{x}_{k-1}) - \nabla^j f(\boldsymbol{x}_{k-1})\|^2_{\boldsymbol{\Lambda}_j^{-1}}\Big].
\end{aligned}$$

Similarly as before, using that the variance of any random variable is bounded by its second moment, and summing from $j=1$ to m, we further have

$$\begin{aligned}
&\sum_{j=1}^m \mathbb{E}_i\Big[\big\|\nabla^j f_i(\boldsymbol{x}_{k-1,j}) - \nabla^j f(\boldsymbol{x}_{k-1,j})\big\|^2_{\boldsymbol{\Lambda}_j^{-1}}\Big] \\
&\le 2\sum_{j=1}^m \mathbb{E}_i\Big[\|\nabla^j f_i(\boldsymbol{x}_{k-1,j}) - \nabla^j f_i(\boldsymbol{x}_{k-1})\|^2_{\boldsymbol{\Lambda}_j^{-1}} + \|\nabla^j f_i(\boldsymbol{x}_{k-1}) - \nabla^j f(\boldsymbol{x}_{k-1})\|^2_{\boldsymbol{\Lambda}_j^{-1}}\Big] \\
&= 2\sum_{j=1}^m \mathbb{E}_i\Big[\|\nabla^j f_i(\boldsymbol{x}_{k-1,j}) - \nabla^j f_i(\boldsymbol{x}_{k-1})\|^2_{\boldsymbol{\Lambda}_j^{-1}}\Big] + 2\mathbb{E}_i\Big[\|\nabla f_i(\boldsymbol{x}_{k-1}) - \nabla f(\boldsymbol{x}_{k-1})\|^2_{\boldsymbol{\Lambda}^{-1}}\Big] \\
&\le 2\tilde{L}\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|^2_{\boldsymbol{\Lambda}} + 2\sigma^2,
\end{aligned}$$

where in the last inequality we used the arguments as in deriving (24) and (25), and Assumption 2.5. Further, taking expectation with all the randomness in the algorithm on both sides and using the tower property of expectation, we have

$$\mathbb{E}\Big[\sum_{j=1}^m \big\|\nabla^j f_i(\boldsymbol{x}_{k-1,j}) - \nabla^j f(\boldsymbol{x}_{k-1,j})\big\|^2_{\boldsymbol{\Lambda}_j^{-1}}\Big] \le 2\tilde{L}\mathbb{E}\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|^2_{\boldsymbol{\Lambda}} + 2\sigma^2. \tag{26}$$

Summing Eq. (23) from $j=1$ to m, taking the expectation w.r.t. all the randomness in the algorithm on both sides, and

applying linearity and the tower property of expectation $\mathbb{E}[\mathbb{E}[\cdot|\mathcal{F}_{k,j-1}]] = \mathbb{E}[\cdot]$ for each $j \in [m]$, we have

$$\begin{aligned}\mathbb{E}\Big[\sum_{j=1}^{m}\|\boldsymbol{g}_{k-1}^{j}-\nabla^{j}f(\boldsymbol{x}_{k-1,j})\|_{\boldsymbol{\Lambda}_j^{-1}}^{2}\Big] \leq \frac{p(n-b)}{b(n-1)}\mathbb{E}\Big[\sum_{j=1}^{m}\|\nabla^{j}f_i(\boldsymbol{x}_{k-1,j})-\nabla^{j}f(\boldsymbol{x}_{k-1,j})\Big\|_{\boldsymbol{\Lambda}_j^{-1}}^{2}\Big] \\ +(1-p)\mathbb{E}\Big[\sum_{j=1}^{m}\|\boldsymbol{g}_{k-2}^{j}-\nabla^{j}f(\boldsymbol{x}_{k-2,j})\|_{\boldsymbol{\Lambda}_j^{-1}}^{2}\Big] \\ +\frac{1-p}{b'}\mathbb{E}\Big[\sum_{j=1}^{m}\|\nabla^{j}f_i(\boldsymbol{x}_{k-1,j})-\nabla^{j}f_i(\boldsymbol{x}_{k-2,j})\|_{\boldsymbol{\Lambda}_j^{-1}}^{2}\Big]\end{aligned} \tag{27}$$

To complete the proof, it remains to plug Inequalities (25)–(26) into Inequality (27) and do some rearrangements. □

Lemma 4.2. *Let Assumption 2.6 hold and let $r^{j,'}(\boldsymbol{x}_k^j) \in \partial r^j(\boldsymbol{x}_k^j)$ be such that $\boldsymbol{x}_k^j = \boldsymbol{x}_{k-1}^j - \eta\boldsymbol{\Lambda}_j^{-1}(\boldsymbol{g}_{k-1}^j + r^{j,'}(\boldsymbol{x}_k^j))$. Then the iterates of Algorithm 2 satisfy*

$$\begin{aligned}F(\boldsymbol{x}_k) \leq F(\boldsymbol{x}_{k-1}) - \frac{1-\eta}{2\eta}v_k + \frac{\eta}{2}u_k \\ -\frac{\eta}{2}\sum_{j=1}^{m}\|\nabla^{j}f(\boldsymbol{x}_{k-1,j})+r^{j,'}(\boldsymbol{x}_k^j)\|_{\boldsymbol{\Lambda}_j^{-1}}^{2}.\end{aligned} \tag{13}$$

Proof. By Assumption 2.2 and the definitions of $\boldsymbol{x}_{k-1,j+1}$ and $\boldsymbol{x}_{k-1,j}$, we have

$$\begin{aligned}&F(\boldsymbol{x}_{k-1,j+1}) = f(\boldsymbol{x}_{k-1,j+1}) + r(\boldsymbol{x}_{k-1,j+1}) \\ &\leq f(\boldsymbol{x}_{k-1,j}) + \langle\nabla^{j}f(\boldsymbol{x}_{k-1,j}), \boldsymbol{x}_k^j - \boldsymbol{x}_{k-1}^j\rangle + \frac{1}{2}\|\boldsymbol{x}_k^j - \boldsymbol{x}_{k-1}^j\|_{\boldsymbol{\Lambda}_j}^2 + \sum_{i=1}^{j-1}r^i(\boldsymbol{x}_k^i) + r^j(\boldsymbol{x}_k^j) + \sum_{i=j+1}^{m}r^i(\boldsymbol{x}_{k-1}^i) \\ &\leq f(\boldsymbol{x}_{k-1,j}) + r(\boldsymbol{x}_{k-1,j}) + \langle\nabla^{j}f(\boldsymbol{x}_{k-1,j}) + r^{j,'}(\boldsymbol{x}_k^j), \boldsymbol{x}_k^j - \boldsymbol{x}_{k-1}^j\rangle + \frac{1}{2}\|\boldsymbol{x}_k^j - \boldsymbol{x}_{k-1}^j\|_{\boldsymbol{\Lambda}_j}^2 \\ &= F(\boldsymbol{x}_{k-1,j}) + \langle\boldsymbol{g}_{k-1}^j + r^{j,'}(\boldsymbol{x}_k^j), \boldsymbol{x}_k^j - \boldsymbol{x}_{k-1}^j\rangle + \frac{1}{2}\|\boldsymbol{x}_k^j - \boldsymbol{x}_{k-1}^j\|_{\boldsymbol{\Lambda}_j}^2 + \langle\nabla^{j}f(\boldsymbol{x}_{k-1,j}) - \boldsymbol{g}_{k-1}^j, \boldsymbol{x}_k^j - \boldsymbol{x}_{k-1}^j\rangle.\end{aligned}$$

Since, by assumption, $r^{j,'}(\boldsymbol{x}_k^j) \in \partial r^j(\boldsymbol{x}_k^j)$ is such that $\boldsymbol{x}_k^j = \boldsymbol{x}_{k-1}^j - \eta\boldsymbol{\Lambda}_j^{-1}(\boldsymbol{g}_{k-1}^j + r^{j,'}(\boldsymbol{x}_k^j))$, we further have

$$F(\boldsymbol{x}_{k-1,j+1}) \leq F(\boldsymbol{x}_{k-1,j}) - \big(\frac{1}{\eta}-\frac{1}{2}\big)\|\boldsymbol{x}_k^j - \boldsymbol{x}_{k-1}^j\|_{\boldsymbol{\Lambda}_j}^2 + \Big\langle\nabla^{j}f(\boldsymbol{x}_{k-1,j}) - \boldsymbol{g}_{k-1}^j, -\eta\boldsymbol{\Lambda}_j^{-1}(\boldsymbol{g}_{k-1}^j + r^{j,'}(\boldsymbol{x}_k^j))\Big\rangle. \tag{28}$$

To simplify (28), we now further simplify the last term appearing in it, as follows:

$$\begin{aligned}&\Big\langle\nabla^{j}f(\boldsymbol{x}_{k-1,j}) - \boldsymbol{g}_{k-1}^j, -\eta\boldsymbol{\Lambda}_j^{-1}(\boldsymbol{g}_{k-1}^j + r^{j,'}(\boldsymbol{x}_k^j))\Big\rangle \\ &= \eta\|\nabla^{j}f(\boldsymbol{x}_{k-1,j}) - \boldsymbol{g}_{k-1}^j\|_{\boldsymbol{\Lambda}_j^{-1}}^2 - \eta\langle\nabla^{j}f(\boldsymbol{x}_{k-1,j}) - \boldsymbol{g}_{k-1}^j, \boldsymbol{\Lambda}_j^{-1}\big(\nabla^{j}f(\boldsymbol{x}_{k-1,j}) + r^{j,'}(\boldsymbol{x}_k^j)\big)\rangle \\ &= \eta\|\nabla^{j}f(\boldsymbol{x}_{k-1,j}) - \boldsymbol{g}_{k-1}^j\|_{\boldsymbol{\Lambda}_j^{-1}}^2 \\ &\quad - \frac{\eta}{2}\Big(\|\nabla^{j}f(\boldsymbol{x}_{k-1,j}) - \boldsymbol{g}_{k-1}^j\|_{\boldsymbol{\Lambda}_j^{-1}}^2 + \|\nabla^{j}f(\boldsymbol{x}_{k-1,j}) + r^{j,'}(\boldsymbol{x}_k^j)\|_{\boldsymbol{\Lambda}_j^{-1}}^2 - \|\boldsymbol{g}_{k-1}^j + r^{j,'}(\boldsymbol{x}_k^j)\|_{\boldsymbol{\Lambda}_j^{-1}}^2\Big) \\ &= \frac{\eta}{2}\|\nabla^{j}f(\boldsymbol{x}_{k-1,j}) - \boldsymbol{g}_{k-1}^j\|_{\boldsymbol{\Lambda}_j^{-1}}^2 - \frac{\eta}{2}\|\nabla^{j}f(\boldsymbol{x}_{k-1,j}) + r^{j,'}(\boldsymbol{x}_k^j)\|_{\boldsymbol{\Lambda}_j^{-1}}^2 + \frac{\eta}{2}\|\boldsymbol{g}_{k-1}^j + r^{j,'}(\boldsymbol{x}_k^j)\|_{\boldsymbol{\Lambda}_j^{-1}}^2 \\ &= \frac{\eta}{2}\|\nabla^{j}f(\boldsymbol{x}_{k-1,j}) - \boldsymbol{g}_{k-1}^j\|_{\boldsymbol{\Lambda}_j^{-1}}^2 - \frac{\eta}{2}\|\nabla^{j}f(\boldsymbol{x}_{k-1,j}) + r^{j,'}(\boldsymbol{x}_k^j)\|_{\boldsymbol{\Lambda}_j^{-1}}^2 + \frac{1}{2\eta}\|\boldsymbol{x}_k^j - \boldsymbol{x}_{k-1}^j\|_{\boldsymbol{\Lambda}_j}^2.\end{aligned} \tag{29}$$

Thus, combining (28) and (29), we have

$$\begin{aligned}F(\boldsymbol{x}_{k-1,j+1}) \leq F(\boldsymbol{x}_{k-1,j}) - \frac{1}{2}\big(\frac{1}{\eta}-1\big)\|\boldsymbol{x}_k^j - \boldsymbol{x}_{k-1}^j\|_{\boldsymbol{\Lambda}_j}^2 + \frac{\eta}{2}\|\nabla^{j}f(\boldsymbol{x}_{k-1,j}) - \boldsymbol{g}_{k-1}^j\|_{\boldsymbol{\Lambda}_j^{-1}}^2 \\ -\frac{\eta}{2}\|\nabla^{j}f(\boldsymbol{x}_{k-1,j}) + r^{j,'}(\boldsymbol{x}_k^j)\|_{\boldsymbol{\Lambda}_j^{-1}}^2.\end{aligned} \tag{30}$$

Summing (30) from $j = 1$ to m,

$$F(\boldsymbol{x}_{k-1,m+1}) \leq F(\boldsymbol{x}_{k-1,1}) - \frac{1}{2}\big(\frac{1}{\eta} - 1\big)\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|_{\boldsymbol{\Lambda}}^2 + \frac{\eta}{2}\sum_{j=1}^{m}\|\nabla^j f(\boldsymbol{x}_{k-1,j}) - \boldsymbol{g}_{k-1}^j\|_{\boldsymbol{\Lambda}_j^{-1}}^2 - \frac{\eta}{2}\sum_{j=1}^{m}\|\nabla^j f(\boldsymbol{x}_{k-1,j}) + r^{j,'}(\boldsymbol{x}_k^j)\|_{\boldsymbol{\Lambda}_j^{-1}}^2. \tag{31}$$

To complete the proof, it remains to observe that $\boldsymbol{x}_k = \boldsymbol{x}_{k-1,m+1}$ and $\boldsymbol{x}_{k-1} = \boldsymbol{x}_{k-1,1}$. □

Lemma 4.3. *Let Assumption 2.7 hold and let $r^{j,'}(\boldsymbol{x}_k^j) \in \partial r^j(\boldsymbol{x}_k^j)$ be such that $\boldsymbol{x}_k^j = \boldsymbol{x}_{k-1}^j - \eta\boldsymbol{\Lambda}_j^{-1}(\boldsymbol{g}_{k-1}^j + r^{j,'}(\boldsymbol{x}_k^j))$. Then for Algorithm 2 we have*

$$s_k \leq 2\hat{L}v_k + 2\sum_{j=1}^{m}\|\nabla^j f(\boldsymbol{x}_{k-1,j}) + r^{j,'}(\boldsymbol{x}_k)\|_{\boldsymbol{\Lambda}_j^{-1}}^2. \tag{14}$$

Proof. Observe first that

$$\begin{aligned} s_k &\leq \|\nabla f(\boldsymbol{x}_k) + r'(\boldsymbol{x}_k)\|_{\boldsymbol{\Lambda}^{-1}}^2 \\ &= \sum_{j=1}^{m}\|\nabla^j f(\boldsymbol{x}_k) + r^{j,'}(\boldsymbol{x}_k)\|_{\boldsymbol{\Lambda}_j^{-1}}^2 \\ &\leq \sum_{j=1}^{m} 2\big(\|\nabla^j f(\boldsymbol{x}_k) - \nabla^j f(\boldsymbol{x}_{k-1,j})\|_{\boldsymbol{\Lambda}_j^{-1}}^2 + \|\nabla^j f(\boldsymbol{x}_{k-1,j}) + r^{j,'}(\boldsymbol{x}_k)\|_{\boldsymbol{\Lambda}_j^{-1}}^2\big), \end{aligned} \tag{32}$$

where we have used Young's inequality. On the other hand, by Assumption 2.7 and the definitions of $\boldsymbol{x}_{k-1,j}$ and $\widehat{\boldsymbol{Q}}^j$, we have

$$\begin{aligned} \sum_{j=1}^{m}\|\nabla^j f(\boldsymbol{x}_k) - \nabla^j f(\boldsymbol{x}_{k-1,j})\|_{\boldsymbol{\Lambda}_j^{-1}}^2 &\leq \sum_{j=1}^{m}(\boldsymbol{x}_k - \boldsymbol{x}_{k-1,j})^T\boldsymbol{Q}^j(\boldsymbol{x}_k - \boldsymbol{x}_{k-1,j}) \\ &= \sum_{j=1}^{m}(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})^T\widehat{\boldsymbol{Q}}^j(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}) \\ &\leq \Big\|\boldsymbol{\Lambda}^{-1/2}\sum_{j=1}^{m}\hat{\boldsymbol{Q}}^j\boldsymbol{\Lambda}^{-1/2}\Big\|\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|_{\boldsymbol{\Lambda}}^2 \\ &= \hat{L}\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|_{\boldsymbol{\Lambda}}^2. \end{aligned} \tag{33}$$

Combining (32) and (33) completes the proof. □

Theorem 4.4. *Suppose that Assumptions 2.2–2.3 and 2.5–2.7 hold. Let $\boldsymbol{x}^*$ be a global minimizer of (1) and $\{\boldsymbol{x}_k\}$ be the iterates generated by Algorithm 2. Then, we have*

$$\begin{aligned} &\mathbb{E}\big[\mathrm{dist}^2(\partial F(\hat{\boldsymbol{x}}_K), \boldsymbol{0})\big] \\ &\leq \frac{4\Delta_0}{\eta K} + \frac{2(1-p)(n-b)\sigma^2}{pb(n-1)K} + \frac{4(n-b)\sigma^2}{b(n-1)}, \end{aligned} \tag{15}$$

where $\Delta_0 = F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)$, and $0 < \eta \leq \frac{-1+\sqrt{1+4c_0}}{2c_0}$ with $c_0 = \frac{2(1-p)\hat{L}}{pb'} + \hat{L} + 2\big(\frac{p(n-b)}{b(n-1)} + \frac{1-p}{b'}\big)\frac{\tilde{L}}{p}$.

Proof. Combining Lemmas 4.2 and 4.3, we have

$$F(\boldsymbol{x}_k) \leq F(\boldsymbol{x}_{k-1}) - \frac{1}{2}\big(\frac{1}{\eta} - 1 - \hat{L}\eta\big)v_k + \frac{\eta}{2}u_k - \frac{\eta}{4}s_k. \tag{34}$$

For notational convenience, denote $r_k = F(\boldsymbol{x}_k)$; then Eq. (34) is equivalent to

$$r_k \leq r_{k-1} - \frac{1}{2}\big(\frac{1}{\eta} - 1 - \hat{L}\eta\big)v_k + \frac{\eta}{2}u_k - \frac{\eta}{4}s_k. \tag{35}$$

Taking expectation on both sides of Eq. (35) with respect to all randomness of the algorithm, and adding $\frac{\eta}{2p} \times$ (12) to (35), we have

$$\begin{aligned}\mathbb{E}[r_k] + \frac{(1-p)\eta}{2p}\mathbb{E}[u_k] \leq \mathbb{E}[r_{k-1}] + \frac{(1-p)\eta}{2p}\mathbb{E}[u_{k-1}] + \frac{(n-b)\sigma^2\eta}{b(n-1)} - \frac{\eta}{4}\mathbb{E}[s_k] \\ - \frac{1}{2}\bigg(\frac{1}{\eta} - 1 - \hat{L}\eta - 4\Big(\frac{p(n-b)}{b(n-1)} + \frac{1-p}{b'}\Big)\frac{\tilde{L}\eta}{2p}\bigg)\mathbb{E}[v_k] + \frac{(1-p)\hat{L}\eta}{pb'}\mathbb{E}[v_{k-1}].\end{aligned}$$

Observe that in the last inequality, the terms corresponding to $\mathbb{E}[r_k]$ and $\mathbb{E}[u_k]$ telescope. To further simplify this inequality, we now make a choice of η that ensures that the terms that correspond to $\mathbb{E}[v_k]$ telescope as well. In particular, for the terms corresponding to $\mathbb{E}[v_k]$ and $\mathbb{E}[v_{k-1}]$ to be telescoping, we need

$$\frac{1}{\eta} - 1 - \hat{L}\eta - 4\Big(\frac{p(n-b)}{b(n-1)} + \frac{1-p}{b'}\Big)\frac{\tilde{L}\eta}{2p} \geq \frac{2(1-p)\hat{L}\eta}{pb'}. \tag{36}$$

To simplify (36), denote by $c_0 = \frac{2(1-p)\hat{L}}{pb'} + \hat{L} + 2\Big(\frac{p(n-b)}{b(n-1)} + \frac{1-p}{b'}\Big)\frac{\tilde{L}}{p}$ the coefficient multiplying η. Then, solving (36), which is a quadratic inequality in η, we get that it suffices to have

$$0 < \eta \leq \frac{-1 + \sqrt{1 + 4c_0}}{2c_0},$$

as required by the theorem assumptions. Thus, we obtain

$$\begin{aligned}&\frac{\eta}{4}\mathbb{E}[s_k] + \mathbb{E}[r_k] + \frac{(1-p)\eta}{2p}\mathbb{E}[u_k] + \frac{(1-p)\hat{L}\eta}{pb'}\mathbb{E}[v_k] \\ &\leq \mathbb{E}[r_{k-1}] + \frac{(1-p)\eta}{2p}\mathbb{E}[u_{k-1}] + \frac{(1-p)\hat{L}\eta}{pb'}\mathbb{E}[v_{k-1}] + \frac{(n-b)\sigma^2\eta}{b(n-1)}.\end{aligned} \tag{37}$$

Telescoping Eq. (37) from 1 to k, we have

$$\begin{aligned}&\sum_{i=1}^{k}\frac{\eta}{4}\mathbb{E}[s_i] + \mathbb{E}[r_k] + \frac{(1-p)\eta}{2p}\mathbb{E}[u_k] + \frac{(1-p)\hat{L}\eta}{pb'}\mathbb{E}[v_k] \\ &\leq \mathbb{E}[r_0] + \frac{(1-p)\eta}{2p}\mathbb{E}[u_0] + \frac{(1-p)\hat{L}\eta}{pb'}\mathbb{E}[v_0] + \frac{(n-b)\sigma^2\eta k}{b(n-1)}.\end{aligned} \tag{38}$$

With the fact that $r_k = F(\boldsymbol{x}_k) \geq F(\boldsymbol{x}^*), u_k \geq 0, v_k \geq 0$, $\mathbb{E}[u_0] = \mathbb{E}\Big[\sum_{j=1}^{m}\|\nabla^j f(\boldsymbol{x}_{-1,j}) - \boldsymbol{g}_{-1}^j\|^2_{\boldsymbol{\Lambda}_j^{-1}}\Big] \leq \frac{(n-b)\sigma^2}{b(n-1)}$ (by Lemma 2.8 adapted to $\boldsymbol{\Lambda}_j$ norms), $v_0 = \|\boldsymbol{x}_0 - \boldsymbol{x}_{-1}\|^2_{\boldsymbol{\Lambda}} = 0$ and the definition of s_k, we have

$$\sum_{i=1}^{k}\frac{\eta}{4}\mathbb{E}\Big[\mathrm{dist}^2(\partial F(\boldsymbol{x}_i), \mathbf{0})\Big] \leq F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*) + \frac{(1-p)(n-b)\eta\sigma^2}{2pb(n-1)} + \frac{(n-b)\sigma^2\eta k}{b(n-1)}. \tag{39}$$

Further, taking $\hat{\boldsymbol{x}}_K$ to be uniformly at random chosen from $\{\boldsymbol{x}_k\}_{k\in[K]}$, we have

$$\mathbb{E}_K\Big[\mathrm{dist}^2(\partial F(\hat{\boldsymbol{x}}_K), \mathbf{0})\Big] = \frac{1}{K}\sum_{i=1}^{K}\mathrm{dist}^2(\partial F(\boldsymbol{x}_i), \mathbf{0}).$$

Taking expectation w.r.t. all the randomness up to iteration K on both sides and using Inequality (39), then we have

$$\begin{aligned}\mathbb{E}\Big[\mathrm{dist}^2(\partial F(\hat{\boldsymbol{x}}_K), \mathbf{0})\Big] &= \frac{1}{K}\sum_{i=1}^{K}\mathbb{E}\Big[\mathrm{dist}^2(\partial F(\boldsymbol{x}_i), \mathbf{0})\Big] \\ &\leq \frac{4\Delta_0}{\eta K} + \frac{2(1-p)(n-b)\sigma^2}{pb(n-1)K} + \frac{4(n-b)\sigma^2}{b(n-1)},\end{aligned}$$

where $\Delta_0 = F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)$, thus completing the proof. □

Corollary 4.5 (Finite-sum)**.** *Choosing* $b = n$, $b' = \sqrt{n}$, *and* $p = \frac{b'}{b+b'}$, *and setting* $K = \frac{4\Delta_0}{\epsilon^2\eta}$, *we have* $\mathbb{E}\big[\mathrm{dist}^2(\partial F(\hat{\boldsymbol{x}}_K), \mathbf{0})\big] \leq \epsilon^2$ *with* $\mathcal{O}\big(nd + \frac{\Delta_0 d\sqrt{n(\hat{L}+\tilde{L})}}{\epsilon^2}\big)$ *arithmetic operations, where* $\Delta_0 = F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)$.

Proof. By the chosen parameters and Inequality (15), we have

$$\mathbb{E}\Big[\mathrm{dist}^2(\partial F(\hat{\boldsymbol{x}}_K), \mathbf{0})\Big] \leq \frac{4(F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*))}{\eta K} + \frac{2(1-p)(n-b)\sigma^2}{pb(n-1)K} + \frac{4(n-b)\sigma^2}{b(n-1)} \leq \epsilon^2.$$

Further, since $0 < \eta \leq \frac{-1+\sqrt{1+4c_0}}{2c_0}$, we have $K = \frac{4(F(\boldsymbol{x}_0)-F(\boldsymbol{x}^*))}{\epsilon^2\eta} = \frac{8c_0(F(\boldsymbol{x}_0)-F(\boldsymbol{x}^*))}{\epsilon^2(-1+\sqrt{1+4c_0})}$. Let $\mathcal{F}_{k,j-1}$ denote the natural filtration, containing all algorithm randomness up to and including outer iteration k and inner iteration $j-1$. Denote m_k^j to be the number of arithmetic operations to update the j-th block at k-th iteration, then we have for $k \geq 1$

$$\mathbb{E}[m_k^j | \mathcal{F}_{k,j-1}] = \mathcal{O}\Big((pb + (1-p)b')d^j\Big).$$

Taking expectation w.r.t to all randomness on both sides, we obtain $\mathbb{E}[m_k^j] = \mathcal{O}\Big((pb+(1-p)b')d^j\Big)$. Let m_k be the number of arithmetic operations in the k-the iteration, then we have for $k \geq 1$

$$\mathbb{E}[m_k] = \mathbb{E}\Big[\sum_{j=1}^m m_k^j\Big] = \mathcal{O}\Big((pb + (1-p)b')\sum_{j=1}^m d^j\Big) = \mathcal{O}\Big((pb + (1-p)b')d\Big).$$

Hence, the total number of arithmetic operations M in K iterations to obtain ϵ-accurate solution is

$$\mathbb{E}[M] = \mathbb{E}\Big[\sum_{k=0}^K m_k\Big] = \mathcal{O}(bd) + \mathbb{E}\Big[\sum_{k=1}^K m_k\Big] = \mathcal{O}\Big(bd + K(pb + (1-p)b')d\Big).$$

Since $b = n$, $b' = \sqrt{b}$ and $p = \frac{b'}{b+b'}$, then we have

$$K(pb + (1-p)b') = \frac{8c_0(F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*))}{\epsilon^2(-1+\sqrt{1+4c_0})}\frac{2bb'}{b+b'} \leq \frac{2(\sqrt{1+4c_0}+1)(F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*))}{\epsilon^2}2b'.$$

Note that

$$c_0 = \frac{2(1-p)\hat{L}}{pb'} + \hat{L} + 2\Big(\frac{p(n-b)}{b(n-1)} + \frac{1-p}{b'}\Big)\frac{\tilde{L}}{p} = \hat{L} + \frac{1-p}{pb'}(2\hat{L} + 2\tilde{L}) = \hat{L} + \frac{b}{b'^2}(2\hat{L} + 2\tilde{L}) = 3\hat{L} + 2\tilde{L} = \mathcal{O}(\hat{L} + \tilde{L}),$$

so we obtain

$$K(pb + (1-p)b') \leq \frac{2(\sqrt{1+4c_0}+1)(F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*))}{\epsilon^2}2b' = \mathcal{O}\Big(\frac{(F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*))\sqrt{n(\hat{L}+\tilde{L})}}{\epsilon^2}\Big),$$

thus completing the proof. □

Corollary 4.6 (Infinite-sum)**.** *Choosing* $b = \lceil\frac{12\sigma^2}{\epsilon^2}\rceil$, $b' = \sqrt{b}$, *and* $p = \frac{b'}{b+b'}$, *and setting* $K = \frac{12\Delta_0}{\epsilon^2\eta} + \frac{1}{2p}$, *we have* $\mathbb{E}\big[\mathrm{dist}^2(\partial F(\hat{\boldsymbol{x}}_K), \mathbf{0})\big] \leq \epsilon^2$ *with* $\mathcal{O}\big(bd + \frac{\Delta_0 d\sqrt{b(\hat{L}+\tilde{L})}}{\epsilon^2}\big)$ *arithmetic operations, where* $\Delta_0 = F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)$.

Proof. By the chosen parameters and Inequality (15), we have

$$\mathbb{E}\Big[\mathrm{dist}^2(\partial F(\hat{\boldsymbol{x}}_K), \mathbf{0})\Big] \leq \frac{4(F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*))}{\eta K} + \frac{2(1-p)(n-b)\sigma^2}{pb(n-1)K} + \frac{4(n-b)\sigma^2}{b(n-1)} \leq \frac{\epsilon^2}{3} + \frac{4\sigma^2}{b} + \frac{4\sigma^2}{b} \leq \epsilon^2.$$

Further, same as in the proof of Corollary 4.5, we have that the total number of arithmetic operations M in K iterations to obtain an ϵ-accurate solution is

$$\mathbb{E}[M] = \mathbb{E}\Big[\sum_{k=0}^K m_k\Big] = \mathcal{O}(bd) + \mathbb{E}\Big[\sum_{k=1}^K m_k\Big] = \mathcal{O}\Big(bd + K(pb + (1-p)b')d\Big).$$

Since $0 < \eta \leq \frac{-1+\sqrt{1+4c_0}}{2c_0}$, we have

$$K = \frac{12(F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*))}{\epsilon^2\eta} + \frac{1}{2p} = \frac{24c_0(F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*))}{\epsilon^2(-1+\sqrt{1+4c_0})} + \frac{b+b'}{2b'} = \frac{6(\sqrt{4c_0+1}+1)(F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*))}{\epsilon^2} + \frac{b+b'}{2b'},$$

which leads to

$$K(pb+(1-p)b') = \Big(\frac{6(\sqrt{4c_0+1}+1)(F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*))}{\epsilon^2} + \frac{b+b'}{2b'}\Big)\frac{2bb'}{b+b'} \leq b + \frac{12b'(\sqrt{1+4c_0}+1)(F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*))}{\epsilon^2}.$$

Since $\frac{p(n-b)}{b(n-1)} \leq \frac{p}{b} = \frac{b'}{b(b+b')} \leq \frac{b}{b'(b+b')} = \frac{1-p}{b'}$ with $b' = \sqrt{b}$ and $p = \frac{b'}{b+b'}$, we have

$$c_0 = \frac{2(1-p)\hat{L}}{pb'} + \hat{L} + 2\Big(\frac{p(n-b)}{b(n-1)} + \frac{1-p}{b'}\Big)\frac{\tilde{L}}{p} \leq \hat{L} + \frac{1-p}{pb'}(2\hat{L}+4\tilde{L}) \leq \hat{L} + \frac{b}{b'^2}(2\hat{L}+4\tilde{L}) = 3\hat{L}+4\tilde{L} = \mathcal{O}(\hat{L}+\tilde{L}).$$

Hence, we obtain

$$K(pb+(1-p)b') \leq b + \frac{12b'(\sqrt{1+4c_0}+1)(F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*))}{\epsilon^2} = \mathcal{O}\Big(b + \frac{(F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*))\sqrt{b(\hat{L}+\tilde{L})}}{\epsilon^2}\Big),$$

thus completing the proof. □

Corollary 4.7. *Suppose that F further satisfies Assumption 2.4, then we have for Algorithm 2,*

$$\begin{aligned}&\mathbb{E}[F(\boldsymbol{x}_K) - F(\boldsymbol{x}^*)]\\ &\leq \big(1+\frac{\eta\mu}{2}\big)^{-K}\Big(\Delta_0 + \frac{\sigma^2\eta(1-p)(n-b)}{pb(n-1)}\Big) + \frac{4(n-b)\sigma^2}{b\mu(n-1)},\end{aligned}$$

where $0 < \eta \leq \min\big\{\frac{p}{\mu(1-p)}, \frac{-1+\sqrt{1+4c_0}}{2c_0}\big\}$ *with* $c_0 = \hat{L} + \frac{4\hat{L}}{pb'} + \frac{4\tilde{L}}{p}\big(\frac{p(n-b)}{b(n-1)} + \frac{1-p}{b'}\big)$, *and* $\Delta_0 = F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)$.

Proof. By Lemma 4.3 and Lemma 4.2, we have

$$F(\boldsymbol{x}_k) \leq F(\boldsymbol{x}_{k-1}) - \frac{1}{2}\big(\frac{1}{\eta} - 1 - \hat{L}\eta\big)v_k + \frac{\eta}{2}u_k - \frac{\eta}{4}s_k.$$

By the PŁ condition,

$$s_k \geq 2\mu(F(\boldsymbol{x}) - F(\boldsymbol{x}^*)),$$

we obtain

$$\big(1+\frac{\eta\mu}{2}\big)(F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*)) \leq F(\boldsymbol{x}_{k-1}) - F(\boldsymbol{x}^*) - \frac{1}{2}\big(\frac{1}{\eta} - 1 - \hat{L}\eta\big)v_k + \frac{\eta}{2}u_k.$$

Denoting $r_k = F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*)$ and taking expectation with respect to all randomness on both sides, and adding $\frac{\eta}{p}\times$ (12) in Lemma 4.1, we obtain

$$\begin{aligned}\big(1+\frac{\eta\mu}{2}\big)\mathbb{E}[r_k] + \frac{\eta(2-p)}{2p}\mathbb{E}[u_k] \leq \mathbb{E}[r_{k-1}] - \Big[\frac{1}{2}\big(\frac{1}{\eta} - 1 - \hat{L}\eta\big) - \frac{2\eta\tilde{L}}{p}\big(\frac{p(n-b)}{b(n-1)} + \frac{1-p}{b'}\big)\Big]\mathbb{E}[v_k]\\ + \frac{\eta(1-p)}{p}\mathbb{E}[u_{k-1}] + \frac{2\eta\hat{L}(1-p)}{pb'}\mathbb{E}[v_{k-1}] + \frac{2\eta(n-b)\sigma^2}{b(n-1)}.\end{aligned}$$

Note that when $0 < \eta \leq \frac{p}{\mu(1-p)}$, we have $\frac{\eta(2-p)}{2p} \geq \big(1+\frac{\eta\mu}{2}\big)\frac{\eta(1-p)}{p}$ and $1 + \frac{\eta\mu}{2} \leq \frac{1}{1-p}$. On the other hand, denote

$$c_0 = \hat{L} + \frac{4\hat{L}}{pb'} + \frac{4\tilde{L}}{p}\big(\frac{p(n-b)}{b(n-1)} + \frac{1-p}{b'}\big)$$

for simplicity, then if $\eta \leq \frac{-1+\sqrt{1+4c_0}}{2c_0}$, we have

$$\big(1+\frac{\eta\mu}{2}\big)\frac{2\eta\hat{L}(1-p)}{pb'} \leq \frac{2\eta\hat{L}}{pb'} \leq \frac{1}{2}\big(\frac{1}{\eta} - 1 - \hat{L}\eta\big) - \frac{2\eta\tilde{L}}{p}\big(\frac{p(n-b)}{b(n-1)} + \frac{1-p}{b'}\big).$$

Hence, when choosing the stepsize η such that

$$0 < \eta \leq \min\Big\{\frac{p}{\mu(1-p)}, \frac{-1+\sqrt{1+4c_0}}{2c_0}\Big\},$$

we have

$$\begin{aligned}
&\big(1+\frac{\eta\mu}{2}\big)\Big[\mathbb{E}[r_k] + \frac{\eta(1-p)}{p}\mathbb{E}[u_k] + \frac{2\eta\hat{L}(1-p)}{pb'}\mathbb{E}[v_k]\Big] \\
&\leq \big(1+\frac{\eta\mu}{2}\big)\mathbb{E}[r_k] + \frac{\eta(2-p)}{2p}\mathbb{E}[u_k] + \Big[\frac{1}{2}\big(\frac{1}{\eta} - 1 - \hat{L}\eta\big) - \frac{2\eta\tilde{L}}{p}\big(\frac{p(n-b)}{b(n-1)} + \frac{1-p}{b'}\big)\Big]\mathbb{E}[v_k] \\
&\leq \mathbb{E}[r_{k-1}] + \frac{\eta(1-p)}{p}\mathbb{E}[u_{k-1}] + \frac{2\eta\hat{L}(1-p)}{pb'}\mathbb{E}[v_{k-1}] + \frac{2\eta(n-b)\sigma^2}{b(n-1)}.
\end{aligned}$$

Let $\Phi_k = r_k + \frac{\eta(1-p)}{p}u_k + \frac{2\eta\hat{L}(1-p)}{pb'}v_k$, then we obtain

$$\mathbb{E}[\Phi_k] \leq \big(1+\frac{\eta\mu}{2}\big)^{-1}\mathbb{E}[\Phi_{k-1}] + \big(1+\frac{\eta\mu}{2}\big)^{-1}\frac{2\eta(n-b)\sigma^2}{b(n-1)}.$$

Telescoping from 1 to K, we have

$$\mathbb{E}[\Phi_K] \leq \big(1+\frac{\eta\mu}{2}\big)^{-K}\mathbb{E}[\Phi_0] + \frac{4(n-b)\sigma^2}{b\mu(n-1)}.$$

As $u_K \geq 0$, $v_K \geq 0$, $\mathbb{E}[u_0] = \mathbb{E}\Big[\sum_{j=1}^m \|\nabla^j f(\boldsymbol{x}_{-1,j}) - g_{-1}^j\|^2_{\boldsymbol{\Lambda}_j^{-1}}\Big] \leq \frac{\sigma^2(n-b)}{b(n-1)}$ and $v_0 = \|\boldsymbol{x}_0 - \boldsymbol{x}_{-1}\|^2_{\boldsymbol{\Lambda}} = 0$, we have

$$\mathbb{E}[F(\boldsymbol{x}_K) - F(\boldsymbol{x}^*)] \leq \big(1+\frac{\eta\mu}{2}\big)^{-K}(F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)) + \big(1+\frac{\eta\mu}{2}\big)^{-K}\frac{\sigma^2\eta(1-p)(n-b)}{pb(n-1)} + \frac{4(n-b)\sigma^2}{b\mu(n-1)}, \tag{40}$$

thus completing the proof. □

Corollary 4.8 (Finite-sum)**.** *Choosing $b = n$, $b' = \sqrt{n}$, and $p = \frac{b'}{b+b'}$, and setting $K = (1+\frac{2}{\eta\mu})\log(\frac{\Delta_0}{\epsilon})$, we have $\mathbb{E}[F(\boldsymbol{x}_K) - F(\boldsymbol{x}^*)] \leq \epsilon$ with $\mathcal{O}\Big(nd + \big(\frac{\sqrt{n(\hat{L}+\tilde{L})}}{\mu} + n\big)d\log(\frac{\Delta_0}{\epsilon})\Big)$ arithmetic operations, where $\Delta_0 = F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)$.*

Proof. By the chosen parameters and Inequality (40), we have

$$\begin{aligned}
\mathbb{E}[F(\boldsymbol{x}_K) - F(\boldsymbol{x}^*)] &\leq \big(1+\frac{\eta\mu}{2}\big)^{-K}(F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)) + \big(1+\frac{\eta\mu}{2}\big)^{-K}\frac{\sigma^2\eta(1-p)(n-b)}{pb(n-1)} + \frac{4(n-b)\sigma^2}{b\mu(n-1)} \\
&\leq \exp\Big(-\frac{n\mu}{2+\eta\mu}K\Big)(F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)) \\
&\leq \epsilon.
\end{aligned}$$

Proceeding same as in the proof for Corollary 4.5, we have the total number of arithmetic operations M in K iterations to obtain an ϵ-accurate solution is

$$\mathbb{E}[M] = \mathbb{E}\Big[\sum_{k=0}^K m_k\Big] = \mathcal{O}(bd) + \mathbb{E}\Big[\sum_{k=1}^K m_k\Big] = \mathcal{O}\Big(bd + K(pb + (1-p)b')d\Big).$$

Since $0 < \eta \leq \min\Big\{\frac{p}{\mu(1-p)}, \frac{-1+\sqrt{1+4c_0}}{2c_0}\Big\}$, we can bound K by

$$K = \Big(1+\frac{2}{\eta\mu}\Big)\log\Big(\frac{F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)}{\epsilon}\Big) \leq \Big(1 + \frac{\sqrt{4c_0+1}+1}{\mu} + \frac{2(1-p)}{p}\Big)\log\Big(\frac{F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)}{\epsilon}\Big),$$

which leads to

$$K(pb+(1-p)b') \le \Big(1+\frac{\sqrt{4c_0+1}+1}{\mu}+\frac{2(1-p)}{p}\Big)\log\Big(\frac{F(\boldsymbol{x}_0)-F(\boldsymbol{x}^*)}{\epsilon}\Big)\frac{2bb'}{b+b'}$$
$$\le \Big(2b'+2b'\frac{\sqrt{4c_0+1}+1}{\mu}+4b\Big)\log\Big(\frac{F(\boldsymbol{x}_0)-F(\boldsymbol{x}^*)}{\epsilon}\Big).$$

Notice that

$$c_0=\hat{L}+\frac{4\hat{L}}{pb'}+\frac{4\tilde{L}}{p}\big(\frac{p(n-b)}{b(n-1)}+\frac{1-p}{b'}\big)\le \hat{L}+\frac{4}{pb'}(\hat{L}+\tilde{L})=\hat{L}+\frac{4(b+b')}{b'^2}(\hat{L}+\tilde{L})\le \hat{L}+8(\hat{L}+\tilde{L})=\mathcal{O}(\hat{L}+\tilde{L}),$$

so we obtain

$$K(pb+(1-p)b')\le\Big(2b'+2b'\frac{\sqrt{4c_0+1}+1}{\mu}+4b\Big)\log\Big(\frac{F(\boldsymbol{x}_0)-F(\boldsymbol{x}^*)}{\epsilon}\Big)=\mathcal{O}\Big(\big(\frac{\sqrt{n(\hat{L}+\tilde{L})}}{\mu}+n\big)\log\big(\frac{F(\boldsymbol{x}_0)-F(\boldsymbol{x}^*)}{\epsilon}\big)\Big),$$

thus completing the proof. □

Corollary 4.9 (Infinite-sum). *Choosing $b=\lceil\frac{12\sigma^2}{\mu\epsilon}\rceil$, $b'=\sqrt{b}$, and $p=\frac{b'}{b+b'}$, and setting $K=(1+\frac{2}{\eta\mu})\log(\frac{3\Delta_0}{\epsilon})$, we have $\mathbb{E}[F(\boldsymbol{x}_K)-F(\boldsymbol{x}^*)]\le\epsilon$ with $\mathcal{O}\Big(bd+\big(\frac{\sqrt{b(\hat{L}+\tilde{L})}}{\mu}+b\big)d\log(\frac{\Delta_0}{\epsilon})\Big)$ arithmetic operations, where $\Delta_0=F(\boldsymbol{x}_0)-F(\boldsymbol{x}^*)$.*

Proof. By the chosen parameters and Inequality (40), we have

$$\mathbb{E}[F(\boldsymbol{x}_K)-F(\boldsymbol{x}^*)]\le\big(1+\frac{\eta\mu}{2}\big)^{-K}(F(\boldsymbol{x}_0)-F(\boldsymbol{x}^*))+\big(1+\frac{\eta\mu}{2}\big)^{-K}\frac{\sigma^2\eta(1-p)(n-b)}{pb(n-1)}+\frac{4(n-b)\sigma^2}{b\mu(n-1)}$$
$$\le \exp\Big(-\frac{n\mu}{2+\eta\mu}K\Big)(F(\boldsymbol{x}_0)-F(\boldsymbol{x}^*))+\exp\Big(-\frac{n\mu}{2+\eta\mu}K\Big)\frac{\sigma^2\eta(1-p)(n-b)}{pb(n-1)}+\frac{4(n-b)\sigma^2}{b\mu(n-1)}$$
$$\overset{(i)}{\le}\frac{\epsilon}{3}+\frac{\epsilon}{3}+\frac{\epsilon}{3}=\epsilon,$$

where for (i) we use $\eta\le\frac{p}{\mu(1-p)}$, thus

$$\exp\Big(-\frac{n\mu}{2+\eta\mu}K\Big)\frac{\sigma^2\eta(1-p)(n-b)}{pb(n-1)}\le\frac{\sigma^2\eta(1-p)(n-b)}{pb(n-1)}\le\frac{\eta(1-p)\mu\epsilon}{12p}\le\frac{\epsilon}{12}.$$

With the same process as in the proof for Corollary 4.5, we have the total number of arithmetic operations M in K iterations to obtain an ϵ-accurate solution is

$$\mathbb{E}[M]=\mathbb{E}\Big[\sum_{k=0}^{K}m_k\Big]=\mathcal{O}(bd)+\mathbb{E}\Big[\sum_{k=1}^{K}m_k\Big]=\mathcal{O}\Big(bd+K(pb+(1-p)b')d\Big).$$

Since $0<\eta\le\min\Big\{\frac{p}{\mu(1-p)},\frac{-1+\sqrt{1+4c_0}}{2c_0}\Big\}$, we can bound K by

$$K=\Big(1+\frac{2}{\eta\mu}\Big)\log\Big(\frac{3(F(\boldsymbol{x}_0)-F(\boldsymbol{x}^*))}{\epsilon}\Big)\le\Big(1+\frac{\sqrt{4c_0+1}+1}{\mu}+\frac{2(1-p)}{p}\Big)\log\Big(\frac{3(F(\boldsymbol{x}_0)-F(\boldsymbol{x}^*))}{\epsilon}\Big),$$

which leads to

$$K(pb+(1-p)b')\le\Big(1+\frac{\sqrt{4c_0+1}+1}{\mu}+\frac{2(1-p)}{p}\Big)\log\Big(\frac{3(F(\boldsymbol{x}_0)-F(\boldsymbol{x}^*))}{\epsilon}\Big)\frac{2bb'}{b+b'}$$
$$\le\Big(2b'+2b'\frac{\sqrt{4c_0+1}+1}{\mu}+4b\Big)\log\Big(\frac{3(F(\boldsymbol{x}_0)-F(\boldsymbol{x}^*))}{\epsilon}\Big).$$

Notice that $\frac{p(n-b)}{b(n-1)} \leq \frac{p}{b} = \frac{b'}{b(b+b')} \leq \frac{b}{b'(b+b')} = \frac{1-p}{b'}$, and thus

$$c_0 = \hat{L} + \frac{4\hat{L}}{pb'} + \frac{4\tilde{L}}{p}\left(\frac{p(n-b)}{b(n-1)} + \frac{1-p}{b'}\right) \leq \hat{L} + \frac{4}{pb'}(\hat{L} + 2\tilde{L}) = \hat{L} + \frac{4(b+b')}{b'^2}(\hat{L} + 2\tilde{L}) = \hat{L} + 8(\hat{L} + 2\tilde{L}) = \mathcal{O}(\hat{L} + \tilde{L}).$$

Hence, we obtain

$$K(pb + (1-p)b') \leq \left(2b' + 2b'\frac{\sqrt{4c_0+1}+1}{\mu} + 4b\right)\log\left(\frac{3(F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*))}{\epsilon}\right) = \mathcal{O}\left(\left(\frac{\sqrt{b(\hat{L}+\tilde{L})}}{\mu} + b\right)\log\left(\frac{\Delta_0}{\epsilon}\right)\right),$$

thus completing the proof. □

D. Additional Experiments and Discussion

We first present the LeNet architectures used in our experiments for MNIST and CIFAR-10 datasets as in Fig. 2.

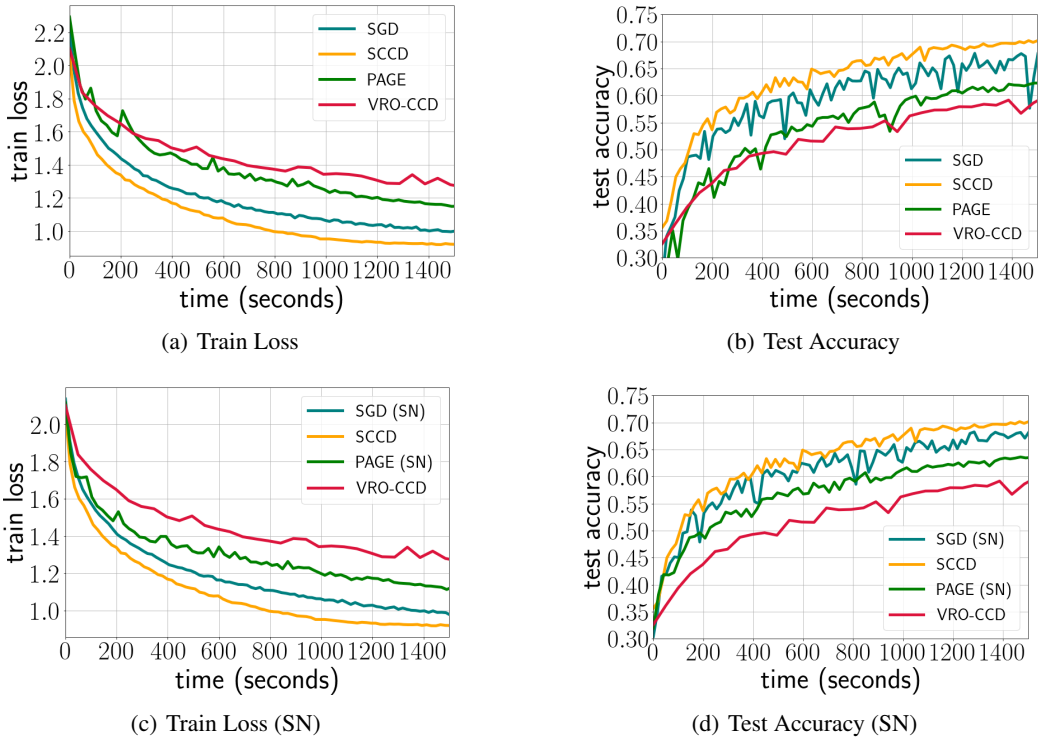
```
# LeNet
class LeNet(nn.Module):
    def __init__(self):
        super(LeNet, self).__init__()
        # For MNIST dataset
        # self.conv1 = nn.Conv2d(1, 6, 5)
        # self.conv2 = nn.Conv2d(6, 16, 5)
        # self.fc1   = nn.Linear(256, 120)
        # self.fc2   = nn.Linear(120, 84)
        # self.fc3   = nn.Linear(84, 10)

        # For CIFAR-10 dataset
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1   = nn.Linear(16*5*5, 120)
        self.fc2   = nn.Linear(120, 84)
        self.fc3   = nn.Linear(84, 10)

    def forward(self, x):
        out = F.relu(self.conv1(x))
        out = F.max_pool2d(out, 2)
        out = F.relu(self.conv2(out))
        out = F.max_pool2d(out, 2)
        out = out.view(out.size(0), -1)
        out = F.relu(self.fc1(out))
        out = F.relu(self.fc2(out))
        out = self.fc3(out)
        return out
```

Figure 2. LeNet architectures used for MNIST and CIFAR-10 datasets.

We then re-plot the train loss and test accuracy in Fig. 1 against wall-clock time based on the average runtime per epoch of each algorithm in Table 1, as is shown in Fig. 3 below. We observe that (i) SCCD still converges faster to solutions with better generalization, in comparison with SGD and PAGE (whether spectral normalized or not); (ii) VRO-CCD converges slower in terms of wall-clock time, which is due to that cyclic updates become major computation bottleneck using small batch size. Some other causes can be sampling p and additional batch operations to form $\mathcal{B}$ and $\mathcal{B}'$ in each iteration for PAGE estimator.



(a) Train Loss

(b) Test Accuracy

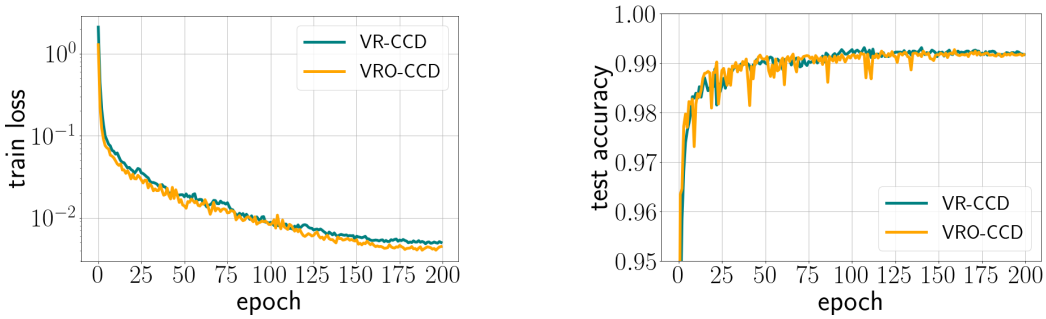
(c) Train Loss (SN)

(d) Test Accuracy (SN)

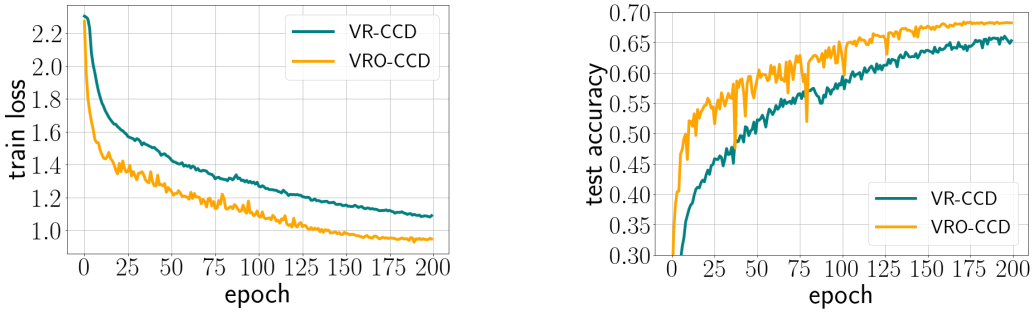
Figure 3. Comparison of SGD, SCCD, PAGE and VRO-CCD in wall-clock time on training LeNet on CIFAR-10.

We also compare the empirical performance of VR-CCD and VRO-CCD on the MNIST and CIFAR-10 datasets in Fig. 4, in one run and without spectral normalization. We set batch size $b = 64$ for the MNIST dataset and $b = 256$ for the CIFAR-10 dataset, and run for 200 epochs. We still use the cosine learning rate scheduler, which is tuned for VR-CCD and VRO-CCD separately. We can see that (i) VR-CCD and VRO-CCD exhibits similar performance on the MNIST dataset; (ii) the empirical convergence of VR-CCD is slower than VRO-CCD's on the CIFAR-10 dataset, due to the smaller learning rate used for VR-CCD. We remark that separate sampling for each block may lead to numerical instability of the algorithm, thus requiring smaller learning rate for more complicated problems. Also, separate sampling for each block increases the sample complexity and introduces additional computational cost.

Finally, we provide a further comparison between SCCD, VRO-CCD, SGD and PAGE algorithms with the same learning rate to illustrate the efficacy of variance reduction methods, motivated by the experimental setup in Li et al. (2021). We use the initial learning rate $\eta_{\text{ini}} = 0.01$ in cosine scheduler for all the algorithms except PAGE without spectral normalization which only admits $\eta_{\text{ini}} = 0.005$ most. We can see that SCCD and VRO-CCD demonstrate better performance than SGD and PAGE do, respectively. Here SCCD stagnates earlier than SGD, because the learning rate with cosine scheduler is too small for each block in SCCD to make progress.

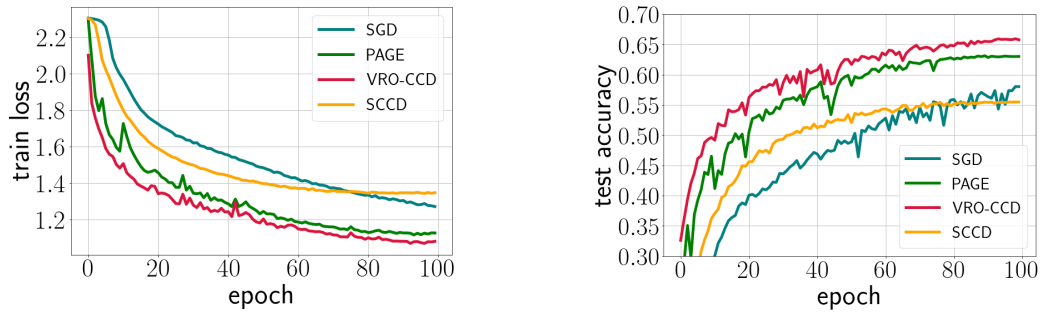


(a) Train Loss (MNIST) (b) Test Accuracy (MNIST)

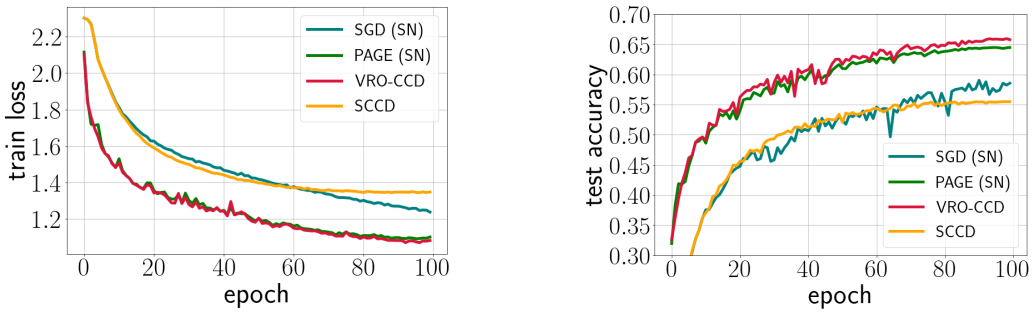


(c) Train Loss (CIFAR-10) (d) Test Accuracy (CIFAR-10)

Figure 4. Comparison of VR-CCD and VRO-CCD on training LeNet on MNIST and CIFAR-10.



(a) Train Loss (b) Test Accuracy



(c) Train Loss (SN) (d) Test Accuracy (SN)

Figure 5. Comparison of SGD, SCCD, PAGE and VRO-CCD with same learning rate on training LeNet on CIFAR-10.