GraB: Finding Provably Better Data Permutations than Random Reshuffling

Yucheng Lu, Wentao Guo, Christopher De Sa
Department of Computer Science
Cornell University
{yl2967, wg247, cmd353}@cornell.edu

Abstract

Random reshuffling, which randomly permutes the dataset each epoch, is widely adopted in model training because it yields faster convergence than with-replacement sampling. Recent studies indicate greedily chosen data orderings can further speed up convergence empirically, at the cost of using more computation and memory. However, greedy ordering lacks theoretical justification and has limited utility due to its non-trivial memory and computation overhead. In this paper, we first formulate an example-ordering framework named herding and answer affirmatively that SGD with herding converges at the rate $O\left(T^{-2/3}\right)$ on smooth, non-convex objectives, faster than the $O\left(n^{1/3}T^{-2/3}\right)$ obtained by random reshuffling, where $n$ denotes the number of data points and $T$ denotes the total number of iterations. To reduce the memory overhead, we leverage discrepancy minimization theory to propose an online Gradient Balancing algorithm (GraB) that enjoys the same rate as herding, while reducing the memory usage from $O(nd)$ to just $O(d)$ and computation from $O(n^2)$ to $O(n)$, where $d$ denotes the model dimension. We show empirically on applications including MNIST, CIFAR10, WikiText and GLUE that GraB can outperform random reshuffling in terms of both training and validation performance, and even outperform state-of-the-art greedy ordering while reducing memory usage over $100\times$.

1 Introduction

Many machine learning problems can be formulated as minimizing a differentiable (loss) function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ for a set of data examples $\{x_i\}_{i=1}^n$. To train a parameterized model, the goal is to obtain a set of target parameters $w^* = \arg \min f(w)$, where $f(w) = \frac{1}{n} \sum_{i=1}^n f(w; x_i)$, and $f(w; x_i)$ denotes the loss incurred on the $i$-th data example $x_i$ (usually a mini-batch of images, sentences, etc.) with model parameters $w$. A typical model training, or optimization process, is to iteratively update the model parameter $w$ starting from some initial $w^{(1)}$ by running

$$w^{(t+1)} = w^{(t)} - \alpha \nabla f(w^{(t)}; x_{\sigma(t)}) \quad t = 1, 2, \cdots$$

where $\alpha$ denotes the step size, and $\sigma : \{1, \ldots, n\} \rightarrow \{1, \ldots, n\}$ denotes a permutation (ordering) from which the examples are chosen to compute the stochastic gradients\(^2\). A widely adopted ordering protocol is Random Reshuffling (RR), where the optimizer scans in an order drawn at random without replacement over the entire training dataset multiple times during training (a common name for a single such scan is an “epoch”). RR allows the optimizer to converge faster empirically and enjoys a

\(^1\)The experimental code is available at https://github.com/EugeneLYC/GraB.

\(^2\)In this paper, we consistently use the term stochastic gradients to refer to gradients computed on a single (or a set of) data example(s), even when those examples are not selected at random.
better convergence rate in theory [1]. Despite RR’s theoretical improvement, it has been proven that RR does not always guarantee a good ordering [2, 3]; in fact a random permutation is far from being optimal even when optimizing a simple quadratic objective [4]. In light of this, a natural research question is:

*Can we find provably better orderings than Random Reshuffling—and do so efficiently?*

Recent studies indicate the possibility of greedily constructing better permutations using stale estimates of each \( \nabla f(w; x_j) \) from the previous epoch [5, 6]. Concretely, Lu et al. [5] proved that for any model parameters \( w \in \mathbb{R}^d \), if sums of consecutive stochastic gradients converge faster to the full gradient, then the optimizer will converge faster. Formally, in one epoch, any permutation \( \sigma \) minimizing the term (named average gradient error)

\[
\max_{k \in \{1, \ldots, n\}} \left\| \sum_{t=1}^{k} \left( \nabla f(w; x_{\sigma(t)}) - \nabla f(w) \right) \right\|
\]

leads to fast convergence. Leveraging this insight, Lu et al. [5] proposes to greedily select \( \sigma(t) \) one at a time for all \( t \in \{1, \ldots, n\} \) at the beginning of the \((k+1)\)-th epoch, using the stochastic gradients computed during the \(k\)-th epoch as an estimate. This strategy works well empirically, but it remains an open question how its convergence rate compares to that of RR. Moreover, the greedy ordering method intensively consumes \( O(nd) \) memory to store the gradients and \( O(n^2) \) computation to order, which significantly hinders its usefulness in practice. For instance, training a simple logistic regression on MNIST would easily cost more than 1 GB additional memory compared to RR.

In this paper, we address the limitations of previous order-selection approaches, culminating in proposing a new algorithm, GraB, which converges faster than RR both in theory and in practice without any blow-up in memory or computational time. First, we address the theoretical gap by connecting the convergence of permuted-order SGD to the classic herding problem [7]: we show how any algorithm for solving the herding problem can be run on stale stochastic gradients to select an example ordering. While the greedy selection method of previous work may fail to select a good ordering, using the best herding algorithms from the literature yields a better convergence rate: \( O(T^{-2/3}) \) compared to RR’s \( O(n^{1/3}T^{-2/3}) \) on smooth non-convex problems and \( \tilde{O}(T^{-2}) \) compared to RR’s \( \tilde{O}(nT^{-2}) \) under PL condition, where \( T \) is the total number of iterations. Building on this, we leverage discrepancy minimization theory [8] to propose an online Gradient Balancing algorithm (GraB) that achieves the same \( O(T^{-2/3}) \) rate but only requires \( O(d) \) memory and \( O(n) \) computation. Perhaps surprisingly, on multiple deep learning applications we observe GraB not only allows fast minimization of the empirical risk, but also lets the model generalize better. Our contributions in this paper can be summarized as follows:

- We formulate a general framework of sorting stochastic gradients with herding, and prove SGD with herding converges as a faster rate than Random Reshuffling (Section 3).
- We propose an online Gradient Balancing (GraB) algorithm that enjoys the same fast convergence rate as herding SGD while significantly reducing the memory from \( O(nd) \) to \( O(d) \) and computation from \( O(n^2) \) to \( O(n) \) (Section 5).
- We conduct extensive experiments on MNIST, CIFAR10, WikiText and GLUE with different models. We demonstrate that GraB converges faster in terms of both training and validation compared to Random Reshuffling and other baselines (Section 6).

2 Related Work

While traditional data ordering research mostly focuses on with-replacement samplers [9–11], without-replacement sampling is more common in practice [12]. Random Reshuffling (RR) [13] and the related shuffle-once (SO) method [14, 15] are among the most popular data permutation methods. Recht and Ré [16] undertakes the first theoretical investigation of RR, while subsequent works like [2, 17] give counter examples where RR orders badly. HaoChen and Sra [18], Gürbüzbalaban et al. [19], and Mishchenko et al. [1] discuss extensively on the conditions needed for RR to benefit. Some recent works [5, 6] suggest constructing better data permutations than RR via a memory-intensive greedy strategy. Concretely, Mohtashami et al. [6] proposes evaluating algorithms on all the examples first to minimize Equation (2) before starting an epoch, applied to Federated Learning; Lu et al. [5]
Algorithm 1 Herding with Greedy Ordering [5]

1: Input: a group of vectors \( \{z_i\}_{i=1}^n \).
2: Center all the vectors: \(z_i \leftarrow z_i - \frac{1}{n} \sum_{j=1}^{n} z_j, \forall i \in [n] \).
3: Initialize an arbitrary \( \sigma \), running partial sum: \( s \leftarrow 0 \), candidate set \( \Phi \leftarrow \{1, \ldots, n\} \).
4: for \( i = 1, \ldots, n \) do
5: 
6: 
7: end for
8: return \( \sigma \).

Algorithm 2 General Framework of SGD with Offline Herding

1: Input: number of epochs \( K \), initialized order \( \sigma_1 \), initialized weight \( w_1 \), learning rate \( \alpha \).
2: for \( k = 1, \ldots, K \) do
3: 
4: Compute gradient \( \nabla f(w_k^{(t)}; x_{\sigma_k(t)}) \).
5: Store the gradient: \( z_i \leftarrow \nabla f(w_k^{(t)}; x_{\sigma_k(t)}) \).
6: Optimizer Step: \( w_k^{(t+1)} \leftarrow w_k^{(t)} - \alpha \nabla f(w_k^{(t)}; x_{\sigma_k(t)}) \).
7: end for
8: Generate new order: \( \sigma_{k+1} \leftarrow \text{Herding}(\{z_i\}_{i=1}^n) \).
9: \( w_{k+1}^{(t)} \leftarrow w_k^{(t+1)} \).
10: end for
11: return \( w_k^{(1)} \).

provides an alternative of minimizing Equation (2) using stale gradients from previous epoch to estimate the gradient on each example. Rajput et al. [4] introduces an interesting variant to RR by reversing the ordering every other epoch, achieving better rates for quadratics. Other approaches, such as curriculum learning [20], try to order the data to mimic human learning and improve generalization [21–23]: these approaches differ from our goal of finding good permutations for minimizing loss in a finite-sum setting.

3 Offline Stale-Gradient Herding

In this section, we study the use of *stale gradients*—the stochastic gradients for each example from the previous epoch, saved in memory—to construct a data permutation at the start of each epoch. Lu et al. [5] and Mohtashami et al. [6] propose to use greedy ordering in this way—selecting examples one at a time to minimize Equation (2). Here, we first show that this objective is closely related to the classic herding problem from discrepancy theory. We demonstrate with examples from the herding formulation that in the adversarial setting, greedy selection is bad at herding and can underperform basic random reshuffling. We conclude this section by proving SGD with proper herding converges faster than random reshuffling.

**Herding.** The herding problem originates from Welling [24] for sampling from a Markov random field that agrees with a given data set. Its discrete version is later formulated in Harvey and Samadi [7]. Given \( n \) vectors \( \{z_i\}_{i=1}^n \) that are \( d \)-dimensional (i.e. in \( \mathbb{R}^d \)) and have norm \( \|z_i\|_2 \leq 1 \), the herding problem we study is the task of finding a permutation \( \sigma^* : [n] \rightarrow [n] \) that minimizes

\[
\max_{k \in \{1, \ldots, n\}} \left\| \sum_{i=1}^{k} \left( z_{\sigma(i)} - \frac{1}{n} \sum_{i=1}^{n} z_i \right) \right\|_\infty. \tag{3}
\]

State-of-the-art algorithms guarantee an \( \tilde{O}(1) \) bound to this objective [7]. It is straightforward to observe the the formulation of Equation (3) generalizes Equation (2) if we replace each \( z_i \) with the stochastic gradient computed on the example \( i \). In other words, any algorithm that minimizes Equation (3) can be used to minimize Equation (2) and find a better data permutation. The simplest way to do this is with the greedy algorithm for herding, which we show in Algorithm 1: this is

\footnote{A small distinction is Harvey and Samadi [7] investigates infinite sequences while here we are studying a finite ordering, i.e. a permutation.}
The averaged loss function is also
\[ L \]
We start by stating some assumptions for non-convex optimization, as well as our herding bound.

\[ \text{Assumption 4.} \quad \left(\text{worst-case guarantee, we propose using some other herding algorithm instead. There have been many}\right) \]

Algorithm 2, and could be contrasted with a fresh-gradient approach (as in Mohtashami et al. \[6\]) that

\[ \text{Herding} \]
\[ \text{Greedy ordering of Lu et al. \cite{5} can be described as running Algorithm 2 using Algorithm 1}\]

\[ \text{Herding SGD with Stale Gradients.} \]

There exist\[ \text{algorithms proposed that can reduce the Equation (3) to}\]

\[ \text{The greedy ordering can}\]

\[ \text{Remark on the assumptions.} \]

Assumptions 1, 2, and 3 are commonly used assumptions in the study of permutation-based SGD, except that we separate different smoothness constants with respect to different norms to tighten the final bound. The cross-norm Lipschitz constant \[L_{2,\infty}\] is often used to obtain tight dimension-independent bounds \[\cite{27, 28}\]. Note that these cross-norm assumptions can

Table 1: Table summarizing the theoretical results in this paper, where \(n\) denotes the total number of data points (examples), \(d\) denotes the model dimension and \(T\) denotes the total number of iterations (\(n\) times the number of epochs).

<table>
<thead>
<tr>
<th></th>
<th>Rate (Non-convex)</th>
<th>Rate (PL)</th>
<th>Storage over RR</th>
<th>Computation over RR</th>
</tr>
</thead>
<tbody>
<tr>
<td>RR</td>
<td>(O(n^\frac{1}{2}T^{-\frac{1}{4}}))</td>
<td>(O(nT^{-\frac{1}{2}}))</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Herding</td>
<td>(O(T^{-\frac{2}{3}}))</td>
<td>(O(T^{-\frac{2}{3}}))</td>
<td>(O(n^2))</td>
<td>(O(nd))</td>
</tr>
<tr>
<td>GraB</td>
<td>(O(T^{-\frac{2}{3}}))</td>
<td>(O(T^{-\frac{2}{3}}))</td>
<td>(O(n))</td>
<td>(O(d))</td>
</tr>
</tbody>
</table>

essentially what the previous work was doing to select an order based on stale gradients. Unfortunately, despite its elegance, it was first pointed out in Chelidze et al. \cite{25} that the greedy ordering can underperform random permutation. We adapt their result to show the following.

**Statement 1.** There exist \(n\) vectors in \(\mathbb{R}^2\) such that when applying Algorithm 1, the objective in Equation (2) becomes \(\Omega(n)\); a random permutation is guaranteed to achieve \(O(\sqrt{n})\) on average.

**Herding SGD with Stale Gradients.** Based on the herding framework, a path to SGD with better data permutation becomes clear: when training models during a epoch, we can simply store all the stochastic gradients, and then herd them (with some herding algorithm) offline at the start of the next epoch to obtain the order for that epoch. This stale-gradient approach is formally described in Algorithm 2, and could be contrasted with a fresh-gradient approach (as in Mohtashami et al. \cite{6}) that herded with newly computed stochastic gradients at the start of each epoch—it is desirable to avoid the use of fresh gradients if possible as they double the gradient computations needed each epoch. The greedy ordering of Lu et al. \cite{5} can be described as running Algorithm 2 using Algorithm 1 as the herding subroutine. However, as greedy herding with Algorithm 1 does not have a good worst-case guarantee, we propose using some other herding algorithm instead. There have been many algorithms proposed that can reduce the Equation (3) to \(\tilde{O}(1)\) in polynomial time. Here, we show that this herding-objective bound is sufficient to prove convergence of herded SGD and even yield a better convergence rate than Random Reshuffling. We defer the details of the herding algorithm to Section 5, as they do not affect the convergence rate of SGD.

We start by stating some assumptions for non-convex optimization, as well as our herding bound.

**Assumption 1. (Smoothness.)** For any example \(x_j\) in the dataset (for \(j \in \{1, \ldots, n\}\)), the loss for \(x_j\) is \(L_{2,\infty}\)-smooth and \(L_{\infty}\)-smooth meaning that, for any \(w, \nu \in \mathbb{R}^d\), it holds that

\[\begin{align*}
\| \nabla f(w; x_j) - \nabla f(w; x_j') \|_2 & \leq L_{2,\infty} \| w - \nu \|_\infty \quad \text{and} \\
\| \nabla f(w; x_j) - \nabla f(w; x_j') \|_\infty & \leq L_{\infty} \| w - \nu \|_\infty.
\end{align*}\]

The averaged loss function is also \(L\)-smooth in that, for any \(w, \nu \in \mathbb{R}^d\), it holds that

\[\| \nabla f(w) - \nabla f(\nu) \|_2 \leq L \| w - \nu \|_2.\]

**Assumption 2. (PL condition.)** We say the loss function \(f\) fulfills the Polyak-Lojasiewicz (PL) condition if there exists \(\mu > 0\) such that for any \(w \in \mathbb{R}^d\),

\[\frac{1}{2} \| \nabla f(w) \|_2^2 \geq \mu (f(w) - f^*), \quad \text{where} \quad f^* = \inf_{w \in \mathbb{R}^d} f(w).\]

**Assumption 3. (Bounded Gradient Error.)** For any \(j \in \{1, \ldots, n\}\) and any \(w \in \mathbb{R}^d\),

\[\| \nabla f(w; x_j) - \frac{1}{n} \sum_{s=1}^n \nabla f(w; x_s) \|_2 \leq \varsigma.\]

**Assumption 4. (Herding Bound.)** In Algorithm 2, the subroutine \(\text{Herding}\) algorithm has the following property \cite{26}: if given input vectors \(z_1, \ldots, z_n \in \mathbb{R}^d\) that satisfy \(\| z_i \|_2 \leq 1\) and \(\sum_{i=1}^n z_i = 0\), \(\text{Herding}\) outputs a permutation \(\sigma\) of \(\{1, \ldots, n\}\) such that for all \(k \in \{1, \ldots, n\}\),

\[\| \sum_{i=1}^k z_{\sigma(i)} \|_\infty \leq H \text{ for some constant } H.\]

**Remarks on the assumptions.** Assumptions 1, 2, and 3 are commonly used assumptions in the study of permutation-based SGD, except that we separate different smoothness constants with respect to different norms to tighten the final bound. The cross-norm Lipschitz constant \(L_{2,\infty}\) is often used to obtain tight dimension-independent bounds \cite{27, 28}. Note that these cross-norm assumptions can
Figure 1: (a) illustrates how Algorithm 3 reorders the vectors with balancing – the new order is obtained by concatenating original order of the examples with +1, followed by the reverse order of the examples with −1; (b) demonstrates a toy example on \( n = 10000 \) vectors \( \{z_i\}_{i=1}^{n} \) where \( z_i \) is randomly sampled from \([0, 1]^{128}\). For a given order \( \sigma \), we plot the norm of the prefix sums \( \|\sum_{t=1}^{k} (z_{\sigma(t)} - 1/n \sum_{s=1}^{n} z_s)\| \) for all \( k = 1, \ldots, n \).

be dropped by incurring extra factors of the dimension \( d \), as \( L_\infty \leq L_{2,\infty} \leq \sqrt{d}L \) for any function. Assumption 4 holds for multiple algorithms for some \( H = \tilde{O}(1) \); we will discuss this in depth in Section 5, while treating \( H \) as a parameter here as we want to observe the dependence on this variable in the final bound. Under these assumptions, the convergence rate of Algorithm 2 is given in the following theorem.

**Theorem 1.** In Algorithm 2, under Assumptions 1, 3 and 4, if we set \( \alpha \) to be

\[
\alpha = \min \left\{ \frac{1}{24nH^2\zeta^2L_{2,\infty}^2K^2}, \frac{1}{8n(L + L_{2,\infty})}, \frac{1}{32nL_{\infty}}, \frac{1}{16HL_{2,\infty}} \right\}
\]

then it converges at the rate

\[
\frac{1}{K} \sum_{k=1}^{K} \|\nabla f(w_k)\|^2 \leq 36\sqrt{\frac{H^2\zeta^2L_{2,\infty}^2F^2}{n^2K^2}} + \frac{\zeta^2}{K} + \frac{32F(L + L_{2,\infty} + L_{\infty})}{K} + \frac{64FH L_{2,\infty}}{nK},
\]

where \( F = f(w_1) - f^* \). Furthermore, under the additional PL condition (Assumption 2), with \( \alpha \) as

\[
\alpha = \min \left\{ \frac{1}{n\mu}, \frac{1}{48HL_{2,\infty}}, \frac{1}{96n(L + L_{2,\infty} + L_{\infty})}, \frac{2}{n\mu K}W_0 \left( \frac{(f(w_1) - f^* + \zeta^2)n^2\mu^3K^2}{192H^2L_{2,\infty}^2\zeta^2} \right) \right\},
\]

where \( W_0 \) denotes the Lambert W function, Algorithm 2 converges at the rate

\[
f(w_K) - f^* \leq \tilde{O} \left( \frac{H^2L_{2,\infty}^2\zeta^2}{\mu^4n^2K^2} \right).
\]

**Improved Rate Compared to RR.** Theorem 1 gives us a better rate than random reshuffling, in that there is no dependence on the number of examples \( n \) in the dominant term. If we let \( T = nK \) be the total number of steps, then Theorem 1 shows Algorithm 2 converges at rate \( \tilde{O}(\frac{1}{T^{2/3}} + \frac{n}{T^2}) \). In contrast, random reshuffling gives a rate of \( \tilde{O}(\frac{n^{1/3}}{T^{1/3}} + \frac{n}{T^2}) \) as given in Mishchenko et al. [1]. Theorem 4. Similarly, herding improves the rate from \( \tilde{O}(\frac{n}{T^2}) \) to \( \tilde{O}(\frac{1}{T^2}) \) in the PL case. This gives us an affirmative answer to our question from the intro: a faster rate can be obtained even if we herd with stale gradients estimates, without computing additional gradients.

### 4 Enabling Memory-Efficient Herding with Balancing

In Section 3 we introduce the herding problem, and prove that with proper herding, SGD converges at a faster rate than Random Reshuffling. However, a downside of offline herding SGD is that it
Algorithm 3 Reordering Vectors based on Balanced Signs [7]

1: Input: A group of signs \( \{\epsilon_i\}_{i=1}^n \) for \( \{z_i\}_{i=1}^n \) from a balancing algorithm, initial order \( \sigma \)
2: Initialize two order-sensitive lists \( L_{\text{positive}} \leftarrow \emptyset \), \( L_{\text{negative}} \leftarrow \emptyset \).
3: for \( i = 1, \ldots, n \) do
4: \hspace{1em} Append \( \sigma(i) \) to \( L_{\text{positive}} \) if \( \epsilon_i \) is +1 else append it to \( L_{\text{negative}} \).
5: end for
6: return new order \( \sigma' = \text{concatenate}(L_{\text{positive}}, \text{reverse}(L_{\text{negative}})) \).

requires storing all the vectors (i.e., in the case of SGD, all the stochastic gradients) which incurs \( O(nd) \) memory overhead. In this section, we present background on the vector balancing problem, the solutions of which will enable us to solve the herding problem in an online fashion.

Background: Online Vector Balancing. The problem of online vector balancing was first formulated in Spencer [29]: given \( n \) vectors \( \{z_i\}_{i=1}^n \in \mathbb{R}^d \), arriving one at a time, the goal is to assign a sign \( \epsilon_i \in \{-1, +1\} \) to each vector upon receiving it so as to minimize \( \max_{k \in [n]} \| \sum_{i=1}^k \epsilon_i z_i \|_\infty \). Over the last few decades, many methods have been developed to bound this \( \ell_\infty \)-norm objective. Specifically, Auer et al. [30] and Bansal and Spencer [31] show that it can be made to be on the order of \( \tilde{O}(1) \) when the coordinates of each vector are independent. Bansal et al. [32] give a poly\((\log(n), d)\) bound when there are dependencies among coordinates. The adversarial setting with interval discrepancy is discussed in [33]. Other works focus on balancing variants such as stochastic arrival [34] or box discrepancy [35].

Solving Herding via Balancing. While the objectives of balancing and herding are different, an algorithm for balancing can be used as a subroutine to do herding as well. Consider the simple case of vectors \( z_1, \ldots, z_n \) in some order, where \( \sum_{i=1}^n z_i = 0 \), and suppose we have obtained a group of signs \( \{\epsilon_i\}_{i=1}^n \) via some balancing algorithm: can we use this to determine an order of \( \{z_i\}_{i=1}^n \) that can solve the herding problem? Harvey and Samadi [7], Chobanyan et al. [36], and Chobanyan [37] approach this problem by a clever reordering that works as in Algorithm 3. Concretely, a new order is obtained by first taking all the vectors with positive signs in order, followed by all the vectors with negative signs in reversed order from the original sequence. This process is visualized in Figure 1. Harvey and Samadi [7] shows that this balancing-and-then-reordering has the following effect on the herding objective in Equation (3).

Theorem 2. ([7, Theorem 10]) Consider \( n \) vectors \( z_1, \ldots, z_n \in \mathbb{R}^d \) that fulfill \( \sum_{i=1}^n z_i = 0 \) and \( \|z_i\| \leq 1 \). Suppose that for some permutation \( \sigma \), the herding objective is bounded by \( \max_k \| \sum_{i=1}^k z_{\sigma(i)} \|_\infty \leq H \) for some constant \( H \). Suppose we have computed some signs \( \epsilon_1, \ldots, \epsilon_n \) such that the vector-balancing objective \( \max_k \| \sum_{i=1}^k \epsilon_i z_{\sigma(i)} \|_\infty \leq A \) for some constant \( A \). Then if we let \( \sigma' \) be the permutation output by Algorithm 3, the herding objective on this new permutation is bounded by \( \max_k \| \sum_{i=1}^k z_{\sigma'(i)} \|_\infty \leq \frac{(A + H)}{2} \).

Theorem 2 shows that calling Algorithm 3 allows us to construct a new ordering from balancing, which reduces the current herding bound by a constant factor if \( A \leq H \). This means that if we have a balancing algorithm that is guaranteed to reduce the balancing objective to less than \( A \), by applying this “reordering” many times, we can also reduce the herding objective to be less than \( A \); this approach is used in many offline herding algorithms that achieve \( O(1) \) bounds.

5 GraB: SGD with Gradient Balancing for Good Data Permutation

Building on the balance-then-reorder paradigm from the previous section, in this section we propose an algorithm named GraB that improves SGD with herding by choosing the ordering online—without storing gradients—thus using only \( O(d) \) memory and \( O(n) \) computation. However, the conditions of applying the balancing-reordering framework bring up additional challenges in this online setting:

- **Challenge I.** To apply Theorem 2, we will need to pre-center all the vectors (as in line 2 of Algorithm 1) to ensure they sum to zero. This cannot be done online, as we need to make decisions on-the-fly before seeing all the vectors.
- **Challenge II.** The reordering strategy, as shown in Theorem 2, only reduces the herding bound by a constant factor, which does not give us the accelerated rate directly since applying Theorem 1
Algorithm 4 SGD with Online Gradient Balancing (GraB)

1: Input: number of epochs $K$, initialized order $\sigma_1$, initialized weight $w_1$, stale mean $m_1 = 0$, step size $\alpha$.
2: for $k = 1, \ldots, K$ do
3:   Initialized indices and running average: $l \leftarrow 1$; $r \leftarrow n$; $s \leftarrow 0$; $m_{k+1} \leftarrow 0$.
4:   for $t = 1, \ldots, n$ do
5:     Compute gradient $\nabla f(w_k^t; x_{\sigma_k(t)})$.
6:     Compute centered gradient and update stale mean:
7:     
8:     \[ g_{k,t} \leftarrow \nabla f(w_k^t; x_{\sigma_k(t)}) - m_k \]
9:     \[ m_{k+1} \leftarrow m_{k+1} + \nabla f(w_k^t; x_{\sigma_k(t)})/n \]
10: Compute sign for the current gradient: $\epsilon_{k,t} \leftarrow \text{Balancing}(s, g_{k,t})$.
11: if $\epsilon_{k,t} = +1$ then
12:   $s \leftarrow s + g_{k,t}; \sigma_{k+1}(l) \leftarrow \sigma_k(t); l \leftarrow l + 1$.
13: else
14:   $s \leftarrow s - g_{k,t}; \sigma_{k+1}(r) \leftarrow \sigma_k(t); r \leftarrow r - 1$.
15: end if
16: Optimizer Step: $w_k^{(t+1)} \leftarrow w_k^{(t)} - \alpha \nabla f(w_k^t; x_{\sigma_k(t)})$.
17: end for
18: $w_{k+1} \leftarrow w_k^{(n+1)}$.
19: return $w_k^{(1)}$.

Algorithm 5 Balancing without normalization.

1: Input: current sum $s$, vector $v$.
2: $\epsilon \leftarrow +1$ if $\|s + v\| < \|s - v\|$ else $\epsilon \leftarrow -1$.
3: return $\epsilon$.

would require Equation (3) to be on the order of $O(1)$. We can repeatedly call the balancing-reordering subroutines, but that would again require storing all the stochastic gradients.

To overcome challenge I, we apply a two-step stale gradient estimate: for any $k$, we use the running average of stale gradients to “center” stochastic gradient in epoch $k + 1$ (this centering is itself stale, and does not guarantee the vectors sum to 0, but this is still enough to prove convergence). Then, the online balancing-reordering subroutine will determine the proper order to use in epoch $k + 2$. To address challenge II, we leverage a noisy reordering process: let $\sigma_k$ denote the order to use in epoch $k$, then the next order $\sigma_{k+1}$ will be obtained from $\sigma_k$. The intuition here is that if we train the model with enough epochs, the reordering will in general push the herding bound towards $A$ in Theorem 2 without storing any additional vectors. We present the full GraB algorithm in Algorithm 4.

We proceed to provide a convergence guarantee for Algorithm 4. Since now we are working with balancing rather than herding, in the convergence analysis we replace Assumption 4 with the following one related to balancing.

**Assumption 5. (Balancing Bound.)** There exists a constant $A > 0$ such that in Algorithm 4, if the subroutine Balancing is given input vectors $z_1, \ldots, z_n \in \mathbb{R}^d$ that satisfy $\|z_i\|_2 \leq 1$, then Balancing outputs signs $\epsilon_i \in \{-1, +1\}$ such that $\|\sum_{i=1}^k \epsilon_i z_{\sigma(i)}\|_\infty \leq A$ for all $k \in \{1, \ldots, n\}$.

We defer discussing the complexity of $A$ to the later part of this section, and first present the convergence guarantee for GraB (Algorithm 4) as follows.

**Theorem 3.** In Algorithm 6, under Assumptions 1, 3 and 5, if we set $\alpha$ to be

\[
\alpha = \min \left\{ \sqrt{\frac{f(w_1) - f^*}{32nA^2\zeta^2L_{2,\infty}^2K}}, \frac{1}{nL}, \frac{1}{26(n + A)L_2,\infty}, \frac{1}{260nL_\infty} \right\}
\]

then it converges at the rate

\[
\frac{1}{K} \sum_{k=1}^K \|\nabla f(w_k)\|^2 \leq 11 \sqrt{\frac{H^2\zeta^2L_{2,\infty}F^2}{n^2K^2}} + \zeta^2 K + \frac{65(F(L + L_2,\infty + L_\infty)}{K} + \frac{8FAL_2,\infty}{nK}.
\]
Algorithm 6 Probabilistic Balancing with Logarithm Bound. [38]

1: **Input:** current sum $s$, received vector $z$, hyperparameter $c$.
2: if $|s, z| > c$ or $||s||_{\infty} > c$ then
3:     **Fail**.
4: end if
5: $\epsilon \leftarrow +1$ with probability $\frac{1}{2} - \frac{(s, z)}{2c}$ and $\epsilon \leftarrow -1$ with probability $\frac{1}{2} + \frac{(s, z)}{2c}$.
6: **return** $\epsilon$.

where $F = f(w_1) - f^*$. Furthermore, under the additional PL condition (Assumption 2) with $\alpha$ as

$$\alpha = \min \left\{ \frac{1}{n\mu}, \frac{1}{nL}, \frac{1}{52(n + A)L_{\infty}}, \frac{1}{520nL_{\infty}}, \frac{2}{n\mu K} W_0 \left( \frac{(f(w_1) - f^* + \varsigma^2) n^2 \mu^3 K^2}{256 A^2 L_{\infty}^3 \varsigma^2} \right) \right\},$$

where $W_0$ denotes the Lambert W function, it converges at the rate

$$f(w_K) - f^* \leq O \left( \frac{A^2 L_{\infty}^3 \varsigma^2}{\mu^3 n^2 K^2} \right).$$

Comparing Theorem 1 and 3, we can observe GraB (Algorithm 4) achieves essentially the same fast asymptotic convergence rate as SGD with herding (Algorithm 2).

**Asymptotic complexity for $A$ and $H$.** So far we have covered the main design of GraB. A remaining yet unanswered question is how do $A$ in Assumption 4 and $H$ in Assumption 5 relate to other factors exactly, and what are the concrete algorithms to achieve small $A$ and $H$. Some recent progress in the theory community shows that there exists an online algorithm (Algorithm 6) that guarantees with high probability that $A$ is on the order of $O(1)$ as shown in the following theorem.

**Theorem 4.** (Alweiss et al. [38]) In Algorithm 6, if all the vectors have $||z_i|| \leq 1$ and the parameter $c$ is set $c = 30 \log(nd)/\delta$, then with probability $1 - \delta$, the maximum partial signed sum is bounded by $\max \epsilon_i \sum_{j=1}^i \epsilon_j z_j ||_{\infty} \leq c = O(\log(nd)) = O(1)$.

This theorem gives us a high-probability bound on $A$ that can easily be converted to a high-probability statement that the result of Theorem 3 holds with $A$ replaced by $O(1)$. Theorem 4 also yields a good offline herding algorithm and a bound on $H$: recall from Theorem 2 that given any herding bound $H$ on an ordering, running Algorithm 3 once allows us to obtain a new herding bound $(A + H)/2$. Given Theorem 4 that a $O(1)$ bound is guaranteed on $A$, then if we repeatedly run Algorithm 3 as described in Section 4, we can eventually push the herding bound $H$ towards $A$, which yields $H = O(1)$. This can be done with probability 1 offline by simply restarting on failure; however, as stated in Section 3, this still requires storing all the vectors.

**On the normalization of vectors.** Note that a typical balancing subroutine (e.g. Algorithm 6) requires normalized vectors $||z_i|| \leq 1$. This, in practice, implies we will need to estimate a large enough constant to normalize gradients when we run GraB. To alleviate this issue, we can instead adopt a balancing subroutine that is invariant to the normalizer, for instance, Algorithm 5. As shown in the next section, this naïve balancing algorithm suffices for non-trivial improvements.

6 Experiment

In this section, we evaluate GraB and other baseline algorithms on multiple machine learning applications. We compare GraB (Algorithm 4) with the following baselines: Random Reshuffling (RR), Shuffle Once (SO) [1], FlipFlop [4] and Greedy Ordering [5]. All the experiments run on an instance configured with a 4-core Intel(R) Xeon(R) 2.50GHz CPU, 32GB memory and an NVIDIA GeForce RTX 2080 Ti GPU. Although we used Algorithm 6 to show a $O(1)$ bound can be obtained for herding, in our experiments, we adopt the simple balancing algorithm of Algorithm 5, which performs well in practice. More results with other balancing algorithms can be found in the Appendix.

---

[4] For FlipFlop, we follow the exact implementation code from [https://github.com/shashankrajput/flipflop](https://github.com/shashankrajput/flipflop); and for Greedy Ordering, we strictly follow [https://github.com/EugeneLYC/qmc-ordering](https://github.com/EugeneLYC/qmc-ordering).
Model and Dataset. We adopt the following model training tasks for evaluation: (1) training logistic regression on MNIST, (2) training LeNet on CIFAR10 [39], (3) training LSTM on wikitext-2 [40], and (4) finetuning BERT-Tiny [41] on GLUE [42]. Detailed information regarding models, datasets and hyperparameters can be found in Appendix A.

Convergence Speedup. The convergence plots for each algorithms are summarized in Figure 2. We observe that in general, GraB is able to converge faster in terms of both training and validation loss on various tasks. Of all the baseline algorithms, Greedy Ordering is able to achieve comparable convergence speed with respect to epochs to GraB. However, Greedy Ordering consumes much more memory and wall-clock time as shown in the figure. In the BERT training case, we observe the Greedy Ordering runs Out Of Memory (OOM), causing an error on our machine. Based on simple calculations, GraB only requires less than 1% the memory used by Greedy Ordering for all the tasks. It is worth pointing out that throughout the experiments, we do not tune hyperparameters for GraB but let it reuse the hyperparameters from RR (details in the Appendix). This implies GraB can provide in-place improvement without any additional tuning in practice, and a well-tuned GraB could benefit with even larger margin. Shuffle Once (SO) performs worst in all the cases, and FlipFlop is generally the same as RR, which is aligned with the conclusion made in Rajput et al. [4].

Ablation Study: are good permutations fixed? In GraB, the example ordering $\sigma$ is carried and improved over epochs, which is motivated from minimizing the herding bound. This raises another question of whether the found permutation changes at later stage of training, and if so, can we find a fixed permutation that outperforms the RR in practice? Here we conduct an ablation study on two fixed order strategies with GraB. We include the following two variants:
Figure 3: Ablation Study of GraB with fixed orders. 1-step GraB refers to the algorithm that uses GraB only in the first epoch to obtain an order, and then use it as a fixed order for the rest of the training. Retrain from GraB refers to the algorithm that uses a fixed order obtained at the end of the epoch from a GraB full run (the latter one is not for practice, only for understanding GraB with fixed orders). The main takeaway here is that for convex problems, we can find a fixed example order that outperforms both RR and SO.

- **1-step GraB**: it only runs GraB for 1 epoch to obtain an ordering, and then use it as a fixed order for the rest of the training.
- **Retrain from GraB**: First launch a full run of GraB, and then use the ordering in the final epoch as a fixed order in a new run.

We test these new variants on two tasks: LeNet on CIFAR10 (non-convex) and logistic regression on MNIST (convex). The training loss and validation accuracy curves can be found in Figure 3. We observe that on both tasks, the 1-step GraB does not work well, which is aligned with our theory and motivation in Section 5 Challenge II. On the other hand, Retrain from GraB achieves the comparable performance than original GraB on the convex task, but not on the non-convex one. This is mainly because a good ordering is going to depend on the local optimum the algorithm is approaching and for a convex problem, there is only one such optimum. It again verifies that GraB finds better orderings compared to baseline Random Reshuffling and Shuffle Once.

### On the granularity of example ordering.

Throughout this paper, we have discussed how to find good data permutations via per-example gradients. In practice, however, per-example gradients are usually not easy to obtain since data is usually loaded in batches, and many ML libraries (e.g. PyTorch) directly accumulate gradients over the batch. A direct workaround is to fix the data within batches and reorder the batches (treating them as coarse-grained examples). This, however, would compromise the benefits of ordering since the total number of examples $n$ is reduced by a factor of batch size, while the statistical improvement of herding (GraB) is in the order of $O(n^{-1/3})$. To alleviate this, we provide two alternatives: (1) Use ML frameworks that support quick per-example gradients computation (e.g. JAX); (2) Leverage gradient accumulation steps as used in large language model training, i.e., use smaller batch sizes in the code but perform optimizer.step once every few steps so that we can obtain the finer-grained gradients on-the-fly while still optimizing the models with the desired batch size. This is the default method we used throughout the experiments, we will include more details in the appendix.

### 7 Conclusion

In this paper, we formulate a herding framework for data-ordering in SGD. We prove SGD with herding finds better data orderings, and converges faster than random reshuffling on smooth non-convex and PL objectives. We propose an online gradient balancing algorithm named GraB that finds a better ordering with little compute or memory overhead. We substantiate our theory and the usefulness of GraB on multiple machine learning applications.

### Acknowledgement

This work is supported by NSF-2046760. Yucheng is supported by Meta PhD Fellowship. The authors would like to thank Si Yi Meng and anonymous reviewers from NeurIPS 2022 for providing valuable feedbacks on this paper.

---

References


Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default [TODO] to [Yes], [No], or [N/A]. You are strongly encouraged to include a justification to your answer, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes]
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The details are given in text.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

4. If you used existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you’re using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]
A Experimental Details

Setup. We first provide the details to models and datasets used in the experiments:

1. **Logistic Regression on MNIST:** The MNIST consists of handwritten digits 0-9, it has a training set of 60,000 examples, and a test set of 10,000 examples. It contains $28 \times 28 = 784$ features and 10 classes. The model is of dimension $d = 784 \times 10 + 10 = 7850$ (with bias).

2. **LeNet on CIFAR10.** The CIFAR10 dataset consists of 60000 32 $\times$ 32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. LeNet is a classic convolutional neural network proposed by [43].

3. **LSTM on WikiText-2.** In this task we train a 2-layer LSTM on the wikitext-2 dataset, which contains 2 million words. We set the embedding size to be 32, number of hidden unit to be 32 and number of head to be 2. We adopt the learning rate schedule from PyTorch example repo. We set the sequence length (bptt) to be 35.

4. **BERT on GLUE.** The General Language Understanding Evaluation (GLUE) benchmark is a collection of resources for training, evaluating, and analyzing natural language understanding systems. GLUE consists of 11 different tasks. In the main paper, we evaluate on the SST-2 and QNLI. For this task, we adopt the BERT-Tiny model released by Google Research. BERT-Tiny contains 768 hidden layers. For each task, we set maximum sequence length to be 32 and enable padding.

Hyperparameters. We tune the hyperparameters for all the algorithms except GraB within a given range. Then we reuse the hyperparameters for RR in GraB. This implies GraB can potentially provide in-place benefit without additional tuning. We use momentum SGD (with its default value 0.9) for all the tasks. The hyperparameters (ranges) for each task are as follows:

1. **MNIST:** LR $\in \{0.1, 0.01, 0.001, 0.0001\}$; BSZ=64; GCC=32; WD=0.0001.
2. **CIFAR10,** LR $\in \{0.1, 0.01, 0.001, 0.0001\}$; BSZ=16; GCC=2; WD=0.0001.
3. **WikiText-2.** We set momentum to be 0.9 and let the learning rate follow ReduceLROnPlateau from Pytorch with variable: {mode='min', factor=0.1, patience=5, threshold=5}. The initial learning rate is set to be 5.
4. **GLUE.** On the SST-2 we adopt BSZ=1; GCC=1; WD=0.01; LR $\in \{0.005, 0.001, 0.0005, 0.0001\}$. On the QNLI task we adopt BSZ=2; GCC=2; WD=0.01; LR $\in \{0.005, 0.001, 0.0005, 0.0001\}$.

LR stands for learning rate, BSZ stands for batch size, GCC stands for the gradient accumulation steps and WD stands for the weight decay.

Gradient Accumulation. As illustrated in the paper, one workaround to obtain fine-grained gradients (subgradients over one example, or a group of examples of size smaller than batch size) is to leverage the gradient accumulation step, especially in frameworks that do not support per-example gradient computation like PyTorch.

```python
for epoch in range(num_epochs):
    epoch_step = 0
    for batch in grab_ordered_batches:
        epoch_step += 1
        optimizer.zero_grad()
        grad = backward(batch)
        ... # GraB related steps
        grad_buffer.add_(grad / grad_accumulation_step)
        if epoch_step % grad_accumulation_step == 0:
            optimizer.apply_gradient(grad_buffer)
            grad_buffer.zero_()
```

Listing 1: A simple workaround to obtain fine-grained gradients in some ML frameworks such as PyTorch.

The effect of different balancing algorithms. We extend the experiment from Figure 1 and run Algorithm 5 and Algorithm 6 for different epochs. Figure 4 summarizes the results: although the

---

6https://github.com/pytorch/examples/tree/main/word_language_model
7https://github.com/google-research/bert.
Azuma-Hoeffding Inequality we know

In Algorithm 2, under Assumptions 1, 3 and 4, if we set

Theorem 1.

B Theoretical Analysis

In this section, we provide the detailed proof for all the theorems in the main paper. Throughout the proof, for simplicity, we denote \( \Delta_k = \max_{m=2,\ldots,n+1} \| w_{k+1}^{(m)} - w_k \|_\infty \) for all \( k \geq 1 \). Additionally, we define the maximum backward deviation within an epoch to be

For any \( k \), we let \( \sigma_k^{-1}(i) = \{ t = 1, \ldots, n \mid \sigma_k(t) = i \} \) to be the step when \( i \)-th example is visited. Unless otherwise specified, we use \( \| \cdot \| \) to denote the \( \ell_2 \)-norm.

B.1 Details to the greedy statement from [25]

Proof. We first construct a group of 2-d vectors. Without the loss of generality, we let \( n \) divides 2. And let \( n/2 \) vectors be \( [1, 1]^\top \) and the other \( n/2 \) vectors be \( [4, -2]^\top \). Algorithm 1 will always select \([1, 1]^\top \) in the first \( n/2 \) steps based on the current sum being \([m, m]^\top \), \( \forall m \leq n/2 \). We show this with induction. Note that when \( t = 1, [1, 1]^\top \) is selected. Then suppose in the \( k \)-th selection the current sum is \([k, k]^\top \), the algorithm will again select \([1, 1]^\top \) since \( 2(k + 1)^2 < (k + 4)^2 + (k - 2)^2 \). This makes the herding objective \( \Omega(n) \) with Algorithm 1.

On the other hand, consider using a random permutation. Let random variable \( X_t \) denote the value of herding objective at \( t \)-th selection. Then we know that \( X_t \) is a 2-d martingale. So that from Azuma-Hoeffding Inequality we know \( \| X_k \| \leq O(\sqrt{n}) \) for all \( k \).

B.2 Proof to Theorem 1

Theorem 1. In Algorithm 2, under Assumptions 1, 3 and 4, if we set \( \alpha \) to be

\[
\alpha = \min \left\{ \frac{f(w_1) - f^*}{24nH^2 + 2L_2 \infty K}, \frac{1}{8n(L + L_2 \infty)^1}, \frac{1}{32nL \infty}, \frac{1}{16HL_2 \infty}\right\}
\]

then it converges at the rate

\[
\frac{1}{K} \sum_{k=1}^K \| \nabla f(w_k) \|^2 \leq 36 \left( \frac{H^2 L_2 \infty F^2}{n^2 K^2} + \frac{\varsigma^2}{K} + \frac{32F(L + L_2 \infty + L_\infty)}{K} + \frac{64FH L_2 \infty}{nK} \right),
\]

where \( F = f(w_1) - f^* \). Furthermore, under the additional PL condition (Assumption 2), with \( \alpha \) as

\[
\alpha = \min \left\{ \frac{1}{n\mu}, \frac{1}{48HL_2 \infty}, \frac{1}{96n(L + L_2 \infty + L_\infty)}, \frac{2}{n\mu K} W_0 \left( \frac{(f(w_1) - f^*)^2 n^2 \mu^3 K^2}{192H^2 L_2 \infty \varsigma^2} \right) \right\},
\]
where $W_0$ denotes the Lambert W function, Algorithm 2 converges at the rate

$$f(w_K) - f^* \leq \tilde{O}\left(\frac{H^2 L_{2,\infty}^2 \varsigma^2}{\mu^3 n^2 K^2}\right).$$

Proof. From Lemma 1, we get

$$\frac{1}{K} \sum_{k=1}^{K} \|\nabla f(w_k)\|^2 \leq \frac{2(f(w_1) - f^*)}{\alpha n K} + \frac{L_{2,\infty}^2}{K} \sum_{k=1}^{K} \max_{t} \|w_k - w_k^{(t)}\|_\infty^2.$$ 

On the other hand, from Lemma 3, we obtain

$$\sum_{k=1}^{K} \Delta_k^2 \leq 16\alpha^2 n^2 \varsigma^2 + 48\alpha^2 H^2 \varsigma^2 K + 48\alpha^2 n^2 \sum_{k=1}^{K} \|\nabla f(w_k)\|_\infty^2.$$ 

Combining them together gives us,

$$\frac{1}{K} \sum_{k=1}^{K} \|\nabla f(w_k)\|^2 \leq \frac{2(f(w_1) - f^*)}{\alpha n K} + \frac{L_{2,\infty}^2}{K} \left(16\alpha^2 n^2 \varsigma^2 + 48\alpha^2 H^2 \varsigma^2 K + 48\alpha^2 n^2 \sum_{k=1}^{K} \|\nabla f(w_k)\|_\infty^2\right)$$

$$\leq \frac{2(f(w_1) - f^*)}{\alpha n K} + \frac{16\alpha^2 n^2 \varsigma^2 L_{2,\infty}^2}{K} + 48\alpha^2 H^2 \varsigma^2 L_{2,\infty}.$$

Note that for any $x \in \mathbb{R}^d$, $\|x\|_\infty \leq \|x\|_2$. And so the last term can be bounded by its $\ell_2$-norm. Moving it to the left side of the inequality gives us

$$\frac{1}{K} \sum_{k=1}^{K} \|\nabla f(w_k)\|^2 \leq \frac{2(f(w_1) - f^*)}{\alpha n K} + \frac{16\alpha^2 n^2 \varsigma^2 L_{2,\infty}^2}{K} + 48\alpha^2 H^2 \varsigma^2 L_{2,\infty}.$$

Finally, we choose $\alpha$ so that

$$\alpha = \min\left\{\frac{3(f(w_1) - f^*)}{24n H^2 \varsigma^2 L_{2,\infty}^2}, \frac{1}{8n(L + L_{2,\infty})}, \frac{1}{32n L_{\infty}}, \frac{1}{16 H L_{2,\infty}}\right\}.$$ 

$$\frac{1}{K} \sum_{k=1}^{K} \|\nabla f(w_k)\|^2 \leq \frac{4(f(w_1) - f^*)}{\alpha n K} + \frac{32\alpha^2 n^2 \varsigma^2 L_{2,\infty}^2}{K} + 96\alpha^2 H^2 \varsigma^2 L_{2,\infty}^2$$

$$\leq 36 \left(\frac{H^2 \varsigma^2 L_{2,\infty}^2}{n^2 K^2} (f(w_1) - f^*)^2 + \varsigma^2 \right)$$

$$+ \frac{32(f(w_1) - f^*)(L + L_{2,\infty} + L_{\infty})}{K} + \frac{64(f(w_1) - f^*) H L_{2,\infty}}{n K}.$$ 

Denoting $F = f(w_1) - f^*$, we finally obtain This gives

$$\frac{1}{K} \sum_{k=1}^{K} \|\nabla f(w_k)\|^2 \leq 36 \left(\frac{H^2 \varsigma^2 L_{2,\infty}^2 F^2}{n^2 K^2} + \varsigma^2 \right)$$

$$+ \frac{32F(L + L_{2,\infty} + L_{\infty})}{K} + \frac{64F H L_{2,\infty}}{n K}.$$ 

This gives us the bound for general smooth-convex case. We proceed to prove the PL case:

From Lemma 3 we have the following relation:

$$\Delta_k \leq 2\alpha H \varsigma + (8\alpha n L_{\infty} + 4\alpha H L_{2,\infty}) \Delta_{k-1} + 2\alpha n \|\nabla f(w_k)\|_\infty, \forall k \geq 2.$$ 

Square on both sides

$$\Delta_k^2 \leq 3\alpha^2 (4HL_{2,\infty} + 8n L_{\infty})^2 \Delta_{k-1}^2 + 12\alpha^2 H^2 \varsigma^2 + 12\alpha^2 n^2 \|\nabla f(w_k)\|^2.$$ 

17
Summing from $k = 1$ to $K - 1$,
\[
\sum_{k=1}^{K-1} \rho^{K-k} \Delta_k^2 \\
= \rho^{K-2} \Delta_1^2 + \sum_{k=2}^{K-1} \rho^{K-k} \Delta_k^2 \\
\leq \rho^{K-2} \Delta_1^2 + \sum_{k=2}^{K-1} \rho^{K-k} \left( 3\alpha^2 (4HL_{2,\infty} + 8nL_{\infty})^2 \Delta_k^2 - \frac{12\alpha^2 H^2 \zeta^2}{1-\rho} \right) + 12\alpha^2 n^2 \sum_{i=0}^{\infty} \rho^i \\
\leq \rho^{K-2} \Delta_1^2 + 3\alpha^2 (4HL_{2,\infty} + 8nL_{\infty})^2 \sum_{k=2}^{K-1} \rho^{K-k} \Delta_k^2 - \frac{12\alpha^2 H^2 \zeta^2}{1-\rho} + 12\alpha^2 n^2 \sum_{k=2}^{K-1} \rho^{K-k} \|
abla f(w_k)\|^2.
\]

Recall from Lemma 3 that
\[
\Delta_1^2 \leq 8\alpha^2 n^2 \|
abla f(w_1)\|^2 + 8\alpha^2 n^2 \zeta^2,
\]
this gives us,
\[
\sum_{k=1}^{K-1} \rho^{K-k} \Delta_k^2 \leq \rho^{K-2} (8\alpha^2 n^2 \|
abla f(w_1)\|^2 + 8\alpha^2 n^2 \zeta^2) + 3\alpha^2 \rho^{-1} (4HL_{2,\infty} + 8nL_{\infty})^2 \sum_{k=1}^{K-1} \rho^{K-k} \Delta_k^2 \\
+ \frac{12\alpha^2 H^2 \zeta^2}{1-\rho} + 12\alpha^2 n^2 \sum_{k=2}^{K-1} \rho^{K-k} \|
abla f(w_k)\|^2.
\]

Using the fact that
\[
\alpha = \min \left\{ \frac{1}{n\mu}, \frac{1}{48HL_{2,\infty}}, \frac{1}{96n(L + L_{2,\infty} + L_{\infty})} \right\}
\]
Solve it, we obtain
\[
\sum_{k=1}^{K-1} \rho^{K-k} \Delta_k^2 \leq 64\alpha \rho^k \alpha^2 n^2 \zeta^2 + 24\alpha^2 H^2 \zeta^2 \frac{1}{1-\rho} + 24\alpha^2 n^2 \sum_{k=1}^{K-1} \rho^{K-k} \|
abla f(w_k)\|^2.
\]

First from Lemma 2 we get
\[
f(w_k) - f^* \leq \rho^K (f(w_1) - f^*) + \frac{\alpha nL_{2,\infty}}{2} \sum_{k=1}^{K-1} \rho^{K-k} \Delta_k^2 - \frac{\alpha n}{4} \sum_{k=1}^{K-1} \rho^{K-k} \|
abla f(w_k)\|^2 \\
\leq \rho^K (f(w_1) - f^*) + 32\rho^k \alpha^3 n^3 L_{2,\infty} \zeta^2 + \frac{24\alpha^2 H^2 L_{2,\infty} \zeta^2}{\mu},
\]
where in the last step, we apply the learning rate bound that
\[
\alpha \leq \frac{1}{96n(L + L_{2,\infty} + L_{\infty})} \leq \frac{1}{96nL_{2,\infty}}.
\]
The RHS of the inequality can be further bounded by
\[
(1 - \alpha n\mu/2)^K (f(w_1) - f^*) + 24\alpha^2 H^2 L_{2,\infty} \zeta^2
\]

18
Theorem 3. In Algorithm 6, under Assumptions 1, 3 and 5, if we set $\alpha$ to be

$$\alpha = \min \left\{ \frac{2}{n^{3/2} K}, \frac{1}{32 n A^2 \zeta^2 L_{2,\infty}^2 K}, \frac{1}{26 (n + A) L_{2,\infty}}, \frac{1}{260 n L_{\infty}} \right\},$$

then it converges at the rate

$$1 \leq \frac{1}{K} \sum_{k=1}^{K} \|\nabla f(w_k)\|^2 \leq 1 + \frac{L_{2,\infty}^2}{\alpha n^2 K^2} \left( \frac{65 F(L + L_{2,\infty} + L_{\infty})}{K} + \frac{8 F A L_{2,\infty}}{n K} \right),$$

where $F = f(w_1) - f^*$. Furthermore, under the additional PL condition (Assumption 2) with $\alpha$ as

$$\alpha = \min \left\{ \frac{1}{n^{3/2} K}, \frac{1}{26 n L_{\infty} L_{2,\infty}}, \frac{1}{520 n L_{\infty}}, \frac{2}{n^{3/2} K W_0} \right\},$$

where $W_0$ denotes the Lambert W function, it converges at the rate

$$f(w_K) - f^* \leq \tilde{O} \left( \frac{A^2 L_{2,\infty}^2 \zeta^2}{\mu^3 n^2 K^2} \right).$$

Proof. From Lemma 1, we get

$$\frac{1}{K} \sum_{k=1}^{K} \|\nabla f(w_k)\|^2 \leq \frac{2}{\alpha n K} \frac{L_{2,\infty}^2}{K} \sum_{k=1}^{K} \max_{t} \|w_k - w^{(t)}_k\|^2.$$

On the other hand, from Lemma 3, we obtain

$$\sum_{k=1}^{K} \Delta_k^2 \leq 120 \alpha^2 n^2 \zeta^2 + 64 \alpha^2 A^2 \zeta^2 K + 48 \alpha^2 n^2 \sum_{k=1}^{K} \|\nabla f(w_k)\|^2.$$

Combining them together gives us,

$$\frac{1}{K} \sum_{k=1}^{K} \|\nabla f(w_k)\|^2 \leq \frac{2}{\alpha n K} \frac{L_{2,\infty}^2}{K} \left( 120 \alpha^2 n^2 \zeta^2 + 64 \alpha^2 A^2 \zeta^2 K + 48 \alpha^2 n^2 \sum_{k=1}^{K} \|\nabla f(w_k)\|^2 \right) \leq \frac{2}{\alpha n K} \frac{L_{2,\infty}^2}{K} \left( 120 \alpha^2 n^2 \zeta^2 L_{2,\infty}^2 + 64 \alpha^2 A^2 \zeta^2 L_{2,\infty}^2 + \frac{48 \alpha^2 n^2 L_{2,\infty}^2}{K} \sum_{k=1}^{K} \|\nabla f(w_k)\|^2 \right).$$
Given
\[ \alpha = \min \left\{ \sqrt[3]{\frac{f(w_1) - f^*}{32nA^2\xi^2L_{2,\infty}^2K}}, \frac{1}{n\bar{L}}, \frac{1}{26(n + A)L_{2,\infty}}, \frac{1}{260nL_\infty} \right\}, \]
we get
\[
\frac{1}{K} \sum_{k=1}^{K} \|\nabla f(w_k)\|^2 \leq \frac{4(f(w_1) - f^*)}{\alpha nK} + \frac{240\alpha^2 n^2 \xi^2 L_{2,\infty}^2}{nK^2} + 128\alpha^2 A^2 \xi^2 L_{2,\infty}^2
\]
\[
\leq 11 \left( \frac{H^2 \xi^2 L_{2,\infty}^2 (f(w_1) - f^*)^2}{n^2 K^2} + \frac{\xi^2}{K} \right.
\]
\[
+ 65(f(w_1) - f^*)(L + L_{2,\infty} + L_\infty) + \frac{8(f(w_1) - f^*)AL_{2,\infty}}{nK}.
\]
This gives us the bound for general smooth-convex case. We proceed to prove the PL case:
From Lemma 4 we obtain the following bound:
\[
\Delta_k^2 \leq 16\alpha^2 (8 \cdot 2^{-k} n + A) \xi^2
\]
\[
+ 16\alpha^2 [(8 \cdot 2^{-k} n + A)L_{2,\infty} + 10nL_\infty]^2 \Delta_{k-1}^2
\]
\[
+ 64\alpha^2 [(8 \cdot 2^{-k} n + A)L_{2,\infty} + 5nL_\infty]^2 \Delta_{k-2}^2 + 16\alpha^2 n^2 \|\nabla f(w_{k-1})\|_\infty^2, \forall k \geq 3
\]
and,
\[
\Delta_k^2 \leq 8\alpha^2 n^2 \|\nabla f(w_k)\|_\infty^2 + 8\alpha^2 n^2 \xi^2, k = 1, 2.
\]
Summing over \(k = 1\) to \(K - 1\), we obtain
\[
\sum_{k=1}^{K-1} \rho^{K-1-k} \Delta_k^2
\]
\[
= 2 \sum_{k=1}^{K-1} \rho^{K-1-k} \Delta_k^2 + \sum_{k=3}^{K-1} \rho^{K-1-k} \Delta_k^2
\]
\[
= 8\alpha^2 n^2 \sum_{k=1}^{K-1} \rho^{K-1-k} \|\nabla f(w_k)\|_\infty^2 + 8(\rho^{K-2} + \rho^{K-3}) \alpha^2 n^2 \xi^2 + 16\alpha^2 \xi^2 \left( 128n^2 \rho^{K-1} \sum_{k=3}^{K-1} 2^{k-4} + \frac{2A^2}{1 - \rho} \right)
\]
\[
+ 16\alpha^2 [(n + A)L_{2,\infty} + 10nL_\infty]^2 \sum_{k=3}^{K-1} \rho^{K-1-k} \Delta_{k-1}^2
\]
\[
+ 64\alpha^2 [(n + A)L_{2,\infty} + 5nL_\infty]^2 \sum_{k=3}^{K-1} \rho^{K-1-k} \Delta_{k-2}^2
\]
\[
+ 16\alpha^2 n^2 \sum_{k=3}^{K-1} \rho^{K-1-k} \|\nabla f(w_{k-1})\|_\infty^2.
\]
Notice that
\[
\sum_{k=3}^{K} \rho^{K-1-k} 4^{-k} = \rho^{K-1} \sum_{k=3}^{K} \rho^{k-4} 4^{-k} \leq \rho^{K-1} \sum_{k=2}^{\infty} 2^{-k} = \rho^{K-1} 2^{-1}.
\]
Putting it back we obtain
\[
\sum_{k=1}^{K-1} \rho^{K-1-k} \Delta_k^2
\]
\[
\leq 96\alpha^2 n^2 \xi^2 + 16\alpha^2 \xi^2 \left( 64\rho^K n^2 + \frac{2A^2}{1 - \rho} \right)
\]
20
+ 16\alpha^2 \rho^{-1} [(n + A)L_{2,\infty} + 10\mu L_{\infty}]^2 \sum_{k=3}^{K-1} \rho^{K-1-k} \Delta^2_{k-1}
+ 64\alpha^2 \rho^{-2} [(n + A)L_{2,\infty} + 5\mu L_{\infty}]^2 \sum_{k=3}^{K-1} \rho^{K-1-(k-2)} \Delta^2_{k-2}
+ 32\alpha^2 n^2 \sum_{k=1}^{K-1} \rho^{K-1-k} \|\nabla f(w_k)\|^2_\infty
\leq 1120\alpha^2 n^2 \rho^K + \frac{32\alpha^2 A^2 \rho^2}{1 - \rho} + 32\alpha^2 n^2 \sum_{k=1}^{K-1} \rho^{K-1-k} \|\nabla f(w_k)\|^2_\infty
+ 16\alpha^2 \rho^{-1} [(n + A)L_{2,\infty} + 10\mu L_{\infty}]^2 \sum_{k=1}^{K-1} \rho^{K-1-k} \Delta^2_k
+ 64\alpha^2 \rho^{-2} [(n + A)L_{2,\infty} + 5\mu L_{\infty}]^2 \sum_{k=1}^{K-1} \rho^{K-1-k} \Delta^2_k.

Given \(\frac{1}{\rho} = \frac{1}{1 - \frac{\mu}{\alpha}} \leq 2\), and
\[
\alpha \leq \min \left\{ \frac{1}{52(n + A)L_{2,\infty}}, \frac{1}{520n L_{\infty}} \right\},
\]
we can simply the bound to be
\[
\sum_{k=1}^{K-1} \rho^{K-1-k} \Delta^2_k \leq 2240\alpha^2 n^2 \rho^K + \frac{64\alpha^2 A^2 \rho^2}{1 - \rho} + 64\alpha^2 n^2 \sum_{k=1}^{K-1} \rho^{K-1-k} \|\nabla f(w_k)\|^2_\infty.
\]
First from Lemma 2 we get
\[
f(w_K) - f^* \leq \rho^K (f(w_1) - f^*) + \frac{\alpha n L_{2,\infty}^2}{2} \sum_{k=1}^{K-1} \rho^{K-1-k} \Delta^2_k - \frac{\alpha n}{4} \sum_{k=1}^{K-1} \rho^{K-1-k} \|\nabla f(w_k)\|^2
\leq \rho^K (f(w_1) - f^*) + 1120\rho^K \alpha^3 n^3 L_{2,\infty}^2 \rho^2 + \frac{64\alpha^2 A^2 L_{2,\infty}^2 \rho^2}{\mu},
\]
where in the last step, we apply the learning rate bound that
\[
\alpha \leq \frac{1}{52n L_{2,\infty}}.
\]
The RHS of the inequality can be further bounded by
\[
(1 - \alpha n \mu/2)^K (f(w_1) - f^* + \rho^2) + \frac{64\alpha^2 A^2 L_{2,\infty}^2 \rho^2}{\mu}
\leq (f(w_1) - f^* + \rho^2) \exp(-\alpha n \mu K/2) + 64\alpha^2 A^2 L_{2,\infty}^2 \rho^2 \mu^{-1}.
\]
Take derivative with respect to \(\alpha\) and set it to zero, we obtain
\[
\alpha = \frac{2}{n \mu K} W_0 \left( \frac{(f(w_1) - f^* + \rho^2)n^2 \mu^3 K^2}{256 A^2 L_{2,\infty}^2 \rho^2} \right),
\]
and
\[
f(w_K) - f^* \leq \frac{320 A^2 L_{2,\infty}^2 \rho^2}{n^2 \mu^3 K^2} \cdot W_0 \left( \frac{(f(w_1) - f^* + \rho^2)n^2 \mu^3 K^2}{256 A^2 L_{2,\infty}^2 \rho^2} \right) \left[ 1 + W_0 \left( \frac{(f(w_1) - f^* + \rho^2)n^2 \mu^3 K^2}{256 A^2 L_{2,\infty}^2 \rho^2} \right) \right]
= O \left( \frac{A^2 L_{2,\infty}^2 \rho^2}{\mu^3 n^2 K^2} \right),
\]
where \(W_0(\cdot)\) denotes the Lambert W function. That completes the proof. \(\square\)
B.4 Technical Lemmas

Lemma 1. In Algorithm 2 and Algorithm 4, if \( \alpha nL < 1 \) holds and Assumption 1 (except PL condition), 3 and 4 hold, then

\[
\frac{1}{K} \sum_{k=1}^{K} \left\| \nabla f(w_k) \right\|^2 \leq \frac{2(f(w_1) - f^*)}{\alpha nK} + \frac{L_{\infty}^2}{2K} \sum_{k=1}^{K} \max_t \left\| w_k - w_k^{(1)} \right\|^2.
\]

Proof. Note that in both algorithms, the update can be written as

\[
w_{k+1} = w_k - \alpha \sum_{t=1}^{n} \nabla f(w_k^{(t)}; x_{\sigma_k(t)}).
\]

By the Taylor Theorem, for all the \( k \),

\[
f(w_{k+1}) \leq f(w_k) - \alpha n \left( \nabla f(w_k), \frac{1}{n} \sum_{t=1}^{n} \nabla f(w_k^{(t)}; x_{\sigma_k(t)}) \right) + \frac{\alpha^2 n L^2}{2} \left\| \frac{1}{n} \sum_{t=1}^{n} \nabla f(w_k^{(t)}; x_{\sigma_k(t)}) \right\|^2
\]

\[
= f(w_k) - \frac{\alpha n}{2} \left\| \nabla f(w_k) \right\|^2 - \frac{\alpha n}{2} \left\| \frac{1}{n} \sum_{t=1}^{n} \nabla f(w_k^{(t)}; x_{\sigma_k(t)}) \right\|^2 + \frac{\alpha^2 n L^2}{2} \left\| \frac{1}{n} \sum_{t=1}^{n} \nabla f(w_k^{(t)}; x_{\sigma_k(t)}) \right\|^2
\]

\[
\leq f(w_k) - \frac{\alpha n}{2} \left\| \nabla f(w_k) \right\|^2 + \frac{\alpha n}{2} \left\| \nabla f(w_k) - \frac{1}{n} \sum_{t=1}^{n} \nabla f(w_k^{(t)}; x_{\sigma_k(t)}) \right\|^2.
\]

In the second step, we apply \(-\langle a, b \rangle = -\frac{1}{2} \left\| a \right\|^2 - \frac{1}{2} \left\| b \right\|^2 + \frac{1}{2} \left\| a - b \right\|^2, \forall a, b\). In the third step, we use the condition that \( \alpha nL < 1 \). Expanding the last term using Assumption 1, we get

\[
\left\| \nabla f(w_k) - \frac{1}{n} \sum_{t=1}^{n} \nabla f(w_k^{(t)}; x_{\sigma_k(t)}) \right\|^2 = \left\| \frac{1}{n} \sum_{t=1}^{n} \nabla f(w_k; x_{\sigma_k(t)}) - \frac{1}{n} \sum_{t=1}^{n} \nabla f(w_k^{(t)}; x_{\sigma_k(t)}) \right\|^2
\]

\[
\leq \frac{1}{n} \sum_{t=1}^{n} \left\| \nabla f(w_k; x_{\sigma_k(t)}) - \nabla f(w_k^{(t)}; x_{\sigma_k(t)}) \right\|^2
\]

\[
\leq \frac{L_{\infty}^2}{n} \sum_{t=1}^{n} \left\| w_k - w_k^{(t)} \right\|^2
\]

\[
\leq L_{\infty}^2 \Delta_k^2.
\]

In the second step we apply the Jensen Inequality. Put it back, we obtain

\[
f(w_{k+1}) \leq f(w_k) - \frac{\alpha n}{2} \left\| \nabla f(w_k) \right\|^2 + \frac{\alpha nL_{\infty}^2 \Delta_k^2}{2}.
\]

Finally, summing from \( k = 1 \) to \( K - 1 \), we obtain

\[
\frac{1}{K} \sum_{k=1}^{K} \left\| \nabla f(w_k) \right\|^2 \leq \frac{2(f(w_1) - f^*)}{\alpha nK} + \frac{L_{\infty}^2}{2K} \sum_{k=1}^{K} \max_t \left\| w_k - w_k^{(1)} \right\|^2.
\]

That completes the proof.

Lemma 2. In Algorithm 2 and Algorithm 4, if \( \alpha nL < 1 \) holds and Assumption 1 (including PL condition), 3 and 4 hold, then

\[
f(w_K) - f^* \leq \rho^K (f(w_1) - f^*) + \frac{\alpha n L_{\infty}^2}{2} \sum_{k=1}^{K-1} \rho^{K-1-k} \Delta_k^2 - \frac{\alpha n}{4} \sum_{k=1}^{K-1} \rho^{K-1-k} \left\| \nabla f(w_k) \right\|^2.
\]
Proof. Since the this lemma is a special case of Lemma 1, we can just borrow the derivation there and get for all the \( k = 1, \cdots, K - 1 \)

\[
f(w_{k+1}) \leq f(w_k) + \frac{\alpha n}{2} L_{2, \infty}^2 \Delta_k^2 - \frac{\alpha n}{2} \|\nabla f(w_k)\|^2
\]

\[
= f(w_k) + \frac{\alpha n}{2} L_{2, \infty}^2 \Delta_k^2 - \frac{\alpha n}{4} \|\nabla f(w_k)\|^2 - \frac{\alpha n}{4} \|\nabla f(w_k)\|^2
\]

\[
\leq f(w_k) + \frac{\alpha n}{2} L_{2, \infty}^2 \Delta_k^2 - \frac{\alpha n}{2}(f(w_k) - f^*) - \frac{\alpha n}{4} \|\nabla f(w_k)\|^2. \quad \text{(PL condition)}
\]

Define \( 1 - \alpha n/2 = \rho \), then

\[
f(w_{k+1}) - f^* \leq \rho (f(w_k) - f^*) + \frac{\alpha n}{2} L_{2, \infty}^2 \Delta_k^2 - \frac{\alpha n}{4} \|\nabla f(w_k)\|^2.
\]

Recursively apply it from \( k = 1 \) to \( K - 1 \), we obtain

\[
f(w_K) - f^* \leq \rho^K (f(w_1) - f^*) + \frac{\alpha n}{2} L_{2, \infty}^2 \sum_{k=1}^{K-1} \rho^{K-k} \Delta_k^2 - \frac{\alpha n}{4} \sum_{k=1}^{K-1} \rho^{K-k} \|\nabla f(w_k)\|^2.
\]

That completes the proof. \( \square \)

**Lemma 3.** In Algorithm 2, if the learning rate \( \alpha \) fulfills

\[
\alpha \leq \min \left\{ \frac{1}{32 n L_{\infty}}, \frac{1}{16 H L_{2, \infty}} \right\},
\]

then the following inequalities hold:

\[
\Delta_k \leq 2 \alpha H \xi + (8 \alpha n L_{\infty} + 4 \alpha H L_{2, \infty}) \Delta_{k-1} + 2 \alpha n \|\nabla f(w_k)\|_{\infty}, \forall k \geq 2
\]

and,

\[
\Delta_k^2 \leq 8 \alpha^2 n^2 \|\nabla f(w_1)\|_{\infty}^2 + 8 \alpha^2 n^2 \xi^2,
\]

and finally,

\[
\sum_{k=1}^{K} \Delta_k^2 \leq 16 \alpha^2 n^2 \xi^2 + 48 \alpha^2 H^2 \xi K + 48 \alpha^2 n^2 \sum_{k=1}^{K} \|\nabla f(w_k)\|_{\infty}^2.
\]

Proof. Without the loss of generality, for all the \( m \in \{2, \cdots, n + 1\} \) and all the \( k \in \{2, \cdots, K\} \),

\[
w_k^{(m)} = w_k - \alpha \sum_{t=1}^{m-1} \nabla f \left( w_k^{(t)}; x_{\sigma_k(t)} \right)
\]

\[
= w_k - \alpha \sum_{t=1}^{m-1} \nabla f \left( w_{k-1}^{(\sigma_k^{-1}(\sigma_k(t)))}; x_{\sigma_k(t)} \right)
\]

\[
= \alpha \sum_{t=1}^{m-1} \left( \nabla f \left( w_k^{(t)}; x_{\sigma_k(t)} \right) - \nabla f \left( w_{k-1}^{(\sigma_k^{-1}(\sigma_k(t)))}; x_{\sigma_k(t)} \right) \right).
\]

Now add and subtract

\[
\alpha \sum_{t=1}^{m-1} \frac{1}{n} \sum_{s=1}^{n} \nabla f \left( w_{k-1}^{(s)}; x_{\sigma_k^{-1}(s)} \right) = \frac{\alpha (m-1)}{n} \sum_{t=1}^{n} \nabla f \left( w_{k-1}^{(t)}; x_{\sigma_k^{-1}(t)} \right),
\]

which gives

\[
w_k^{(m)} = w_k - \alpha \sum_{t=1}^{m-1} \left( \nabla f \left( w_k^{(t)}; x_{\sigma_k(t)} \right) - \frac{1}{n} \sum_{s=1}^{n} \nabla f \left( w_{k-1}^{(t)}; x_{\sigma_k^{-1}(s)} \right) \right)
\]

\[
= \frac{\alpha (m-1)}{n} \sum_{t=1}^{n} \nabla f \left( w_k^{(t)}; x_{\sigma_k^{-1}(t)} \right)
\]
\[- \alpha \sum_{t=1}^{m-1} \left( \nabla f \left( w^{(t)}_k; x_{\sigma_k(t)} \right) - \nabla f \left( w^{(\sigma_{k-1}(\sigma_k(t)))}_{k-1}; x_{\sigma_k(t)} \right) \right).\]

We further add and subtract
\[
\frac{\alpha(m-1)}{n} \sum_{t=1}^{n} \nabla f(w_k; x_{\sigma_k(t)}) = \alpha(m-1) \nabla f(w_k)
\]
to arrive at
\[
w^{(m)}_k = w_k - \alpha \sum_{t=1}^{m-1} \left( \nabla f \left( w^{(\sigma_{k-1}(\sigma_k(t)))}_{k-1}; x_{\sigma_k(t)} \right) - \frac{1}{n} \sum_{s=1}^{n} \nabla f \left( w^{(s)}_{k-1}; x_{\sigma_k(t)} \right) \right)
- \alpha(m-1) \nabla f(w_k) + \frac{\alpha(m-1)}{n} \sum_{t=1}^{n} \left( \nabla f \left( w_k; x_{\sigma_k(t)} \right) - \nabla f \left( w^{(t)}_{k-1}; x_{\sigma_k(t)} \right) \right)
- \alpha \sum_{t=1}^{m-1} \left( \nabla f \left( w^{(t)}_k; x_{\sigma_k(t)} \right) - \nabla f \left( w^{(\sigma_{k-1}(\sigma_k(t)))}_{k-1}; x_{\sigma_k(t)} \right) \right).
\]

We can now re-arrange, take norms on both sides and apply the triangle inequality,
\[\left\| w^{(m)}_k - w_k \right\|_\infty \leq \alpha \left( \sum_{t=1}^{m-1} \left\| \nabla f \left( w^{(\sigma_{k-1}(\sigma_k(t)))}_{k-1}; x_{\sigma_k(t)} \right) - \frac{1}{n} \sum_{s=1}^{n} \nabla f \left( w^{(s)}_{k-1}; x_{\sigma_k(t)} \right) \right\|_\infty \right)
+ \alpha(m-1) \left\| \nabla f(w_k) \right\|_\infty
+ \frac{\alpha(m-1)}{n} \left( \sum_{t=1}^{n} \left\| \nabla f \left( w_k; x_{\sigma_k(t)} \right) - \nabla f \left( w^{(t)}_{k-1}; x_{\sigma_k(t)} \right) \right\|_\infty \right)
+ \alpha \left( \sum_{t=1}^{m-1} \left\| \nabla f \left( w^{(t)}_k; x_{\sigma_k(t)} \right) - \nabla f \left( w^{(\sigma_{k-1}(\sigma_k(t)))}_{k-1}; x_{\sigma_k(t)} \right) \right\|_\infty \right). \quad (4)
\]

There are four different terms on the right hand side, we will apply the Assumption 4 on the first term, and Assumption 1 on the last two terms. First, for the first term,
\[\left\| \nabla f \left( w^{(\sigma_{k-1}(\sigma_k(t)))}_{k-1}; x_{\sigma_k(t)} \right) - \frac{1}{n} \sum_{s=1}^{n} \nabla f \left( w^{(s)}_{k-1}; x_{\sigma_k(t)} \right) \right\|_\infty \leq \left\| \nabla f \left( w^{(\sigma_{k-1}(\sigma_k(t)))}_{k-1}; x_{\sigma_k(t)} \right) - \frac{1}{n} \sum_{s=1}^{n} \nabla f \left( w^{(\sigma_{k-1}(\sigma_k(t)))}_{k-1}; x_{\sigma_k(t)} \right) \right\|_\infty \]
\[+ \frac{1}{n} \sum_{s=1}^{n} \left\| \nabla f \left( w^{(\sigma_{k-1}(\sigma_k(t)))}_{k-1}; x_{\sigma_k(t)} \right) - \frac{1}{n} \sum_{s=1}^{n} \nabla f \left( w^{(s)}_{k-1}; x_{\sigma_k(t)} \right) \right\|_\infty \]
Assume, 1 and 3
\[\leq \xi + \frac{L_{2,\infty}}{n} \sum_{s=1}^{n} \left\| w^{(\sigma_{k-1}(\sigma_k(t)))}_{k-1} - w^{(s)}_{k-1} \right\|_\infty \]
\[\leq \xi + \frac{L_{2,\infty}}{n} \sum_{s=1}^{n} \left( \left\| w_{k-1} - w^{(\sigma_{k-1}(\sigma_k(t)))}_{k-1} \right\|_\infty + \left\| w_{k-1} - w^{(s)}_{k-1} \right\|_\infty \right) \]
\[\leq \xi + 2L_{2,\infty} \Delta_{k-1}
\]
This implies if we denote
\[u_t := \nabla f \left( w^{(\sigma_{k-1}(\sigma_k(t)))}_{k-1}; x_{\sigma_k(t)} \right) - \frac{1}{n} \sum_{s=1}^{n} \nabla f \left( w^{(s)}_{k-1}; x_{\sigma_k(t)} \right) \]
We can now use Assumption 4 to obtain a bound on the prefix sum
\[\left\| \sum_{t=1}^{m-1} u_t \right\|_\infty \leq H,
\]

24
that is,
\[
\left\| \sum_{t=1}^{m-1} \left( \nabla f \left( w_{k-1}^{(\sigma^{-1}_k(t))}; x_{\sigma_1(t)} \right) - \frac{1}{n} \sum_{s=1}^{n} \nabla f \left( w_{k-1}^{(s)}; x_{\sigma_{k-1}(s)} \right) \right) \right\|_\infty \leq H(\varsigma + 2L_{2,\infty}\Delta_{k-1}).
\]

Now we have a bound for the first term in Equation (4), we proceed to bound the last two terms where we apply Assumption 1. We can then rewrite Equation (4) into,
\[
\left\| w_k^{(m)} - w_k \right\|_\infty \leq \alpha H(\varsigma + 2L_{2,\infty}\Delta_{k-1}) + \alpha(m - 1)\|\nabla f(w_k)\|_\infty + \frac{\alpha L_\infty}{n} \sum_{t=1}^{n} \left\| w_k - w_k^{(t)} \right\|_\infty
\]
\[
+ \alpha L_\infty \sum_{t=1}^{m-1} \left\| w_k^{(t)} - w_{k-1}^{(\sigma^{-1}_k(t))} \right\|_\infty.
\]

Furthermore, applying the triangle inequality to the norms in the last two terms, we obtain
\[
\left\| w_k^{(t)} - w_{k-1} \right\|_\infty = \left\| w_k^{(t)} - w_k + w_k - w_{k-1} + w_{k-1} - w_{k-1}^{(n+1)} \right\|_\infty \leq 2\Delta_{k-1}
\]
and similarly,
\[
\left\| w_k^{(t)} - w_{k-1}^{(\sigma^{-1}_k(t))} \right\|_\infty = \left\| w_k^{(t)} - w_k + w_k - w_{k-1} + w_{k-1} - w_{k-1}^{(\sigma^{-1}_k(t))} \right\|_\infty \leq \Delta_k + 2\Delta_{k-1}.
\]

This gives us
\[
\left\| w_k^{(m)} - w_k \right\|_\infty \leq \alpha H(\varsigma + 2L_{2,\infty}\Delta_{k-1}) + \alpha(m - 1)\|\nabla f(w_k)\|_\infty + \frac{\alpha L_\infty}{n} \sum_{t=1}^{n} \left\| w_k - w_k^{(t)} \right\|_\infty
\]
\[
+ \alpha L_\infty \sum_{t=1}^{m-1} \left\| w_k^{(t)} - w_{k-1}^{(\sigma^{-1}_k(t))} \right\|_\infty
\]
\[
\leq \alpha H(\varsigma + 2L_{2,\infty}\Delta_{k-1}) + \alpha(m - 1)\|\nabla f(w_k)\|_\infty + \frac{\alpha L_\infty}{n} \sum_{t=1}^{n} \left\| w_k - w_k^{(t)} \right\|_\infty
\]
\[
+ \alpha L_\infty \sum_{t=1}^{m-1} \left\| w_k^{(t)} - w_{k-1}^{(\sigma^{-1}_k(t))} \right\|_\infty
\]
\[
\leq \alpha H(\varsigma + 2L_{2,\infty}\Delta_{k-1}) + \alpha(m - 1)L_\infty\Delta_1 + \alpha(m - 1)\varsigma.
\]
\[
\leq \alpha(n-1)\|\nabla f(w_1)\|_\infty + \alpha L_\infty \Delta_1 + \alpha n\varsigma.
\]

Note that Equation (5) only holds with \( k \in \{2, \ldots, K\} \) and \( m \in \{2, \ldots, n + 1\} \). We now discuss the boundary cases. Note that the bound of Equation (5) trivially holds with \( m = 1 \) for any \( k \) since the left hand side becomes zero. On the other hand, when \( k = 1 \), we have,
\[
w_1^{(m)} \equiv w_1 - \alpha \sum_{t=1}^{m-1} \nabla f \left( w_1^{(t)}; x_{\sigma_1(t)} \right)
\]
\[
= w_1 - \alpha \sum_{t=1}^{m-1} \frac{1}{n} \sum_{s=1}^{n} \nabla f \left( w_1; x_{\sigma_1(s)} \right) + \alpha \sum_{t=1}^{m-1} \nabla f \left( w_1^{(t)}; x_{\sigma_1(t)} \right) - \alpha \sum_{t=1}^{m-1} \nabla f \left( w_1; x_{\sigma_1(t)} \right)
\]
\[
+ \alpha \sum_{t=1}^{m-1} \nabla f \left( w_1; x_{\sigma_1(t)} \right) - \alpha \sum_{t=1}^{m-1} \frac{1}{n} \sum_{s=1}^{n} \nabla f \left( w_1; x_{\sigma_1(s)} \right),
\]
take norms and apply the triangle inequality, we obtain
\[
\left\| w_1^{(m)} - w_1 \right\|_\infty \leq \alpha \left\| \sum_{t=1}^{m-1} \frac{1}{n} \sum_{s=1}^{n} \nabla f \left( w_1; x_{\sigma_1(s)} \right) \right\|_\infty + \alpha \left\| \sum_{t=1}^{m-1} \left( \nabla f \left( w_1^{(t)}; x_{\sigma_1(t)} \right) - \nabla f \left( w_1; x_{\sigma_1(t)} \right) \right) \right\|_\infty
\]
\[
+ \alpha \left\| \sum_{t=1}^{m-1} \left( \nabla f \left( w_1; x_{\sigma_1(t)} \right) - \frac{1}{n} \sum_{s=1}^{n} \nabla f \left( w_1; x_{\sigma_1(s)} \right) \right) \right\|_\infty
\]
\[
\leq \alpha(m - 1)\|\nabla f(w_1)\|_\infty + \alpha(m - 1)L_\infty\Delta_1 + \alpha(m - 1)\varsigma.
\]
\[
\leq \alpha n\|\nabla f(w_1)\|_\infty + \alpha L_\infty \Delta_1 + \alpha n\varsigma.
\]

Now that we have the bounds for \( \Delta_k \), we next will sum them up. Taking a max over \( m \) on both side in Equation (5), this implies for all the \( k \geq 2 \),
\[
\Delta_k \leq \alpha H(\varsigma + 2L_{2,\infty}\Delta_{k-1}) + \alpha L_\infty n(4\Delta_{k-1} + \Delta_k) + \alpha n\|\nabla f(w_k)\|_\infty
\]
as \( m - 1 \leq n \). Considering the fact that \( \alpha L_\infty n < 1/2 \), we get
\[
\Delta_k \leq 2\alpha H\varsigma + (8\alpha n L_\infty + 4\alpha HL_{2,\infty})\Delta_{k-1} + 2\alpha n\|\nabla f(w_k)\|_\infty.
\]
We can apply the similar trick to Equation (6) and get

\[ \Delta_k \leq (8\alpha n L_\infty + 4\alpha H L_{2,\infty})^{k-1} \Delta_1 + \sum_{i=1}^\infty (8\alpha n L_\infty + 4\alpha H L_{2,\infty})^i (2\alpha H \zeta + 2\alpha n \|\nabla f(w_k)\|_\infty). \]

Applying the learning rate conditions that 32\alpha n L_\infty \leq 1 and 16\alpha H L_{2,\infty} \leq 1, we obtain

\[ \Delta_k \leq \left( \frac{1}{2} \right)^{k-1} \Delta_1 + 4\alpha H \zeta + 4\alpha n \|\nabla f(w_k)\|_\infty. \]

Square on both sides,

\[ \Delta_k^2 \leq \left( \frac{1}{4} \right)^{k-1} \Delta_1^2 + 8\alpha^2 H^2 \zeta^2 + 48\alpha^2 n^2 \|\nabla f(w_k)\|_\infty^2. \]

We can apply the similar trick to Equation (6) and get

\[ \Delta_k^2 \leq 8\alpha^2 n^2 \|\nabla f(w_1)\|_\infty^2 + 8\alpha^2 n^2 \zeta^2. \]

This completes the proof of the second inequality in the lemma. Summing from \( k = 1 \) to \( K \), we will get

\[
\sum_{k=1}^K \Delta_k^2 = \Delta_0^2 + \sum_{k=2}^K \Delta_k^2 \\
= \Delta_1^2 + 3\Delta_1^2 \sum_{k=2}^K \left( \frac{1}{4} \right)^{k-1} + 48\alpha^2 H^2 \zeta^2 (K - 1) + 48\alpha^2 n^2 \sum_{k=2}^K \|\nabla f(w_k)\|_\infty^2 \\\n\leq \Delta_1^2 + 3\Delta_1^2 \sum_{k=1}^\infty \left( \frac{1}{4} \right)^k + 48\alpha^2 H^2 \zeta^2 (K - 1) + 48\alpha^2 n^2 \sum_{k=2}^K \|\nabla f(w_k)\|_\infty^2 \\\n\leq 16\alpha^2 n^2 \|\nabla f(w_1)\|_\infty^2 + 16\alpha^2 n^2 \zeta^2 + 48\alpha^2 H^2 \zeta^2 (K - 1) + 48\alpha^2 n^2 \sum_{k=2}^K \|\nabla f(w_k)\|_\infty^2 \\\n\leq 16\alpha^2 n^2 \zeta^2 + 48\alpha^2 H^2 \zeta^2 K + 48\alpha^2 n^2 \sum_{k=1}^K \|\nabla f(w_k)\|_\infty^2.
\]

That completes the third inequality, and we have finished proving all three inequalities. \( \square \)

**Lemma 4.** In Algorithm 6, if the learning rate \( \alpha \) fulfills

\[ \alpha \leq \min \left\{ \frac{1}{26(n + A)L_{2,\infty}}, \frac{1}{260n L_\infty} \right\}, \]

then the following inequalities hold:

\[ \Delta_k^2 \leq 16\alpha^2 (8 \cdot 2^{-k} n + A)^2 \zeta^2 \]
\[ + 16\alpha^2 \left[ (8 \cdot 2^{-k} n + A)L_{2,\infty} + 10n L_\infty \right]^2 \Delta_{k-1}^2 \]
\[ + 64\alpha^2 \left[ (8 \cdot 2^{-k} n + A)L_{2,\infty} + 5n L_\infty \right]^2 \Delta_{k-2}^2 + 16\alpha^2 n^2 \|\nabla f(w_{k-1})\|_\infty^2, \forall k \geq 3 \]

and,

\[ \Delta_k^2 \leq 8\alpha^2 n^2 \|\nabla f(w_k)\|_\infty^2 + 8\alpha^2 n^2 \zeta^2, k = 1, 2. \]

and finally,

\[ \sum_{k=1}^K \Delta_k^2 \leq 120\alpha^2 n^2 \zeta^2 + 64\alpha^2 A^2 \zeta^2 K + 48\alpha^2 n^2 \sum_{k=1}^K \|\nabla f(w_k)\|_\infty^2. \]
Proof. With out the loss of generality, for all the \( m \in \{2, \cdots, n + 1\} \) and all the \( k \in \{3, \cdots, K\} \),

\[
w^{(m)}_k = w_k - \alpha \sum_{t=1}^{m-1} \nabla f \left( w^{(t)}_k; x_{\sigma_k(t)} \right)
= w_k - \alpha \sum_{t=1}^{m-1} \nabla f \left( w^{(\sigma_{k-1}^{-1}(\sigma_k(t)))}_{k-1}; x_{\sigma_k(t)} \right)
- \alpha \sum_{t=1}^{m-1} \left( \nabla f \left( w^{(t)}_k; x_{\sigma_k(t)} \right) - \nabla f \left( w^{(\sigma_{k-1}^{-1}(\sigma_k(t)))}_{k-1}; x_{\sigma_k(t)} \right) \right).
\]

Now add and subtract

\[
\alpha \sum_{t=1}^{m-1} \frac{1}{n} \sum_{s=1}^{n} \nabla f \left( w^{(s)}_{k-2}; x_{\sigma_{k-2}(s)} \right) = \frac{(m-1)}{n} \sum_{t=1}^{n} \nabla f \left( w^{(s)}_{k-2}; x_{\sigma_{k-2}(s)} \right),
\]

which gives

\[
w^{(m)}_k = w_k - \alpha \sum_{t=1}^{m-1} \left( \nabla f \left( w^{(\sigma_{k-1}^{-1}(\sigma_k(t)))}_{k-1}; x_{\sigma_k(t)} \right) - \frac{1}{n} \sum_{s=1}^{n} \nabla f \left( w^{(s)}_{k-2}; x_{\sigma_{k-2}(s)} \right) \right)
- \alpha \sum_{t=1}^{m-1} \left( \nabla f \left( w^{(t)}_k; x_{\sigma_k(t)} \right) - \nabla f \left( w^{(\sigma_{k-1}^{-1}(\sigma_k(t)))}_{k-1}; x_{\sigma_k(t)} \right) \right).
\]

We further add and subtract

\[
\frac{(m-1)}{n} \sum_{t=1}^{n} \nabla f \left( w^{(s)}_{k-1}; x_{\sigma_{k-2}(s)} \right) = \alpha (m-1) \nabla f \left( w^{(s)}_{k-1} \right)
\]
to arrive at

\[
w^{(m)}_k = w_k - \alpha \sum_{t=1}^{m-1} \left( \nabla f \left( w^{(\sigma_{k-1}^{-1}(\sigma_k(t)))}_{k-1}; x_{\sigma_k(t)} \right) - \frac{1}{n} \sum_{s=1}^{n} \nabla f \left( w^{(s)}_{k-2}; x_{\sigma_{k-2}(s)} \right) \right)
- \alpha (m-1) \nabla f \left( w^{(s)}_{k-1} \right)
- \alpha \sum_{t=1}^{m-1} \left( \nabla f \left( w^{(t)}_k; x_{\sigma_k(t)} \right) - \nabla f \left( w^{(\sigma_{k-1}^{-1}(\sigma_k(t)))}_{k-1}; x_{\sigma_k(t)} \right) \right).
\]

We can now re-arrange, take norms on both sides and apply the triangle inequality,

\[
\left\| w^{(m)}_k - w_k \right\| \leq \alpha \sum_{t=1}^{m-1} \left\| \nabla f \left( w^{(\sigma_{k-1}^{-1}(\sigma_k(t)))}_{k-1}; x_{\sigma_k(t)} \right) - \frac{1}{n} \sum_{s=1}^{n} \nabla f \left( w^{(s)}_{k-2}; x_{\sigma_{k-2}(s)} \right) \right\|_\infty
+ \alpha (m-1) \left\| \nabla f \left( w^{(s)}_{k-1} \right) \right\|_\infty
+ \alpha \sum_{t=1}^{m-1} \left\| \nabla f \left( w^{(t)}_k; x_{\sigma_k(t)} \right) - \nabla f \left( w^{(\sigma_{k-1}^{-1}(\sigma_k(t)))}_{k-1}; x_{\sigma_k(t)} \right) \right\|_\infty
+ \alpha \sum_{t=1}^{m-1} \left\| \nabla f \left( w^{(t)}_k; x_{\sigma_k(t)} \right) - \nabla f \left( w^{(t)}_{k-2}; x_{\sigma_{k-2}(t)} \right) \right\|_\infty.
\]

Similar to the proof of Lemma 3, for the last two terms, we simply apply the Assumption 1 to obtain

\[
\frac{(m-1)}{n} \sum_{t=1}^{n} \left\| \nabla f \left( w^{(t)}_{k-1}; x_{\sigma_{k-2}(t)} \right) - \nabla f \left( w^{(t)}_{k-2}; x_{\sigma_{k-2}(t)} \right) \right\|_\infty
\]
\[
\alpha \frac{m-1}{n} \sum_{t=1}^{n} \left\| \nabla f \left( w_{k-1}; x_{\sigma_{k-2}(t)} \right) - \nabla f \left( w_{k-2}^{(t)}; x_{\sigma_{k-2}(t)} \right) \right\|_{\infty}
\]
\[
\leq \frac{\alpha (m-1) L_{\infty}}{n} \sum_{t=1}^{n} \left\| w_{k-1} - w_{k-2}^{(t)} \right\|_{\infty}
\]
\[
\leq \frac{\alpha (m-1) L_{\infty}}{n} \sum_{t=1}^{n} \left( \left\| w_{k-1} - w_{k-2} \right\|_{\infty} + \left\| w_{k-2} - w_{k-2}^{(t)} \right\|_{\infty} \right)
\]
\[\leq \alpha (m-1) L_{\infty} (2\Delta_{k-2}), \]

and
\[
\alpha \left\| \sum_{t=1}^{m-1} \left( \nabla f \left( w_{k-1}^{(t)}; x_{\sigma_{k-1}(t)} \right) - \nabla f \left( w_{k-1}^{(t)}; x_{\sigma_{k-2}(t)} \right) \right) \right\|_{\infty}
\]
\[
\leq 2L_{\infty} \sum_{t=1}^{m-1} \left\| w_{k-1}^{(t)} - w_{k-1}^{(\sigma_{k-2}(t))} \right\|_{\infty}
\]
\[
\leq 2L_{\infty} \sum_{t=1}^{m-1} \left( \left\| w_{k-1}^{(t)} - w_{k-1} \right\|_{\infty} + \left\| w_{k-1} - w_{k-2} \right\|_{\infty} + \left\| w_{k-2} - w_{k-2}^{(\sigma_{k-2}(t))} \right\|_{\infty} \right)
\]
\[\leq \alpha (m-1) L_{\infty} (\Delta_{k} + 2\Delta_{k-1}). \]

We then use Lemma 6 to bound the first term, combining them together gives us
\[
\Delta_{k} \leq 2\alpha \left( 8 \cdot 2^{-k} n + A \right) (\zeta + L_{2,\infty} (\Delta_{k-1} + 2\Delta_{k-2})) + 2\alpha n L_{\infty} (\Delta_{k} + 4\Delta_{k-1} + 4\Delta_{k-2})
\]
\[+ \alpha n L_{\infty} (2\Delta_{k-2}) + \alpha n L_{\infty} (\Delta_{k} + 2\Delta_{k-1}) + 4\alpha n \| \nabla f (w_{k-1}) \|_{\infty} \cdot \Delta_{k-2} .
\]

Rearrange and merge the terms, given that \(3\alpha n L_{2,\infty} < 1/2\),
\[
\Delta_{k} \leq 2\alpha \left( 8 \cdot 2^{-k} n + A \right) \zeta
\]
\[+ 2\alpha \left[ \left( 8 \cdot 2^{-k} n + A \right) L_{2,\infty} + 10 n L_{\infty} \right] \Delta_{k-1}
\]
\[+ 4\alpha \left[ \left( 8 \cdot 2^{-k} n + A \right) L_{2,\infty} + 5 n L_{\infty} \right] \Delta_{k-2}
\]
\[+ 2\alpha n \| \nabla f (w_{k-1}) \|_{\infty} .
\]

Now square on both sides, we obtain for any \(k \geq 3\)
\[
\Delta_{k}^{2} \leq 16\alpha^{2} \left( 8 \cdot 2^{-k} n + A \right)^{2} \zeta^{2}
\]
\[+ 16\alpha^{2} \left[ \left( 8 \cdot 2^{-k} n + A \right) L_{2,\infty} + 10 n L_{\infty} \right]^{2} \Delta_{k-1}^{2}
\]
\[+ 16\alpha^{2} \left[ \left( 8 \cdot 2^{-k} n + A \right) L_{2,\infty} + 5 n L_{\infty} \right]^{2} \Delta_{k-2}^{2}
\]
\[+ 16\alpha^{2} n^{2} \| \nabla f (w_{k-1}) \|_{\infty}^{2} .
\]

We have proved the first inequality, and it holds with \(k \geq 3\). We next discuss the \(k = 1, 2\) case, note that these cases follow exactly Equation (6), so that we can reapply the \(k = 1\) case from Lemma 3 and obtain
\[
\Delta_{k}^{2} \leq 8\alpha^{2} n^{2} \| \nabla f (w_{k}) \|_{\infty}^{2} + 8\alpha^{2} n^{2} \zeta^{2} , k = 1, 2.
\]

We have proved the second inequality. Finally, summing from \(k = 1\) to \(K\), we obtain
\[
\sum_{k=1}^{K} \Delta_{k}^{2}
\]
\[= \sum_{k=1}^{2} \Delta_{k}^{2} + \sum_{k=3}^{K} \Delta_{k}^{2}
\]
\[= 8\alpha^{2} n^{2} \sum_{k=1}^{2} \| \nabla f (w_{k}) \|_{\infty}^{2} + 16\alpha^{2} n^{2} \zeta^{2} + 16\alpha^{2} \zeta^{2} \sum_{k=3}^{K} \left( 8 \cdot 2^{-k} n + A \right)^{2}
\]
\[+ 16\alpha^{2} \sum_{k=3}^{K} \left[ \left( 8 \cdot 2^{-k} n + A \right) L_{2,\infty} + 10 n L_{\infty} \right]^{2} \Delta_{k-1}^{2} .
\]
Given the \( \alpha \) fulfills

\[
\alpha \leq \min \left\{ \frac{1}{26(n + A)L_{2,\infty}}, \frac{1}{260nL_{\infty}} \right\},
\]

we obtain,

\[
\sum_{k=1}^{K} \Delta^2_k \leq 120 \alpha^2 n^2 \zeta^2 + 64 \alpha^2 A^2 \zeta^2 K + 48 \alpha^2 n^2 \sum_{k=1}^{K} \|\nabla f(w_k)\|_\infty^2.
\]

That completes the proof. \( \square \)

**Lemma 5.** Consider a group of time-variant input vectors \( z_{1,k}, \ldots, z_{n,k} \in \mathbb{R}^d \) changing over time \( k = 3, \ldots, K \) that satisfy the following conditions:

\[
\|z_{i,k+1} - z_{i,k}\|_{\infty} \leq \alpha, \forall k, i
\]

\[
\left\| \sum_{i=1}^{n} z_{i,k} \right\|_{\infty} \leq b, \forall k
\]

\[
\|z_{i,k}\|_2 \leq c, \forall k.
\]

Now considering a sequence of ordering of \( \{\sigma_k\}_{k=3}^{K} \) that fulfills condition: for any \( k \geq 3 \), \( \sigma_k \) and \( \sigma_{k+1} \) are the input and output of Algorithm 3, then it holds that for any \( 3 \leq k \leq K \),

\[
\max_{1 \leq i \leq n} \left\| \sum_{j=1}^{t} z_{\sigma_k(j),k} \right\|_{\infty} \leq 8 \cdot 2^{-k} cn + A c + 2b + 2an.
\]

**Proof.** We prove this lemma by induction. The bound trivially holds when \( k = 3 \) in that

\[
\max_{1 \leq i \leq n} \left\| \sum_{j=1}^{t} z_{\sigma_1(j),1} \right\|_{\infty} \leq cn \leq 8 \cdot 2^{-3} cn + A c + 2b + 2an.
\]
Suppose the bound holds at time $k \geq 3$, now for $k + 1$, from the triangle inequality we have

$$\max_{1 \leq t \leq n} \left\| \sum_{j=1}^{t} z_{\sigma_{k+1}(j),k+1} \right\|_{\infty} = \max_{1 \leq t \leq n} \left\| \sum_{j=1}^{t} z_{\sigma_{k+1}(j),k} + z_{\sigma_{k+1}(j),k+1} - z_{\sigma_{k+1}(j),k} \right\|_{\infty}$$

$$\leq \max_{1 \leq t \leq n} \left\| \sum_{j=1}^{t} z_{\sigma_{k+1}(j),k} \right\|_{\infty} + \max_{1 \leq t \leq n} \left\| \sum_{j=1}^{t} (z_{\sigma_{k+1}(j),k+1} - z_{\sigma_{k+1}(j),k}) \right\|_{\infty}$$

$$\leq \max_{1 \leq t \leq n} \left\| \sum_{j=1}^{t} z_{\sigma_{k+1}(j),k} \right\|_{\infty} + \max_{1 \leq t \leq n} \left\| \sum_{j=1}^{t} (z_{\sigma_{k+1}(j),k+1} - z_{\sigma_{k+1}(j),k}) \right\|_{\infty}$$

$$\leq \max_{1 \leq t \leq n} \left\| \sum_{j=1}^{t} z_{\sigma_{k+1}(j),k} \right\|_{\infty} + \max_{1 \leq t \leq n} \sum_{j=1}^{t} \left\| z_{\sigma_{k+1}(j),k+1} - z_{\sigma_{k+1}(j),k} \right\|_{\infty}$$

$$\leq \max_{1 \leq t \leq n} \left\| \sum_{j=1}^{t} z_{\sigma_{k+1}(j),k} \right\|_{\infty} + \sum_{j=1}^{t} \left\| z_{\sigma_{k+1}(j),k+1} - z_{\sigma_{k+1}(j),k} \right\|_{\infty} + an.$$
We first need to derive $b + \frac{Ac + B}{2}$.

And so that,

\[
\max_{1 \leq t \leq n} \left\| \sum_{j=1}^{t} z_{\sigma_{k+1}(j),k+1} \right\|_{\infty} \leq \max_{1 \leq t \leq n} \left\| \sum_{j=1}^{t} z_{\sigma_{k+1}(j),k} \right\|_{\infty} + an \\
\leq \frac{1}{2} B + \frac{1}{2} (Ac + 2b + 2an) \\
= \frac{1}{2} (8 \cdot 2^{-k}cn + Ac + 2b + 2an) + \frac{1}{2} (Ac + 2b + 2an) \\
= 8 \cdot 2^{-(k+1)}cn + Ac + 2b + 2an.
\]

That completes the induction. \( \square \)

**Lemma 6.** In Algorithm 4, when $k \in \{3, \ldots, K\}$,

\[
\max_{1 \leq t \leq n} \left\| \sum_{j=1}^{t} \left( \nabla f \left( w_{k-1}^{(\sigma_{k-1}(\sigma(j)))}; x_{\sigma(j)} \right) - \frac{1}{n} \sum_{s=1}^{n} \nabla f \left( w_{k-2}^{(s)}; x_{\sigma_{k-2}(s)} \right) \right) \right\|_{\infty} \\
\leq (8 \cdot 2^{-k}n + A) (c + L_{2,\infty}(\Delta_{k-1} + 2\Delta_{k-2})) + 2nL_{\infty} (\Delta_{k} + 4\Delta_{k-1} + 4\Delta_{k-2}).
\]

**Proof.** We will apply Lemma 5 to the main derivation. Denote

\[
z_{j,k} = \nabla f \left( w_{k-1}^{(\sigma_{k-1}(j))}; x_{j} \right) - \frac{1}{n} \sum_{s=1}^{n} \nabla f \left( w_{k-2}^{(s)}; x_{\sigma_{k-2}(s)} \right),
\]

and so

\[
z_{\sigma_{k}(j),k} = \nabla f \left( w_{k-1}^{(\sigma_{k-1}(j))}; x_{\sigma_{k}(j)} \right) - \frac{1}{n} \sum_{s=1}^{n} \nabla f \left( w_{k-2}^{(s)}; x_{\sigma_{k-2}(s)} \right).
\]

We first need to derive $a$, $b$, $c$ in Lemma 5 in order to apply it, we will repeatedly use Assumption 1. In the derivation, we will also use the relation that $\|a\|_{\infty} \leq \|a\|_{2}$.

First, we derive $a$,

\[
\|z_{j,k+1} - z_{j,k}\|_{\infty} \\
\leq \left\| \nabla f \left( w_{k}^{(\sigma_{k-1}(j))}; x_{j} \right) - \nabla f \left( w_{k}^{(\sigma_{k-1}(j))}; x_{j} \right) \right\|_{\infty} \\
+ \left\| \frac{1}{n} \sum_{s=1}^{n} \nabla f \left( w_{k-1}^{(s)}; x_{\sigma_{k-1}(s)} \right) - \frac{1}{n} \sum_{s=1}^{n} \nabla f \left( w_{k-2}^{(s)}; x_{\sigma_{k-2}(s)} \right) \right\|_{\infty} \\
= \left\| \nabla f \left( w_{k}^{(\sigma_{k-1}(j))}; x_{j} \right) - \nabla f \left( w_{k}^{(\sigma_{k-1}(j))}; x_{j} \right) \right\|_{\infty} \\
+ \left\| \frac{1}{n} \sum_{s=1}^{n} \nabla f \left( w_{k-1}^{(s)}; x_{\sigma_{k-1}(s)} \right) - \frac{1}{n} \sum_{s=1}^{n} \nabla f \left( w_{k-2}^{(s)}; x_{\sigma_{k-2}(s)} \right) \right\|_{\infty} \\
\leq L_{\infty} \left\| w_{k}^{(\sigma_{k-1}(j))} - w_{k}^{(\sigma_{k-1}(j))} \right\|_{\infty} + \frac{L_{\infty}}{n} \sum_{s=1}^{n} \left\| w_{k-1}^{(\sigma_{k-1}(s))} - w_{k-2}^{(\sigma_{k-1}(s))} \right\|_{\infty} \\
\leq L_{\infty} \left( \| w_{k}^{(\sigma_{k-1}(j))} - w_k \|_{\infty} + \| w_k - w_{k-1} \|_{\infty} + \| w_{k-1} - w_{k-1}^{(\sigma_{k-1}(j))} \|_{\infty} \right) \\
+ \frac{L_{\infty}}{n} \sum_{s=1}^{n} \left( \| w_{k-1}^{(\sigma_{k-1}(s))} - w_{k-1} \|_{\infty} + \| w_{k-1} - w_{k-2} \|_{\infty} + \| w_{k-2} - w_{k-2}^{(\sigma_{k-1}(s))} \|_{\infty} \right) \\
\leq L_{\infty} (\Delta_k + 2\Delta_{k-1}) + L_{\infty} (\Delta_{k-1} + 2\Delta_{k-2})
\]
We proceed to derive $b$,
\[
\left\| \sum_{j=1}^{n} z_{j,k} \right\|_{\infty} = \sum_{j=1}^{n} \left\| \nabla f \left( \mathbf{w}_{k-1}^{\sigma_k^{-1}(j)} ; x_j \right) - \frac{1}{n} \sum_{s=1}^{n} \nabla f \left( \mathbf{w}_{k-2}^{\sigma_k^{-1}(s)} ; x_s \right) \right\|_{\infty}
\leq \sum_{j=1}^{n} \left( \nabla f \left( \mathbf{w}_{k-1}^{\sigma_k^{-1}(j)} ; x_j \right) - \frac{1}{n} \sum_{s=1}^{n} \nabla f \left( \mathbf{w}_{k-1}^{\sigma_k^{-1}(s)} ; x_s \right) \right)
+ \sum_{j=1}^{n} \left( \frac{1}{n} \sum_{s=1}^{n} \nabla f \left( \mathbf{w}_{k-1}^{\sigma_k^{-1}(s)} ; x_s \right) - \frac{1}{n} \sum_{s=1}^{n} \nabla f \left( \mathbf{w}_{k-2}^{\sigma_k^{-1}(s)} ; x_s \right) \right)
\leq 0 + \sum_{j=1}^{n} \frac{1}{n} \sum_{s=1}^{n} L_{\infty} \left\| \mathbf{w}_{k-1}^{\sigma_k^{-1}(s)} - \mathbf{w}_{k-2}^{\sigma_k^{-1}(s)} \right\|_{\infty}
\leq n \left( \left\| \mathbf{w}_{k-1} - \mathbf{w}_{k-2} \right\|_{\infty} + \left\| \mathbf{w}_{k-2} - \mathbf{w}_{k-2}^{\sigma_k^{-1}(s)} \right\|_{\infty} \right)
\leq n L_{\infty} (\Delta_{k-1} + 2 \Delta_{k-2}).
\]

And finally for $c$,
\[
\left\| z_{j,k} \right\|_{2} = \left\| \nabla f \left( \mathbf{w}_{k-1}^{\sigma_k^{-1}(j)} ; x_j \right) - \frac{1}{n} \sum_{s=1}^{n} \nabla f \left( \mathbf{w}_{k-2}^{\sigma_k^{-1}(s)} ; x_s \right) \right\|_{2}
\leq \left\| \nabla f \left( \mathbf{w}_{k-1}^{\sigma_k^{-1}(j)} ; x_j \right) - \frac{1}{n} \sum_{s=1}^{n} \nabla f \left( \mathbf{w}_{k-1}^{\sigma_k^{-1}(s)} ; x_s \right) \right\|_{2}
+ \frac{1}{n} \sum_{s=1}^{n} \left\| \nabla f \left( \mathbf{w}_{k-1}^{\sigma_k^{-1}(s)} ; x_s \right) - \frac{1}{n} \sum_{s=1}^{n} \nabla f \left( \mathbf{w}_{k-2}^{\sigma_k^{-1}(s)} ; x_s \right) \right\|_{2}
\leq \varsigma + L_{2,\infty} \left( \left\| \mathbf{w}_{k-1}^{\sigma_k^{-1}(s)} - \mathbf{w}_{k-1} \right\|_{\infty} + \left\| \mathbf{w}_{k-1} - \mathbf{w}_{k-2} \right\|_{\infty} + \left\| \mathbf{w}_{k-2} - \mathbf{w}_{k-2}^{\sigma_k^{-1}(s)} \right\|_{\infty} \right)
\leq \varsigma + L_{2,\infty} (\Delta_{k-1} + 2 \Delta_{k-2}).
\]

Now we have the $a$, $b$ and $c$ ready, we can now apply Lemma 5, and get
\[
\max_{1 \leq t \leq n} \sum_{j=1}^{t} z_{\sigma_k(j),k} \leq (8 \cdot 2^{-k} n + A) \left( \varsigma + L_{2,\infty} (\Delta_{k-1} + 2 \Delta_{k-2}) \right)
+ 2 n L_{\infty} (\Delta_{k-1} + 2 \Delta_{k-2}) + 2 n L_{\infty} (\Delta_{k} + 3 \Delta_{k-1} + 2 \Delta_{k-2})
\leq (8 \cdot 2^{-k} n + A) \left( \varsigma + L_{2,\infty} (\Delta_{k-1} + 2 \Delta_{k-2}) \right) + 2 n L_{\infty} (\Delta_{k} + 4 \Delta_{k-1} + 4 \Delta_{k-2}).
\]

That completes the proof. \qed