PREDICTING BLOCK-COORDINATE PERFORMANCE VIA CROSS-CURVATURE

Anonymous authors
Paper under double-blind review

ABSTRACT

Simultaneous and sequential block updates are two basic optimization strategies used across machine learning, such as neural-network training, federated learning, and low-rank adaptation. Choosing between them is difficult because their relative advantage depends on both the objective geometry and the number of iterations. We develop a unified theory for comparing Jacobi (JC), Gauss–Seidel (GS), and partially sequential deterministic block-gradient updates. Our analysis expresses the one-step loss difference through cross-block curvature, with an $\mathcal{O}(\eta^3)$ remainder, where η is the learning rate. We derive a signed loss comparison after K iterations with $\mathcal{O}(K\eta^3)$ error under regularity conditions and $\eta K \leq T$ for fixed T, identifying the better method when the predicted difference exceeds this error. We evaluate these formulas along observed training trajectories across different machine learning settings. Over 500 iterations, our theory correctly identifies the lower-loss method in 98.0% of iterations for the neural network, 83.4% for federated learning, and 97.6% for LoRA. Applying the loss recursion at each step using the measured parameter difference raises these rates to 100.0%, 93.2%, and 99.6%, respectively.

1 INTRODUCTION

Block-coordinate optimization is widely used in machine learning. Across deep learning, federated learning, large model training and fine-tuning, and many other domains. High-dimensional parameters $\boldsymbol{\theta} \in \mathbb{R}^d$ are typically partitioned into blocks $(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N)$ that are updated iteratively. A fundamental question arises: should blocks update simultaneously or sequentially? This choice gives rise to two classical strategies. In *Jacobi* iteration, all blocks update in parallel using gradients evaluated at the current shared state, which maximizes parallelism but relies on outdated information as other blocks change. In *Gauss-Seidel* iteration, blocks update sequentially, with each block incorporating the latest values from previously updated blocks; this approach sacrifices parallelism but benefits from fresher gradient information. The choice between these strategies, or hybrid schemes mixing both, profoundly affects convergence, yet remains guided by intuition and empirical experience rather than principled theory.

Despite the widespread use of block-coordinate methods, *we lack a unified theory predicting which update scheme performs better under what conditions*. Figure 1 illustrates this challenge in three settings (see Details in Appendix E): (a) an 8-layer MLP on MNIST with layer-wise simultaneous versus GS-like in-place backward updates; (b) federated ResNet-18 on CIFAR-10 with FedAlt (alternating personal/shared updates, GS-like) versus FedSim (simultaneous updates, JC-like); and (c) T5-base LoRA fine-tuning on SST-2 with alternating versus simultaneous updates of the A and B low-rank factors. Across these settings, the recorded training losses favor sequential updates early in training and simultaneous updates later. This reversal motivates a criterion for predicting when and under what conditions the relative advantage changes.

Classical results from numerical linear algebra (Young, 1971; Saad, 2003) analyze Gauss-Seidel versus Jacobi for solving linear systems $Ax = b$, proving that Gauss-Seidel converges faster for symmetric positive definite matrices. However, these results do not extend to nonlinear objectives $\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$, nor do they predict the instance-specific performance gaps visible in Figure 1. Modern optimization theory (Nesterov, 2012; Attouch et al., 2013; Beck, 2015; Wright, 2015; Wang et al., 2019) establishes convergence rates for various block-coordinate descent variants, but focuses on

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

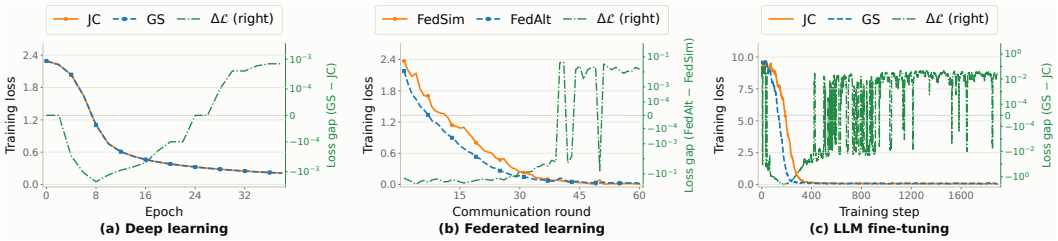


Figure 1: Training loss for (a) an 8-layer MLP on MNIST, (b) federated ResNet-18 on CIFAR-10, and (c) T5-base LoRA fine-tuning on SST-2. Blue and orange curves denote sequential/alternating (GS, FedAlt) and simultaneous (JC, FedSim) updates, respectively. Green dash-dotted curves (right axes) show $\Delta\mathcal{L} = \mathcal{L}_{\mathrm{GS}} - \mathcal{L}_{\mathrm{JC}}$ (FedAlt minus FedSim in panel (b)).

asymptotic guarantees that obscure the fine-grained question: *for a given problem at a given iterate, which update scheme makes more progress, and by how much?* Asynchronous optimization (Lian et al., 2015) and federated learning (Pillutla et al., 2022) literature documents empirical benefits of fresher parameter values but lacks geometric characterizations of when staleness helps versus hurts. The gap between classical linear-system theory and modern nonlinear optimization leaves practitioners without actionable guidance for choosing update schemes in applications.

We develop a curvature-based theory of *iteration selection*: given an objective, a block partition, and an iteration budget, which update scheme should be used to obtain a lower objective value? Our framework compares simultaneous, sequential, and partially sequential block-gradient updates through the interactions between block gradients and cross-block curvature. The resulting criterion is state-dependent: the preferred scheme can change as optimization progresses. Over multiple iterations, the loss gap also depends on how earlier updates have changed the methods' parameter trajectories. We account for this history through trajectory and loss recursions, yielding a cumulative comparison after a prescribed number of iterations. Our contributions are summarized as follows:

- We derive a unified one-step loss comparison for acyclic block-gradient schemes (Theorem 1). From a common starting point, the GS–JC gap is $\frac{\eta^2}{2}\mathcal{C}_{\mathrm{cross}} + \mathcal{O}(\eta^3)$, where $\mathcal{C}_{\mathrm{cross}} = 2\sum_{i<j} \boldsymbol{g}_i^\top \boldsymbol{H}_{ij}\boldsymbol{g}_j$. The sign of cross-curvature identifies the lower-loss one-step update whenever the leading term dominates the remainder.
- We derive trajectory and loss recursions with $\mathcal{O}(\eta^3)$ local remainders (Lemma 3; Theorem 2). Under the stated regularity conditions and $\eta K \le T$ for fixed T, their propagation gives a signed terminal loss comparison with $\mathcal{O}(K\eta^3)$ error (Theorem 3), identifying the lower-loss method when the estimated gap magnitude exceeds the remainder bound.
- We evaluate recursive and cumulative estimates using observed trajectory derivatives in DNN, federated learning, and LoRA experiments (Section 5). Over 500 iterations per setting, cumulative winner agreement is 98.0%, 83.4%, and 97.6%, respectively. The recursive estimate improves winner agreement and loss-gap accuracy in all three settings.

2 Related Work

Coordinate descent methods. Coordinate descent exploits a block partition to reduce the cost of individual updates. Its theory covers cyclic, randomized, and greedy block selection, as well as smooth and composite objectives (Wright, 2015). Tseng (2001) establishes convergence results for nondifferentiable block-coordinate minimization, while Nesterov (2012); Richtárik & Takáč (2014) derive non-asymptotic complexity bounds for randomized coordinate methods. Block successive-minimization frameworks further accommodate inexact updates through local surrogate subproblems (Razaviyayn et al., 2013). These results characterize how objective structure, block selection, and subproblem accuracy affect convergence.

Theory of iterative methods. Classical Jacobi and Gauss–Seidel analyses use matrix splittings and iteration spectra to study stability and convergence rates (Young, 1971; Saad, 2003). Compar-

isons depend on the matrix structure and the update definition; exact block solves and block-gradient steps induce different iteration operators. Asynchronous optimization extends convergence analysis to delayed or inconsistent information, relating admissible parallelism to sparsity, smoothness, and delay assumptions (Recht et al., 2011; Lian et al., 2015). Beyond convergence bounds, Rosca et al. (2021) use backward error analysis to characterize the distinct discretization effects of simultaneous and alternating updates in two-player games. Initialization-based approximations offer another view of training dynamics: Lee et al. (2019) show that, in the infinite-width limit, neural-network evolution is governed by a linearization of the network output around initialization.

Applications in machine learning. Coupled parameter blocks arise throughout learning. Backpropagation computes gradients for layer-structured neural networks (Le Cun, 1988). In federated learning (McMahan et al., 2017), shared representations interact with client-specific parameters: Pillutla et al. (2022) analyze simultaneous FedSim and alternating FedAlt updates under nonconvex objectives and partial participation, identifying regimes favorable to alternation. Low-rank adaptation introduces a different coupling through $\Delta W = BA$ (Hu et al., 2022). LoRA+ studies unequal learning rates for the two factors (Hayou et al., 2024), whereas LoRA-E^2 combines a modified initialization with B-first GS warm-up followed by simultaneous updates (Zhu et al., 2026). These applications illustrate how parameter coupling motivates choices of update order, information freshness, and factor-specific step sizes. Beyond machine learning, the theory of iterative methods also plays a central role in optimization (Richtárik & Takáč, 2016; Xu & Yin, 2017), signal processing (Alaee-Kerahroodi et al., 2017), and computational imaging (Wang et al., 2008; Xu & Yin, 2013).

Taken together, these studies establish convergence guarantees, characterize iterative dynamics, and develop application-specific update rules. Our focus is the complementary problem of *instance-specific iteration selection*: comparing the signed terminal losses of fixed JC and GS methods at a prescribed budget. For deterministic block-gradient updates, cross-curvature yields a one-step comparison with $\mathcal{O}(\eta^3)$ remainder, and signed trajectory propagation extends the comparison to a prescribed number of iterations. This distinguishes local update preference from comparison after multiple iterations without asserting universal superiority of either scheme.

3 PRELIMINARIES

3.1 BLOCK-COORDINATE OPTIMIZATION

We use bold lowercase letters for vectors (e.g., $\boldsymbol{\theta}, \boldsymbol{g}$) and bold uppercase letters for matrices (e.g., $\boldsymbol{H}, \boldsymbol{A}$). For vectors $\boldsymbol{x} \in \mathbb{R}^d$, $\|\boldsymbol{x}\|$ denotes the Euclidean norm. For matrices $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\|\boldsymbol{A}\|$ denotes the spectral norm (largest singular value). We write $[N] = \{1, 2, \ldots, N\}$ for index sets. We consider the optimization problem $\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \mathcal{L}(\boldsymbol{\theta})$ where $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ is twice continuously differentiable with a locally Lipschitz Hessian on a neighborhood containing the updates and their intermediate states. The parameter vector $\boldsymbol{\theta}$ is partitioned into N blocks: $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N)$ with $\boldsymbol{\theta}_i \in \mathbb{R}^{d_i}$ and $\sum_{i=1}^N d_i = d$. At iterate $\boldsymbol{\theta}^k$, we denote the gradient $\boldsymbol{g} = \nabla\mathcal{L}(\boldsymbol{\theta}^k) \in \mathbb{R}^d$ with block components $\boldsymbol{g}_i = \nabla_{\boldsymbol{\theta}_i}\mathcal{L}(\boldsymbol{\theta}^k) \in \mathbb{R}^{d_i}$, and the Hessian $\boldsymbol{H} = \nabla^2\mathcal{L}(\boldsymbol{\theta}^k) \in \mathbb{R}^{d\times d}$ with blocks $\boldsymbol{H}_{ij} = \nabla^2_{\boldsymbol{\theta}_i,\boldsymbol{\theta}_j}\mathcal{L}(\boldsymbol{\theta}^k) \in \mathbb{R}^{d_i\times d_j}$ satisfying $\boldsymbol{H}_{ij} = \boldsymbol{H}_{ji}^\top$ by Schwarz's theorem. We decompose the Hessian as $\boldsymbol{H} = \boldsymbol{D}+\boldsymbol{L}+\boldsymbol{L}^\top$ where $\boldsymbol{D} = \mathrm{diag}(\boldsymbol{H}_{11}, \ldots, \boldsymbol{H}_{NN})$ is block-diagonal and $\boldsymbol{L}$ is strictly lower triangular with $(\boldsymbol{L})_{ij} = \boldsymbol{H}_{ij}$ for $i > j$ and zero otherwise.

Definition 1 (Jacobi and Gauss-Seidel Updates)**.** Given learning rate $\eta > 0$ and current iterate $\boldsymbol{\theta}^k$, we define two fundamental update schemes:

- **Jacobi (JC):** All blocks update in parallel using gradients computed at $\boldsymbol{\theta}^k$: $\boldsymbol{\theta}_i^{k+1} = \boldsymbol{\theta}_i^k - \eta\boldsymbol{g}_i^k$ for all $i \in [N]$, where $\boldsymbol{g}_i^k = \nabla_{\boldsymbol{\theta}_i}\mathcal{L}(\boldsymbol{\theta}^k)$. All blocks use the same parameter state, enabling parallel computation.

- **Gauss-Seidel (GS):** Blocks update sequentially in order $i = 1, 2, \ldots, N$, each using the freshest available values: $\boldsymbol{\theta}_i^{k+1} = \boldsymbol{\theta}_i^k - \eta\tilde{\mathbf{g}}_i^k$, where the fresh gradient $\tilde{\mathbf{g}}_i^k = \nabla_{\boldsymbol{\theta}_i}\mathcal{L}(\tilde{\boldsymbol{\theta}}^{(i)})$ is computed at the partially updated state $\tilde{\boldsymbol{\theta}}^{(i)} = (\boldsymbol{\theta}_1^{k+1}, \ldots, \boldsymbol{\theta}_{i-1}^{k+1}, \boldsymbol{\theta}_i^k, \ldots, \boldsymbol{\theta}_N^k)$. Block i uses new values $\boldsymbol{\theta}_j^{k+1}$ for all $j < i$, creating sequential dependencies.

Remark 1 (Geometric interpretation). The difference between JC and GS is the parameter state used to evaluate each block gradient. For GS in the order $1, \ldots, N$, a first-order Taylor expansion gives

$$\tilde{\mathbf{g}}_i^k = \boldsymbol{g}_i^k + \sum_{j<i} \boldsymbol{H}_{ij}(\boldsymbol{\theta}_j^{k+1} - \boldsymbol{\theta}_j^k) + \mathcal{O}(\eta^2) = \boldsymbol{g}_i^k - \eta \sum_{j<i} \boldsymbol{H}_{ij}\boldsymbol{g}_j^k + \mathcal{O}(\eta^2).$$

Thus, each block update $-\eta \boldsymbol{g}_j^k$ changes the gradient of block i through $\boldsymbol{H}_{ij}$. In vector form, the GS gradient is $\tilde{\mathbf{g}}^k = \boldsymbol{g}^k - \eta \boldsymbol{L}\boldsymbol{g}^k + \mathcal{O}(\eta^2)$, whereas JC uses $\boldsymbol{g}^k$ without this correction.

3.2 General Update Schemes via Delay Patterns

Jacobi and Gauss-Seidel represent two extreme strategies: JC uses completely outdated parameters (all blocks see $\boldsymbol{\theta}^k$), while full GS uses maximally fresh parameters (block i sees all updates $\boldsymbol{\theta}_j^{k+1}$ for $j < i$). However, practical distributed systems often operate between these extremes: some blocks may share fresh values while others remain outdated due to communication delays, asynchronous execution, or deliberate scheduling. To analyze such hybrid schemes that encompass JC, GS, and everything in between, we introduce the delay pattern.

Definition 2 (Delay Pattern)**.** A *delay pattern* is a set $\mathcal{S} \subseteq \{(i,j) : i,j \in [N]\}$, where $(i,j) \in \mathcal{S}$ means that block i uses the fresh value $\boldsymbol{\theta}_j^{k+1}$ rather than $\boldsymbol{\theta}_j^k$ when computing its update. We call $\mathcal{S}$ *acyclic* if the blocks can be ordered so that, for every $(i,j) \in \mathcal{S}$, block j is updated before block i.

To illustrate the expressiveness of this formalism, consider $N = 3$ blocks: Jacobi corresponds to $\mathcal{S} = \emptyset$ (no fresh values), while full GS in order $1 \to 2 \to 3$ gives $\mathcal{S} = \{(2,1),(3,1),(3,2)\}$ (each block uses all preceding updates). Between these extremes lie hybrid schemes such as partial GS with $\mathcal{S} = \{(2,1)\}$ (only block 2 uses fresh $\boldsymbol{\theta}_1$) or a chain pattern $\mathcal{S} = \{(2,1),(3,2)\}$ (block 2 uses fresh $\boldsymbol{\theta}_1$, block 3 uses fresh $\boldsymbol{\theta}_2$). The key insight is that any acyclic pattern $\mathcal{S}$ admits a first-order gradient correction, formalized in the following lemma.

Lemma 1 (Gradient Approximation)**.** *For acyclic $\mathcal{S}$ and small η, the fresh gradient satisfies:*

$$\tilde{\mathbf{g}}_i = \boldsymbol{g}_i - \eta \sum_{j:(i,j)\in\mathcal{S}} \boldsymbol{H}_{ij}\boldsymbol{g}_j + \mathcal{O}(\eta^2).$$

In vector form, $\tilde{\mathbf{g}}^{\mathcal{S}} = \boldsymbol{g} - \eta \boldsymbol{M}_{\mathcal{S}}\boldsymbol{g} + \mathcal{O}(\eta^2)$, where $(\boldsymbol{M}_{\mathcal{S}})_{ij} = \boldsymbol{H}_{ij}$ if $(i,j) \in \mathcal{S}$ and zero otherwise.

This lemma separates the update schedule from the objective geometry: $\mathcal{S}$ selects the fresh-value dependencies, while the retained Hessian determine how those updates change each block gradient. JC has $\boldsymbol{M}_{\mathcal{S}} = 0$, full GS in the order $1, \ldots, N$ has $\boldsymbol{M}_{\mathcal{S}} = \boldsymbol{L}$. The proof is given in Appendix A.1.

4 The Main Theorem

All proofs and intermediate derivations are collected in Appendices A–D. The main text states the results and their interpretation.

4.1 Main Result

We now leverage the delay pattern formalism and the gradient approximation from Lemma 1 to derive a unified second-order expansion that applies to any acyclic update scheme. This expansion reveals how the delay pattern $\mathcal{S}$ induces an extra curvature term $\mathcal{C}_{\mathcal{S}}$ beyond the standard descent, quantifying the performance gap between different update strategies.

Theorem 1 (Main Theorem)**.** *Let $\mathcal{S}$ be acyclic, $\eta > 0$ sufficiently small, and consider both update schemes starting from the same iterate $\boldsymbol{\theta}^k$ with gradient $\boldsymbol{g} = \nabla\mathcal{L}(\boldsymbol{\theta}^k)$ and Hessian $\boldsymbol{H} = \nabla^2\mathcal{L}(\boldsymbol{\theta}^k)$. Then after one step:*

$$\mathcal{L}_{JC}^{k+1} = \mathcal{L}^k - \eta\|\boldsymbol{g}\|^2 + \frac{\eta^2}{2}\boldsymbol{g}^\top \boldsymbol{H}\boldsymbol{g} + \mathcal{O}(\eta^3), \tag{1}$$

$$\mathcal{L}_{\mathcal{S}}^{k+1} = \mathcal{L}^k - \eta\|\boldsymbol{g}\|^2 + \frac{\eta^2}{2}\boldsymbol{g}^\top \boldsymbol{H}\boldsymbol{g} + \eta^2\mathcal{C}_{\mathcal{S}} + \mathcal{O}(\eta^3), \tag{2}$$

where $\mathcal{C}_{\mathcal{S}} = \boldsymbol{g}^\top \boldsymbol{M}_{\mathcal{S}} \boldsymbol{g} = \sum_{(i,j)\in\mathcal{S}} c_{ij}$ *and* $c_{ij} = \boldsymbol{g}_i^\top \boldsymbol{H}_{ij} \boldsymbol{g}_j$. *Therefore:*

$$\boxed{\mathcal{L}_{\mathcal{S}}^{k+1} - \mathcal{L}_{JC}^{k+1} = \eta^2 \mathcal{C}_{\mathcal{S}} + \mathcal{O}(\eta^3).}$$

The theorem isolates the effect of the update scheme: all schemes share the same first-order decrease, while the use of fresh block values contributes the second-order correction $\eta^2 \mathcal{C}_{\mathcal{S}}$. Each term c_{ij} measures the signed interaction between two block gradients through their cross-Hessian block. Consequently, $\mathcal{C}_{\mathcal{S}} < 0$ favors scheme $\mathcal{S}$ over JC, whereas $\mathcal{C}_{\mathcal{S}} > 0$ favors JC, provided the leading term dominates the remainder. This gives a local selection criterion for one complete block sweep from a common starting point. The proof is given in Appendix A.2.

For full GS, let $\mathcal{S}_{\mathrm{GS}} = \{(i,j) : i > j\}$ and define the cross-curvature by

$$\mathcal{C}_{\mathrm{cross}} := 2 \sum_{i<j} \boldsymbol{g}_i^\top \boldsymbol{H}_{ij} \boldsymbol{g}_j = \boldsymbol{g}^\top (\boldsymbol{H} - \boldsymbol{D}) \boldsymbol{g}. \tag{3}$$

The following corollary states the resulting comparison with JC; its derivation and the update-order argument are given in Appendix A.3.

Corollary 1 (GS vs. JC Criterion)**.** *Applying the Main Theorem to the specific case of full Gauss-Seidel versus Jacobi, we obtain:*

$$\mathcal{L}_{GS}^{k+1} - \mathcal{L}_{JC}^{k+1} = \frac{\eta^2}{2} \mathcal{C}_{cross} + \mathcal{O}(\eta^3).$$

When the displayed leading term dominates the remainder, this gives the following sign criterion:

- $\mathcal{C}_{cross} < 0 \implies$ ***GS wins*** *(achieves lower loss than JC). Negative cross-curvature indicates that off-diagonal Hessian blocks couple the gradients cooperatively, meaning fresh parameter updates in earlier blocks help reduce the loss in later blocks.*
- $\mathcal{C}_{cross} > 0 \implies$ ***JC wins*** *(achieves lower loss than GS). Positive cross-curvature indicates that block gradients compete through off-diagonal couplings, and using fresh updates actually increases the loss compared to using stale but consistent gradients.*
- $\mathcal{C}_{cross} = 0 \implies$ ***No difference*** *to* $\mathcal{O}(\eta^2)$*. The two methods are equivalent at the leading order, with differences appearing only at* $\mathcal{O}(\eta^3)$ *or higher.*

4.2 Multi-Step Dynamics

At the common initialization $\boldsymbol{\theta}^0$, the one-step comparison is $\mathcal{L}_{\mathrm{GS}}^1 - \mathcal{L}_{\mathrm{JC}}^1 = \eta^2 \mathcal{C}_{\mathcal{S}_{\mathrm{GS}}}(\boldsymbol{\theta}^0) + \mathcal{O}(\eta^3)$. After step 0, the trajectories diverge: $\boldsymbol{\theta}_{\mathrm{GS}}^k \neq \boldsymbol{\theta}_{\mathrm{JC}}^k$. The Main Theorem no longer applies because it requires both methods to start from the same point. Instead, we track the loss difference through the coupled pair of Equations (5) and (6) stated in Lemma 3 and Theorem 2, in which the trajectory difference is the state variable and the loss difference is read off from it at each step (see Appendix B.3).

4.2.1 Standing assumptions and notation

To compare the two methods over K iterations, we need to control how their iterates separate and how approximation errors accumulate. The following assumptions bound the gradient, Hessian, and Hessian variation throughout both sequences of iterates and the intermediate block updates. We assume $\eta K \leq T$, where $T > 0$ is fixed independently of η and K.

Assumption 1. There is an open convex set Ω containing all block-intermediate states, both trajectories $\{\boldsymbol{\theta}_{\mathcal{S}}^k\}_{k\leq K}$ and $\{\boldsymbol{\theta}_{\mathrm{JC}}^k\}_{k\leq K}$ together with the segments joining $\boldsymbol{\theta}_{\mathrm{JC}}^k$ to $\boldsymbol{\theta}_{\mathcal{S}}^k$, on which $\mathcal{L}$ is three times continuously differentiable with

(A1) $\|\nabla \mathcal{L}(\boldsymbol{\theta})\| \leq G$ (bounded gradients)

(A2) $\|\nabla^2 \mathcal{L}(\boldsymbol{\theta})\| \leq L$ (L-smoothness)

(A3) $\|\nabla^2 \mathcal{L}(\boldsymbol{\theta}) - \nabla^2 \mathcal{L}(\boldsymbol{\theta}')\| \leq \rho \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|$ (Lipschitz Hessian)

for all $\boldsymbol{\theta}, \boldsymbol{\theta}' \in \Omega$. We write $\eta K \leq T$ with $T = \mathcal{O}(1)$.

At iteration k, the differences in parameters and loss between the two methods are

$$\Delta\boldsymbol{\theta}^k := \boldsymbol{\theta}_{\mathcal{S}}^k - \boldsymbol{\theta}_{\mathrm{JC}}^k, \qquad \Delta\mathcal{L}^k := \mathcal{L}(\boldsymbol{\theta}_{\mathcal{S}}^k) - \mathcal{L}(\boldsymbol{\theta}_{\mathrm{JC}}^k).$$

The subscript specifies the method and the superscript specifies the iteration. For example, $\boldsymbol{g}_{\mathcal{S}}^k = \nabla\mathcal{L}(\boldsymbol{\theta}_{\mathcal{S}}^k)$ and $\boldsymbol{H}_{\mathcal{S}}^k = \nabla^2\mathcal{L}(\boldsymbol{\theta}_{\mathcal{S}}^k)$. The matrix $\boldsymbol{M}_{\mathcal{S}}$ uses Hessian blocks evaluated at $\boldsymbol{\theta}_{\mathcal{S}}^k$, although we omit k from its notation. We set $L_{\mathcal{S}} := |\mathcal{S}|L$, where $|\mathcal{S}|$ is the number of pairs in $\mathcal{S}$, so that $\|\boldsymbol{M}_{\mathcal{S}}\| \leq L_{\mathcal{S}}$. The constants in the $\mathcal{O}(\cdot)$ bounds may depend on G, L, ρ, N, T, but are independent of the step size η and iteration index k. The supporting bounds are given in Appendix B.1.

4.2.2 Trajectory and loss recursions

Comparing JC and scheme $\mathcal{S}$ over multiple iterations requires accounting for the different parameter values they produce. The comparison must include both the parameter difference from previous iterations and the effect of fresh block updates at the current iteration. We bound $\Delta\boldsymbol{\theta}^k$ to control approximation errors, then derive how this difference evolves and how it affects the loss difference.

Lemma 2 (Size of the trajectory difference)**.** *Under Assumption 1, with $\Delta\boldsymbol{\theta}^0 = 0$ and $\eta k \leq T$,*

$$\|\Delta\boldsymbol{\theta}^k\| \;\leq\; \eta\, c_1\, \frac{e^{\eta k L} - 1}{L} \;\leq\; \eta\, \frac{c_1(e^{TL} - 1)}{L} \;=\; \mathcal{O}(\eta), \qquad c_1 := L_{\mathcal{S}} G + \eta C_\varepsilon, \tag{4}$$

where C_ε bounds the $\mathcal{O}(\eta^3)$ remainder of Lemma 1.

The lemma shows that the parameters of JC and scheme $\mathcal{S}$ remain within $\mathcal{O}(\eta)$ of each other for all k with $\eta k \leq T$. This bound allows the higher-order terms in the trajectory and loss to be bounded by $\mathcal{O}(\eta^3)$. The proof is given in Appendix B.2. We now derive the update relation for $\Delta\boldsymbol{\theta}^k$.

Lemma 3 (Trajectory Recursion for General Delay Patterns)**.** *Under Assumption 1, let $\mathcal{S}$ be an acyclic delay pattern with associated matrix $\boldsymbol{M}_{\mathcal{S}}$. Suppose $\Delta\boldsymbol{\theta}^0 = 0$ and $\eta K \leq T = \mathcal{O}(1)$. Then for $0 \leq k < K$, the trajectory difference satisfies:*

$$\Delta\boldsymbol{\theta}^{k+1} = \underbrace{(\boldsymbol{I} - \eta\boldsymbol{H}_{JC}^k)\Delta\boldsymbol{\theta}^k}_{\textit{contract inherited}} + \underbrace{\eta^2\boldsymbol{M}_{\mathcal{S}}\boldsymbol{g}_{\mathcal{S}}^k}_{\textit{new forcing}} + \mathcal{O}\big(\eta\|\Delta\boldsymbol{\theta}^k\|^2 + \eta^3\big)\,. \tag{5}$$

The first term propagates the existing trajectory difference through the linearized JC update, while the second introduces the effect of fresh block updates. By Lemma 2, $\|\Delta\boldsymbol{\theta}^k\| = \mathcal{O}(\eta)$, so the displayed remainder is $\mathcal{O}(\eta^3)$. The proof is given in Appendix B.3. Substituting this trajectory recursion into a second-order expansion of $\mathcal{L}$ around $\boldsymbol{\theta}_{\mathrm{JC}}^{k+1}$ yields the following loss recursion.

Theorem 2 (Loss Evolution Recursion)**.** *Under Assumption 1, let $\mathcal{S}$ be an arbitrary acyclic delay pattern with associated staleness matrix $\boldsymbol{M}_{\mathcal{S}}$, and let $\eta K \leq T = \mathcal{O}(1)$ so that Lemma 2 applies. For any step $0 \leq k < K$, define the trajectory difference $\Delta\boldsymbol{\theta}^k := \boldsymbol{\theta}_{\mathcal{S}}^k - \boldsymbol{\theta}_{JC}^k$ and loss difference $\Delta\mathcal{L}^{k+1} := \mathcal{L}_{\mathcal{S}}^{k+1} - \mathcal{L}_{JC}^{k+1}$ between the $\mathcal{S}$-delayed method and Jacobi, where $\boldsymbol{g}_{JC}^k := \nabla\mathcal{L}(\boldsymbol{\theta}_{JC}^k)$, $\boldsymbol{H}_{JC}^k := \nabla^2\mathcal{L}(\boldsymbol{\theta}_{JC}^k)$, and $\boldsymbol{g}_{\mathcal{S}}^k := \nabla\mathcal{L}(\boldsymbol{\theta}_{\mathcal{S}}^k)$. Then the loss difference at step $k+1$ satisfies:*

$$\Delta\mathcal{L}^{k+1} = \underbrace{\boldsymbol{g}_{JC}^{k+1\top}(\boldsymbol{I} - \eta\boldsymbol{H}_{JC}^k)\Delta\boldsymbol{\theta}^k}_{\textit{propagation: } \mathcal{O}(\eta)} + \underbrace{\tfrac{1}{2}(\Delta\boldsymbol{\theta}^k)^\top\boldsymbol{H}_{JC}^k\Delta\boldsymbol{\theta}^k}_{\textit{quadratic: } \mathcal{O}(\eta^2)} + \underbrace{\eta^2\boldsymbol{g}_{JC}^{k+1\top}\boldsymbol{M}_{\mathcal{S}}\boldsymbol{g}_{\mathcal{S}}^k}_{\textit{freshness: } \mathcal{O}(\eta^2)} + \mathcal{O}(\eta^3). \tag{6}$$

The recursion separates the effect of previous updates from that of the current update scheme. The first term carries the existing parameter difference through the linearized JC update and measures its effect on the loss; the second accounts for curvature along this difference; and the third captures the additional effect of fresh block values. Since $\|\Delta\boldsymbol{\theta}^k\| = \mathcal{O}(\eta)$, the first term is $\mathcal{O}(\eta)$ and the other two are $\mathcal{O}(\eta^2)$. When $\Delta\boldsymbol{\theta}^k = 0$, the loss difference reduces to $\eta^2\mathcal{C}_{\mathcal{S}} + \mathcal{O}(\eta^3)$, recovering the common-start comparison in Theorem 1. The proof is given in Appendix C, using Lemma 3 and a second-order Taylor expansion of the loss.

4.3 Multi-Step Cumulative Loss Difference

The preceding recursions describe how the parameter and loss differences change from one iteration to the next. To compare JC and scheme $\mathcal{S}$ after K iterations, we must account for how differences introduced at earlier steps affect later updates, rather than simply adding the one-step comparisons from common starting points. Starting from $\Delta\boldsymbol{\theta}^0 = 0$, we repeatedly apply Lemma 3 to collect these contributions to $\Delta\boldsymbol{\theta}^K$, then substitute the resulting expression into the second-order loss expansion. The following theorem gives the resulting loss comparison, with an accumulated error of $\mathcal{O}(K\eta^3)$ under the stated assumptions.

Theorem 3 (Signed Cumulative Terminal Comparison)**.** *Under Assumption 1, let $\eta K \le T$ and $\mathcal{S}$ be acyclic. Define*

$$\boldsymbol{\Phi}_{K,j+1} = \prod_{\ell=j+1}^{K-1} (\boldsymbol{I} - \eta \boldsymbol{H}_{\mathrm{JC}}^{\ell}), \qquad \boldsymbol{z}_K = \eta^2 \sum_{j=0}^{K-1} \boldsymbol{\Phi}_{K,j+1} \boldsymbol{M}_{\mathcal{S}}^{j} \boldsymbol{g}_{\mathcal{S}}^{j}, \tag{7}$$

where factors with larger indices appear to the left, and we set $\boldsymbol{\Phi}_{K,K} = \boldsymbol{I}$ for $j = K-1$. The matrix $\boldsymbol{M}_{\mathcal{S}}^{j}$ is evaluated at $\boldsymbol{\theta}_{\mathcal{S}}^{j}$. Then

$$\mathcal{L}(\boldsymbol{\theta}_{\mathcal{S}}^{K}) - \mathcal{L}(\boldsymbol{\theta}_{\mathrm{JC}}^{K}) = (\boldsymbol{g}_{\mathrm{JC}}^{K})^{\top} \boldsymbol{z}_K + \tfrac{1}{2} \boldsymbol{z}_K^{\top} \boldsymbol{H}_{\mathrm{JC}}^{K} \boldsymbol{z}_K + \mathcal{O}(K\eta^3). \tag{8}$$

In particular, define $b_K = \eta^2 \sum_{j=0}^{K-1} (1+\eta L)^{K-1-j} \|\boldsymbol{M}_{\mathcal{S}}^{j} \boldsymbol{g}_{\mathcal{S}}^{j}\|$. Then

$$|\Delta\mathcal{L}^K| \le G b_K + \tfrac{L}{2} b_K^2 + \mathcal{O}(K\eta^3). \tag{9}$$

The matrix $\boldsymbol{\Phi}_{K,j+1}$ describes how subsequent updates change the parameter difference introduced at iteration j, and $\boldsymbol{z}_K$ combines these contributions to approximate $\Delta\boldsymbol{\theta}^K$. The two terms in Equation (8) give a signed estimate of the loss difference: a negative estimate favors $\mathcal{S}$ and a positive estimate favors JC. The proof is given in Appendix D.

Together, these results extend the one-step cross-curvature comparison to multiple iterations by accounting for both the accumulation and propagation of update-induced parameter differences. The resulting loss estimate connects local block interactions to the sign and magnitude of the loss gap along the training trajectories. We next evaluate how accurately the recursive and cumulative estimates capture this gap.

5 Experiments

Our experiments evaluate whether the loss-comparison formulas explain which update rule performs better after a given number of iterations. We ask three questions: (i) does the cumulative estimate identify the lower-loss method and capture changes in the winner as K varies; (ii) does it predict the magnitude of the terminal loss gap; and (iii) how does propagating an approximate parameter difference affect accuracy relative to evaluating the loss recursion with the measured difference? These questions assess the retained terms in Theorems 3 and 2 against independently measured JC and GS losses.

We address the same three questions in settings with different block-coupling structures: layer parameters in a DNN, personal and shared parameters in federated learning, and the two low-rank factors in LoRA. In each setting, we compare the cumulative and recursive estimates with the measured GS–JC loss differences over training, using the same metrics for winner agreement and gap-estimation error. All three studies use intermediate gradients and Hessians from the actual training trajectories, providing direct diagnostics of the formulas. We first specify the common evaluation protocol, then compare the results across the three settings for each validation question.

5.1 Evaluation Protocol and Theoretical Estimates

Controlled trajectories. For each task, JC and GS start from the same parameters and optimize the same fixed training objective using full-batch block-gradient descent with a constant learning rate, without momentum or adaptive preconditioning. We evaluate both losses after every complete block sweep $K = 1, \ldots, 500$. Gradients use the entire fixed data subset, dropout is disabled, and batch-normalization running statistics are fixed. There is no stochastic minibatch or client sampling during training. Appendix E describes the experimental details.

Table 1: Winner identification and loss-gap accuracy of the cumulative and recursive estimates at $K = 500$ iterations. Arrows indicate whether higher (↑) or lower (↓) values are better. Bold denotes the best value for each metric within each task, including ties.

Task	Estimate	Correct ↑	JC hits ↑	GS hits ↑	MAE ↓	Max. AE ↓
DNN	Cumulative	98.0%	**45/45**	445/455	5.58×10^{-4}	2.09×10^{-3}
	Recursive	**100.0%**	**45/45**	**455/455**	$\mathbf{1.15\times10^{-4}}$	$\mathbf{4.86\times10^{-4}}$
FL	Cumulative	83.4%	**25/28**	392/472	2.95×10^{-1}	2.28×10^{1}
	Recursive	**93.2%**	**25/28**	**441/472**	$\mathbf{1.79\times10^{-2}}$	$\mathbf{3.86\times10^{-1}}$
LoRA	Cumulative	97.6%	**74/75**	414/425	5.31×10^{-2}	4.54×10^{0}
	Recursive	**99.6%**	**74/75**	**424/425**	$\mathbf{5.97\times10^{-4}}$	$\mathbf{1.01\times10^{-1}}$

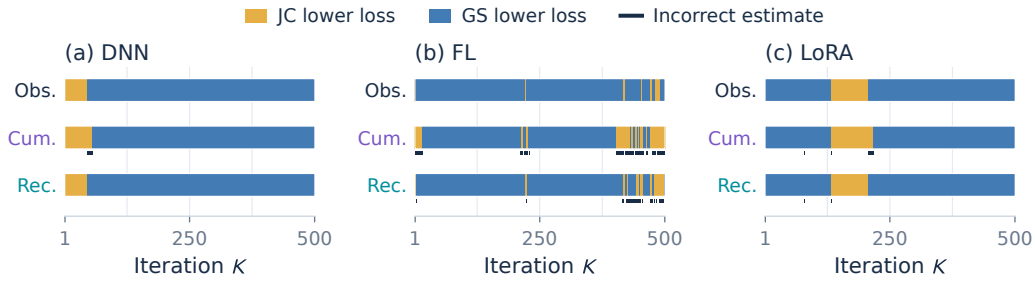


Figure 2: Measured winners (Obs.) and predictions from the cumulative (Cum.) and recursive (Rec.) estimates.

Recursive and cumulative estimates. We compare the measured GS–JC loss gap with the recursive and cumulative estimates from Theorems 2 and 3, respectively. The recursive estimate uses the actual parameter difference from the previous iteration. The cumulative estimate instead propagates an approximate parameter difference from the common initialization. Both use gradients and Hessians evaluated along the actual JC and GS training trajectories.

Metrics and baselines. For a run of K iterations, let $\widehat{\Delta\mathcal{L}}^k$ be the estimated gap at iteration k and $\Delta\mathcal{L}^k = \mathcal{L}(\boldsymbol{\theta}^k_{\mathrm{GS}}) - \mathcal{L}(\boldsymbol{\theta}^k_{\mathrm{JC}})$ the measured gap, for $k = 1, \ldots, K$. Positive gaps favor JC and negative gaps favor GS. For both measured and estimated gaps, define $w(x) = \mathrm{JC}$ when $x > 0$, and $w(x) = \mathrm{GS}$ when $x < 0$. With $\mathbf{1}\{\cdot\}$ denoting an indicator, the winner agreement is $\mathrm{Correct}_K = \frac{100\%}{K}\sum_{k=1}^{K}\mathbf{1}\{w(\widehat{\Delta\mathcal{L}}^k) = w(\Delta\mathcal{L}^k)\}$. For $m \in \{\mathrm{JC}, \mathrm{GS}\}$, define $n_m = \sum_{k=1}^{K}\mathbf{1}\{w(\Delta\mathcal{L}^k) = m\}$ and $h_m = \sum_{k=1}^{K}\mathbf{1}\{w(\Delta\mathcal{L}^k) = m\}\mathbf{1}\{w(\widehat{\Delta\mathcal{L}}^k) = m\}$. The table displays *JC hits* and *GS hits* as the count pairs h_m/n_m. For gap magnitude, let $e_k = |\widehat{\Delta\mathcal{L}}^k - \Delta\mathcal{L}^k|$ and report $\mathrm{MAE}_K = \frac{1}{K}\sum_{k=1}^{K} e_k$ and $\mathrm{Max.\ AE}_K = \max_{1\le k\le K} e_k$. MAE measures average gap-estimation error, whereas Max. AE captures the largest error.

5.2 Validation Across Block-Coupling Structures

We apply the same evaluation protocol to three block structures. For the DNN, we train an eight-layer ReLU MLP on 256 fixed MNIST examples, with each affine layer forming a block and GS updating from input to output. For federated learning, we use a ResNet-18 on CIFAR-10 with three fully participating clients and eight fixed examples per client. Both methods optimize the same mean client objective, with GS updating all personal parameters before the shared parameters. For LoRA, we adapt the query and value projections of T5-base on eight fixed SST-2 examples using rank-eight adapters. The two blocks contain all B factors and all A factors, with GS updating B first.

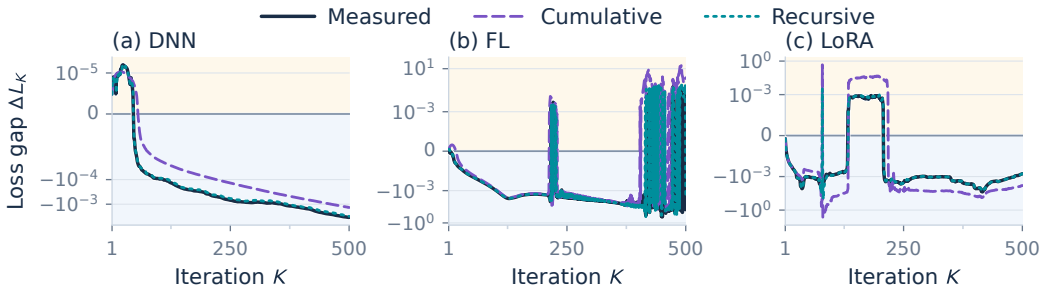


Figure 3: Measured and estimated loss gaps, $\Delta L_K = L_{\mathrm{GS}}^K - L_{\mathrm{JC}}^K$, over 500 iterations. Positive/negative values favor JC/GS.

Identifying the lower-loss method. Table 1 reports overall winner agreement and correct predictions for each method, while Figure 2 shows when errors occur. For the DNN, JC has lower loss for the first 45 iterations and GS thereafter. The cumulative estimate reproduces this ordering but places the transition at $K = 56$ instead of $K = 46$. Its ten errors are therefore confined to $K = 46$–55, giving 98.0% agreement; the recursive estimate matches the measured winner at all 500 iterations.

For federated learning, GS has lower loss at 472 of the 500 iterations, interrupted by brief intervals favoring JC (Figure 2(b)). Both estimates correctly identify 25 of the 28 JC wins. Their difference in agreement comes from the GS intervals: the cumulative estimate incorrectly predicts JC at 80 such iterations, compared with 31 for the recursive estimate. This accounts for the improvement in agreement from 83.4% to 93.2%, with most errors occurring late in training.

For LoRA, GS has lower loss at $K = 1$–132, JC at $K = 133$–207, and GS again at $K = 208$–500 (Figure 2(c)). The cumulative estimate identifies 74 of the 75 JC wins and 414 of the 425 GS wins, giving 97.6% agreement. It places the sustained JC interval at $K = 134$–217: its errors are the one-step delay at $K = 133$, the ten-step delay in returning to GS at $K = 208$–217, and an isolated false JC prediction at $K = 79$. The recursive estimate errors only at $K = 79$ and $K = 133$, reaching 99.6% agreement.

Overall, JC and GS each achieve lower loss at different stages of training in all three settings. Both the recursive and cumulative formulas capture the changes. These results support the use of our theories to identify which method is favored along the observed training trajectories.

Estimating the loss-gap magnitude. The recursive estimate also has a lower mean absolute error (MAE) in all three settings. The cumulative and recursive MAEs are, respectively, 5.58×10^{-4} and 1.15×10^{-4} for the DNN, 0.295 and 0.0179 for federated learning, and 5.31×10^{-2} and 5.97×10^{-4} for LoRA. The maximum absolute errors show the same pattern. For federated learning, they are 22.8 for the cumulative estimate and 0.386 for the recursive estimate (Figure 3(b)). For LoRA, the corresponding values are 4.54 and 1.01×10^{-1} (Figure 3(c)). Overall, the recursive formula estimates the loss gap more accurately than the cumulative formula across all three settings, with lower mean and maximum absolute errors.

6 Conclusion and Future Work

We developed a unified theory for comparing simultaneous, sequential, and partially sequential block-gradient updates. Cross-block curvature characterizes the leading one-step loss difference, while trajectory and loss recursions extend the comparison to multiple iterations. Experiments on neural networks, federated learning, and LoRA show that the resulting estimates capture changes in the lower-loss method along observed training trajectories. The recursive estimate, which uses measured parameter differences, achieves higher winner agreement and lower loss-gap errors than the cumulative estimate in all three settings. Further directions include extending the analysis to stochastic gradients and optimizers with momentum or adaptive step sizes.

AI USE STATEMENT

In this work, we used generative AI tools to assist with writing and language polishing, as well as literature retrieval and discovery of related work. We have reviewed all AI-assisted work, including suggested text and references, and made the final decisions on the manuscript's wording and citations. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Mohammad Alaee-Kerahroodi, Augusto Aubry, Antonio De Maio, Mohammad Mahdi Naghsh, and Mahmoud Modarres-Hashemi. A coordinate-descent framework to design low PSL/ISL sequences. *IEEE Transactions on Signal Processing*, 65(22):5942–5956, 2017.

Hédy Attouch, Jérôme Bolte, and Benar Fux Svaiter. Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward-backward splitting, and regularized gauss-seidel methods. *Mathematical Programming*, 137(1–2):91–129, 2013.

Amir Beck. On the convergence of alternating minimization for convex programming with applications to iteratively reweighted least squares and decomposition schemes. *SIAM Journal on Optimization*, 25(1):185–209, 2015.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. LoRA+: Efficient low rank adaptation of large models. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 17783–17806, 2024.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Yann Le Cun. A theoretical framework for back-propagation. In *Proceedings of the 1988 Connectionist Models Summer School*, pp. 21–28. Morgan Kaufmann, 1988.

Jaehoon Lee, Lechao Xiao, Samuel S. Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

Xiangru Lian, Yijun Huang, Yuncheng Li, and Ji Liu. Asynchronous parallel stochastic gradient for nonconvex optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 28, 2015.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1273–1282, 2017.

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

Krishna Pillutla, Kshitiz Malik, Abdel-rahman Mohamed, Michael Rabbat, Maziar Sanjabi, and Lin Xiao. Federated learning with partial model personalization. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, volume 162 of *Proceedings of Machine Learning Research*, pp. 17716–17758, 2022.

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

Benjamin Recht, Christopher Ré, Stephen Wright, and Feng Niu. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 24, 2011.

Peter Richtárik and Martin Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1–2):1–38, 2014.

Peter Richtárik and Martin Takáč. Parallel coordinate descent methods for big data optimization. *Mathematical Programming*, 156(1–2):433–484, 2016.

Mihaela Rosca, Yan Wu, Benoit Dherin, and David G. T. Barrett. Discretization drift in two-player games. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9064–9074, 2021.

Yousef Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2nd edition, 2003.

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

Yilun Wang, Junfeng Yang, Wotao Yin, and Yin Zhang. A new alternating minimization algorithm for total variation image reconstruction. *SIAM Journal on Imaging Sciences*, 1(3):248–272, 2008.

Yu Wang, Wotao Yin, and Jinshan Zeng. Global convergence of ADMM in nonconvex nonsmooth optimization. *Journal of Scientific Computing*, 78:29–63, 2019.

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

Yangyang Xu and Wotao Yin. 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.

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

David M. Young. *Iterative Solution of Large Linear Systems*. Academic Press, 1971.

Shengkun Zhu, Jinshan Zeng, Yiming Wang, Sheng Wang, Yuan Sun, Shangfeng Chen, Yuan Yao, and Qiang Yang. LoRA-E^2: Effective and efficient low-rank adaptation. In *Proceedings of the ACM Web Conference (WWW)*, pp. 7342–7353, 2026.