

Block Coordinate Descent BFGS Method on Convex Non-smooth Problems

Yinyu Yao Michael O'Neill
yyao2@unc.edu mikeoneill@unc.edu

November 26, 2025

Abstract

We introduce Block BFGS, a block coordinate descent adaptation of BFGS for composite objectives with overlapping groups. Curvature is refreshed only on the active block, and a composite directional-derivative Wolfe rule threads steps through non-smoothness—no proximal detours. We prove global convergence under standard assumptions and under a milder smooth extension around the iterates. In head-to-head experiments, Block BFGS matches the accuracy of proximal-gradient methods while delivering substantial runtime gains.

1 Background

Quasi-Newton methods, such as the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm, have been used for long as powerful tools for smooth optimization due to their efficient use of gradient information, yet its potential in non-smooth optimization—a realm traditionally dominated by proximal methods—has long been overlooked. In recent years, an increasing number of research indicates that quasi-Newton methods can also provide significant computational advantages for non-smooth problems. For example, in [1], the authors discovered that BFGS outperforms the subgradient methods on non-smooth problems in terms of both solution accuracy and runtime efficiency. We illustrate this on the classical LASSO:

$$\min_{x \in \mathbb{R}^p} F(x) = \frac{1}{2} \|Ax - b\|_2^2 + \lambda \|x\|_1,$$

with $A \in \mathbb{R}^{n \times p}$, $b \in \mathbb{R}^n$, and $\lambda > 0$. The objective is non-smooth due to the ℓ_1 term. As a baseline we use a standard Proximal Gradient (PG) method, and we then made a simple comparison between BFGS and PG.

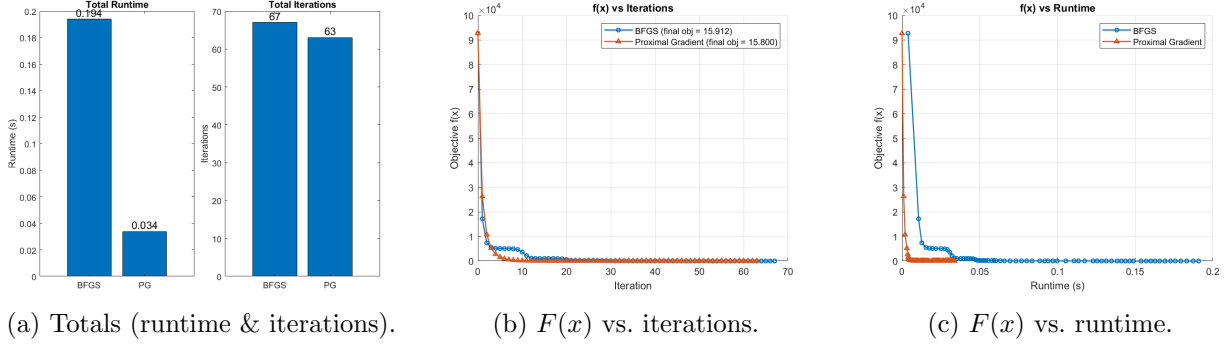


Figure 1: Comparison of BFGS and PG on LASSO.

Preliminary computational result in Figure 1 confirms that BFGS does handle the non-smooth LASSO and converges in a comparable number of steps to PG, but its per-iteration cost makes it much slower in total runtime, as shown in (a), for one typical instance (1000 samples, 100 features). Specifically, both algorithms descend rapidly and converge at the same point within 100 iterations, but BFGS is about 5.7 times slower than PG, with only 4 more iterations, to converge to a very close objective function value as PG. These numerical results indicate significant gains in both speed and solution quality, motivating deeper theoretical and algorithmic exploration.

GroupLASSO Problems

GroupLasso is widely used to impose structured sparsity in real data; for example, a protein-complex-based GroupLasso logistic model (PCLassoLog) has been applied to proteomic data for pan-cancer classification, selecting entire protein complexes rather than isolated proteins [14]. For overlapping groups, Yuan, Liu, and Ye’s *FoGLasso* computes the overlapping-group proximal map by solving a smooth convex dual and embeds it in an accelerated gradient scheme, yielding strong runtime improvements [2]. Dai and Robinson’s *InexactPG* studies proximal-gradient methods when the proximal operator lacks a closed form; for the overlapping group ℓ_1 regularizer, they design an inner solver with adaptive stopping while preserving $\mathcal{O}(\tau^{-2})$ iteration complexity and proving support identification [3].

In what follows, we compare these proximal-gradient approaches with BFGS applied to the same non-smooth objectives, focusing on computation time and solution quality. Concretely, we consider the composite overlapping-group problem

$$\min_{x \in \mathbb{R}^p} F(x) = \frac{1}{2} \|Ax - b\|^2 + \left(\lambda_1 \|x\|_1 + \lambda_2 \sum_{i=1}^m w_i \|x_{G_i}\|_2 \right), \quad (1.1)$$

where $A \in \mathbb{R}^{n \times p}$ and $b \in \mathbb{R}^n$ define the data-fidelity term, $\lambda_1, \lambda_2 \geq 0$ are regularization parameters, $\{G_i\}_{i=1}^m$ are (possibly overlapping) index sets specifying groups, x_{G_i} is the subvector of x on group G_i , and $w_i > 0$ are optional group weights (e.g., to normalize by group size). The penalty combines elementwise sparsity (ℓ_1) with structured sparsity across groups (ℓ_2), and remains convex even when the G_i overlap.

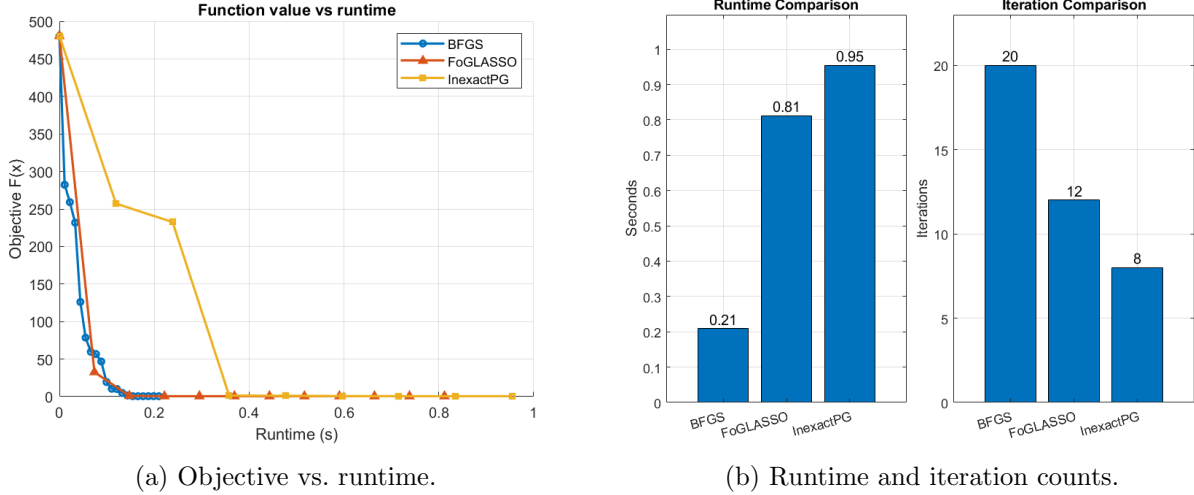


Figure 2: Comparison of BFGS, FoGLASSO, and InexactPG on $n = 100$ samples, $p = 100$ features, and $m = 4$ groups.

From Figure 2, BFGS is the fastest on this instance. It reaches the target tolerance in 0.21s, whereas FoGLASSO and InexactPG take 0.81s and 0.95s, respectively—about $3.9\times$ and $4.5\times$ longer. Panel (a) shows that all three methods attain the same final objective value (within 10^{-4}), while panel (b) illustrates that iteration counts are not directly comparable: BFGS requires 20 iterations versus 12 (FoGLASSO) and 8 (InexactPG) yet still finishes earlier, reflecting differences in per-iteration work and progress per step. These observations motivate a closer examination of when quasi-Newton updates are competitive for non-smooth composite objectives with overlapping group structure and when proximal-gradient variants prevail at larger scales.

2 Block Coordinate Descent BFGS

2.1 Limitations of BFGS

Although BFGS is able to solve non-smooth problems with a comparable accuracy as the proximal gradient methods, however, for large-scale optimization problems, the BFGS algorithm faces significant computational and storage challenges due to the complexity of full Hessian updates [6]. Block coordinate descent methods naturally address these scalability issues by decomposing problems into smaller, more manageable subproblems. Moreover, block coordinate descent techniques can exploit problem structure to reduce per-iteration computational burden and enable parallel or distributed implementations, thereby significantly enhancing efficiency in high-dimensional settings [7]. We scale up the instance from Figure 2:

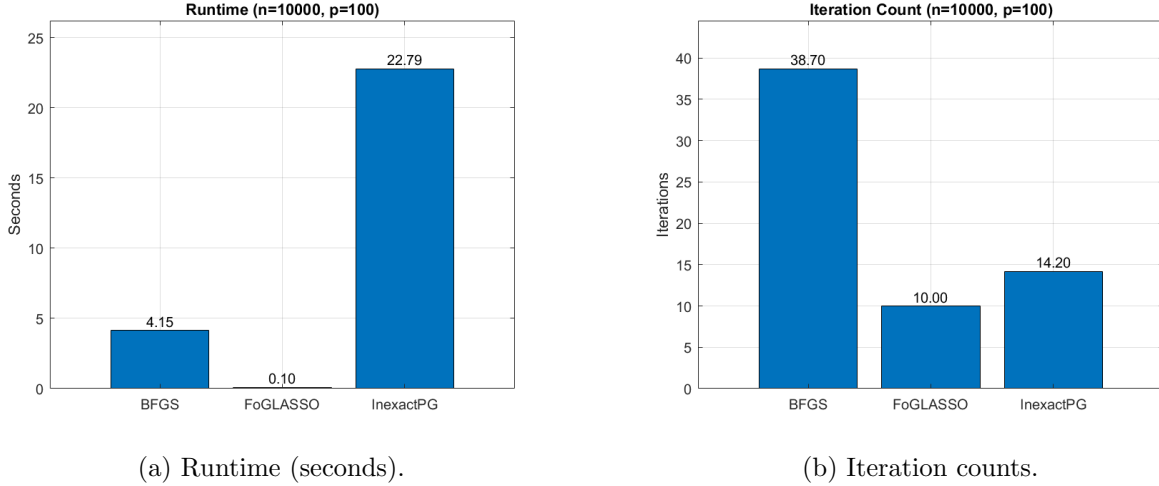


Figure 3: $n = 10,000$ and $p = 100$. BFGS becomes much slower than FoGLASSO.

Compared with Figure 2, the large-scale case in Figure 3 shows a reversal: BFGS becomes significantly slower—about $41.5\times$ slower than FoGLASSO, highlighting that with larger datasets, the algebraic per-iteration cost scales chiefly for full BFGS. Building on these observations and the promising small-scale results, we develop a *Block Coordinate Descent BFGS* method tailored to medium-to-large-scale composite, non-smooth structured problems. By maintaining a block-diagonal Hessian approximation, our approach retains key quasi-Newton properties while reducing per-iteration cost and improving scalability.

2.2 Problem Definition

Throughout this paper, we consider a composite function

$$F(x) = f(x) + g(x), \quad (2.1)$$

where f is smooth and convex, and g is convex (possibly non-smooth). We make the following assumptions and notational conventions.

2.3 Level Set

Given an initial point $x_0 \in \mathbb{R}^n$, define the level set

$$\mathcal{D} = \{x \in \mathbb{R}^p : F(x) \leq F(x_0)\}. \quad (2.2)$$

All iterates $\{x_k\}$ generated by our algorithm remain in \mathcal{D} .

2.4 Strong Convexity of the Smooth Part f

We assume f is twice continuously differentiable on \mathcal{D} . Moreover, there exist constants $0 < m \leq M$ such that

$$m \|z\|^2 \leq z^\top \nabla^2 f(x) z \leq M \|z\|^2, \quad (2.3)$$

$$\forall z \in \mathbb{R}^p, \forall x \in \mathcal{D}.$$

Since f is m -strongly convex, and g is convex, the overall function $F = f + g$ is also m -strongly convex on \mathcal{D} .

2.5 Block Partition of the Variables

We partition $x \in \mathbb{R}^p$ into n_B blocks, writing

$$x = (x^{(1)}, x^{(2)}, \dots, x^{(n_B)}). \quad (2.4)$$

At each iteration k , we select one block index $i_k \in \{1, \dots, n_B\}$ to update (e.g. in random fashion). The other blocks remain temporarily fixed.

2.6 Blockwise Hessian Approximation

We maintain a positive definite matrix $B_k \in \mathbb{R}^{p \times p}$ that is often block-diagonal with one block for each $x^{(i)}$. In practice, B_k is only modified in the rows and columns corresponding to the updated block i_k at iteration k . The submatrix $B_k^{(i_k, i_k)}$ thus serves as a local Hessian approximation for the i_k -th block.

2.7 Algorithm Description

Partial Gradient and Search Direction. At iteration k , we compute the partial gradient of f with respect to block i_k :

$$g_k^{(i_k)} = \nabla_{i_k} f(x_k). \quad (2.5)$$

We then form a search direction $p_k^{(i_k)}$ for this block by

$$p_k^{(i_k)} = - \left[B_k^{(i_k, i_k)} \right]^{-1} g_k^{(i_k)}. \quad (2.6)$$

Wolfe-Type Line Search We seek a step length $\alpha_k > 0$ such that x_{k+1} satisfies the usual Wolfe conditions. Specifically, define $u_{i_k}(\cdot)$ to embed a block vector into the full dimension (all other components zero), and require

$$F(x_k + \alpha_k u_{i_k}(p_k^{(i_k)})) \leq F(x_k) + c_1 \alpha_k (g_k^{(i_k)})^\top p_k^{(i_k)}, \quad (2.7)$$

$$\nabla_{i_k} f(x_k + \alpha_k u_{i_k}(p_k^{(i_k)}))^\top p_k^{(i_k)} \geq c_2 (g_k^{(i_k)})^\top p_k^{(i_k)}, \quad (2.8)$$

where $0 < c_1 < c_2 < 1$ are fixed constants.

Block Update Once α_k is chosen, we update the i_k -th block by

$$x_{k+1}^{(i_k)} = x_k^{(i_k)} + \alpha_k p_k^{(i_k)}, \quad \text{and } x_{k+1}^{(j)} = x_k^{(j)} \text{ for all } j \neq i_k. \quad (2.9)$$

We then define

$$s_k^{(i_k)} = x_{k+1}^{(i_k)} - x_k^{(i_k)}, \quad y_k^{(i_k)} = g_{k+1}^{(i_k)} - g_k^{(i_k)}, \quad (2.10)$$

where $g_{k+1}^{(i_k)} = \nabla_{i_k} f(x_{k+1})$.

Wolfe Conditions and $\cos \theta_k$ Under the Wolfe conditions (2.7)–(2.8), we obtain two useful inequalities:

1. (*Sufficient Decrease*)

$$F(x_{k+1}) \leq F(x_k) + c_1 \alpha_k (g_k^{(i_k)})^\top p_k^{(i_k)}. \quad (2.11)$$

2. (*Curvature*)

$$\nabla_{i_k} f(x_{k+1})^\top p_k^{(i_k)} \geq c_2 (g_k^{(i_k)})^\top p_k^{(i_k)}. \quad (2.12)$$

We also introduce the angle θ_k between $-g_k^{(i_k)}$ and $s_k^{(i_k)}$:

$$\cos \theta_k^{(i_k)} = \frac{(s_k^{(i_k)})^\top (-g_k^{(i_k)})}{\|s_k^{(i_k)}\| \|g_k^{(i_k)}\|}. \quad (2.13)$$

2.8 Algorithm Pseudocode

Algorithm 1 Block Coordinate Descent BFGS (Block BFGS)

Given:

- Block partition $\{x^{(1)}, \dots, x^{(n_B)}\}$ and initial $x_0 \in \mathbb{R}^p$,
- Initial block Hessians $\{B_0^{(i,i)}\}$,
- Wolfe parameters $0 < c_1 < c_2 < 1$,
- Tolerance $\varepsilon > 0$,
- Maximum iterations K_{\max} .

function BLOCK BFGS($f, g, x_0, \{B_0^{(i,i)}\}, \varepsilon, K_{\max}, c_1, c_2$)

$x \leftarrow x_0; \quad k \leftarrow 0; \quad \text{flag} \leftarrow 0$

while $k < K_{\max}$ **do**

$S_k \leftarrow 0$

▷ Accumulate total norm of per-block steps this iteration

$\pi_k \leftarrow \text{RandPerm}(\{1, \dots, n_B\})$

for $i \in \pi_k$ **do**

▷ Random order of blocks in this sweep

Compute partial gradient: $r_k^{(i)} \leftarrow \nabla_i f(x) + \partial_i g(x)$

Search direction: $p_k^{(i)} = -[B_k^{(i,i)}]^{-1} r_k^{(i)}$

Wolfe line-search: Find $\alpha^{(i)}$ such that:

Armijo: $F(x + \alpha^{(i)} u_i(p_k^{(i)})) \leq F(x) + c_1 \alpha^{(i)} (g_k^{(i)})^\top p_k^{(i)}$

Curvature: $\nabla_i F(x + \alpha^{(i)} u_i(p_k^{(i)}))^\top p_k^{(i)} \geq c_2 (g_k^{(i)})^\top p_k^{(i)}$

Update block i : $x^{(i)} \leftarrow x^{(i)} + \alpha^{(i)} p_k^{(i)}; \quad s_k^{(i)} \leftarrow \alpha^{(i)} p_k^{(i)}; \quad S_k \leftarrow S_k + \|s_k^{(i)}\|$

Compute $y_k^{(i)}$ and update BFGS: $y_k^{(i)} = \nabla_i f(x) - \nabla_i f(x - s_k^{(i)})$

if $(y_k^{(i)})^\top s_k^{(i)} > 0$ **then**

$$B_{k+1}^{(i,i)} = B_k^{(i,i)} - \frac{B_k^{(i,i)} s_k^{(i)} (s_k^{(i)})^\top B_k^{(i,i)}}{(s_k^{(i)})^\top B_k^{(i,i)} s_k^{(i)}} + \frac{y_k^{(i)} (y_k^{(i)})^\top}{(y_k^{(i)})^\top s_k^{(i)}}$$

end if

end for

if $S_k \leq \varepsilon$ **then**

$\text{flag} \leftarrow 1; \quad \text{break}$

end if

```

    k ← k + 1
  end while
  return (x, {Bk(i,i)}, flag)
end function

```

2.9 Computational Efficiency

We follow the standard coordinate descent for the smooth part of the composite function $F(x)$, and extend it to variable-metric block updates in the composite setting. For example, the design for least squares follows exactly from [13]: Let x_k be the current iterate and $\{\mathcal{I}_i\}$ a block partition (or cover). We cache the smooth image $w_k = Ax_k$ and, per block, $V_i = A_{\cdot, \mathcal{I}_i}^\top b$, so that a trial on block \mathcal{I}_i updates without re-multiplying Ax :

$$w_k^{\text{trial}}(\Delta x_{\mathcal{I}_i}) = w_k + A_{\cdot, \mathcal{I}_i} \Delta x_{\mathcal{I}_i} \quad (2.14)$$

$$\nabla_{\mathcal{I}_i} f(x_k) = A_{\cdot, \mathcal{I}_i}^\top (w_k - b) = A_{\cdot, \mathcal{I}_i}^\top w_k - V_i. \quad (2.15)$$

Meanwhile, assume the non-smooth (possibly overlapping) group decomposition $g(x) = \sum_{\gamma \in \mathcal{G}} \psi_\gamma(x_{G_\gamma})$, where $\psi_\gamma : \mathbb{R}^{|G_\gamma|} \rightarrow \mathbb{R}$ is convex. For a current block \mathcal{I}_i , define the set of *affected groups*

$$\mathcal{H}_i = \{\gamma \in \mathcal{G} : G_\gamma \cap \mathcal{I}_i \neq \emptyset\}. \quad (2.16)$$

Only these groups can change during a block move. The block subdifferential of g obeys

$$\partial_{\mathcal{I}_i} g(x) = \sum_{\gamma \in \mathcal{H}_i} [\partial \psi_\gamma(x_{G_\gamma})]_{\mathcal{I}_i}, \quad (2.17)$$

so assembling a block (sub)gradient touches only \mathcal{H}_i . Here, we propose two practices:

- *Sequential*: pick a consistent selection of subgradients $v_\gamma(x) \in \partial \psi_\gamma(x_{G_\gamma})$ once per sweep and slice $[\sum_\gamma v_\gamma(x)]_{\mathcal{I}_i}$.
- *Pre-Index (we used)*: pre-index variables to groups $\mathcal{G}(j) = \{\gamma : j \in G_\gamma\}$ and update only $\{x_{G_\gamma}\}_{\gamma \in \mathcal{H}_i}$, costing $O(\sum_{\gamma \in \mathcal{H}_i} |G_\gamma \cap \mathcal{I}_i|)$.

Block line search. Let $u_i : \mathbb{R}^{|\mathcal{I}_i|} \rightarrow \mathbb{R}^n$ embed a block vector with zeros elsewhere. Form a composite block residual

$$g^{(i)} \in \nabla_{\mathcal{I}_i} f(x_k) + \partial_{\mathcal{I}_i} g(x_k),$$

choose a SPD block metric $B_i \succ 0$, and set the search direction $p^{(i)} = -B_i^{-1} g^{(i)}$. Define $\phi(\alpha) = F(x_k + \alpha u_i(p^{(i)}))$ for $\alpha \geq 0$. We use weak Wolfe conditions stated with the *one-sided* directional derivative

$$\phi'_+(0) = \langle \nabla f(x_k), u_i(p^{(i)}) \rangle + Dg(x_k; u_i(p^{(i)})), \quad Dg(x; d) := \lim_{\tau \downarrow 0} \frac{g(x + \tau d) - g(x)}{\tau},$$

and analogously $\phi'_+(\alpha)$ at $x_k + \alpha u_i(p^{(i)})$. Given $0 < c_1 < c_2 < 1$, accept $\alpha > 0$ if

$$\phi(\alpha) \leq \phi(0) + c_1 \alpha \phi'_+(0) \quad \text{and} \quad \phi'_+(\alpha) \geq c_2 \phi'_+(0). \quad (2.18)$$

All quantities in (2.18) are computed *block-locally*:

$$\begin{aligned}
(\text{smooth}) \quad w_k^{\text{trial}}(\alpha p^{(i)}) &= w_k + A_{\mathcal{I}_i}(\alpha p^{(i)}), \quad f(x_k + \alpha u_i(p^{(i)})) = \frac{1}{2} \|w_k^{\text{trial}}(\alpha p^{(i)}) - b\|_2^2, \\
(\text{non-smooth}) \quad g(x_k + \alpha u_i(p^{(i)})) &= \sum_{\gamma \in \mathcal{H}_i} \psi_\gamma(x_{G_\gamma} + \alpha p_{G_\gamma}^{(i)}) + \text{const.}, \\
Dg(x_k + \alpha u_i(p^{(i)}); u_i(p^{(i)})) &= \sum_{\gamma \in \mathcal{H}_i} D\psi_\gamma(x_{G_\gamma} + \alpha p_{G_\gamma}^{(i)}; p_{G_\gamma}^{(i)}).
\end{aligned}$$

Cheap backtracking for group norms. For $\psi_\gamma(z) = \lambda w_\gamma \|z\|_2$, precompute on entry to block \mathcal{I}_i

$$a_\gamma = \|x_{G_\gamma}\|_2^2, \quad b_\gamma = 2\langle x_{G_\gamma}, p_{G_\gamma}^{(i)} \rangle, \quad c_\gamma = \|p_{G_\gamma}^{(i)}\|_2^2,$$

so each trial updates only scalars:

$$\|x_{G_\gamma} + \alpha p_{G_\gamma}^{(i)}\|_2 = \sqrt{a_\gamma + b_\gamma \alpha + c_\gamma \alpha^2}, \quad D\psi_\gamma = \begin{cases} \lambda w_\gamma \frac{\langle x_{G_\gamma} + \alpha p_{G_\gamma}^{(i)}, p_{G_\gamma}^{(i)} \rangle}{\|x_{G_\gamma} + \alpha p_{G_\gamma}^{(i)}\|_2}, & \text{if } \|x_{G_\gamma} + \alpha p_{G_\gamma}^{(i)}\|_2 > 0, \\ \lambda w_\gamma \|p_{G_\gamma}^{(i)}\|_2, & \text{otherwise.} \end{cases}$$

Block-diagonal variable metric. We maintain a block SPD metric and update only the active block using *smooth* curvature:

$$s^{(i)} = x_{\mathcal{I}_i}^{k+1} - x_{\mathcal{I}_i}^k, \quad y^{(i)} = \nabla_{\mathcal{I}_i} f(x^{k+1}) - \nabla_{\mathcal{I}_i} f(x^k).$$

Whenever $(s^{(i)})^\top y^{(i)} > 0$, apply the BFGS update to B_i . This preserves descent compatibility with a non-smooth g while reducing storage/solves from $O(n^2)$ to $O(\sum_i |\mathcal{I}_i|^2)$. Therefore, a single trial in the line search on block \mathcal{I}_i costs

$$O \left(\underbrace{\text{nnz}(A_{\mathcal{I}_i})}_{\text{smooth update of } w_k} + \underbrace{\sum_{\gamma \in \mathcal{H}_i} |G_\gamma \cap \mathcal{I}_i|}_{\text{one-time group setup}} + \underbrace{|\mathcal{I}_i|^2}_{\text{solve/apply } B_i^{-1}} \right), \quad (2.19)$$

independent of n and of groups disjoint from \mathcal{I}_i . Each *backtrack* thereafter touches $\text{nnz}(A_{\mathcal{I}_i})$ plus $O(|\mathcal{H}_i|)$ scalar updates $(a_\gamma, b_\gamma, c_\gamma)$. In separable cases, \mathcal{H}_i reduces to groups contained in \mathcal{I}_i , recovering the coordinate descent economy in [13]; with overlap, restricting work to \mathcal{H}_i ensures we revisit only genuinely coupled terms, retaining low per-iteration cost even when coordinates are not perfectly separable.

3 Bounds on the Stepsize α_k

In our analysis of the Block BFGS method applied to composite non-smooth optimization problems, an essential ingredient is the control of the stepsize α_k [4]. The line search in our algorithm (via the Wolfe-type conditions) plays a crucial role in ensuring sufficient decrease in the objective function while maintaining adequate progress along each block update. In particular, bounding α_k both from below and above allows us to relate the curvature information—encoded by the local Hessian approximation—to the progress made in the descent step. The following bounds on the stepsize α_k ,

trace analysis, and the subsequent global convergence follow the framework developed in [4], where Byrd, Nocedal, and Yuan established convergence properties for a class of quasi-Newton methods on smooth convex problems.

In this section, we establish bounds on the stepsize α_k in terms of the ratio

$$\frac{(s_k^{(i_k)})^\top B_k^{(i_k, i_k)} s_k^{(i_k)}}{\|s_k^{(i_k)}\|^2},$$

where the constants depend on c_1, c_2, m, M . This analysis assumes one iteration of a block BFGS update, with the partial gradient $g_k^{(i_k)}$, search direction $p_k^{(i_k)} = -[B_k^{(i_k, i_k)}]^{-1} g_k^{(i_k)}$, and a Wolfe line search satisfying the conditions $0 < c_1 < c_2 < 1$.

Lemma 3.1. *There exist positive constants c'_1 and c'_2 (depending on m, M, c_1, c_2) such that*

$$c'_1 \frac{(s_k^{(i_k)})^\top B_k^{(i_k, i_k)} s_k^{(i_k)}}{\|s_k^{(i_k)}\|^2} \leq \alpha_k \leq c'_2 \frac{(s_k^{(i_k)})^\top B_k^{(i_k, i_k)} s_k^{(i_k)}}{\|s_k^{(i_k)}\|^2} \quad (3.1)$$

Proof. Define the block average Hessian

$$\bar{G}^{(i_k)} = \int_0^1 \left[\nabla^2 f(x_k + \tau u_{i_k}(s_k^{(i_k)})) \right]_{i_k, i_k} d\tau. \quad (3.2)$$

By the mean-value theorem in the i_k -th block direction, we have

$$y_k^{(i_k)} = \nabla_{i_k} f(x_{k+1}) - \nabla_{i_k} f(x_k) = \left(\int_0^1 \nabla_{i_k}^2 f(x_k + \tau u_{i_k}(s_k^{(i_k)})) d\tau \right) s_k^{(i_k)},$$

which upon restricting to rows and columns i_k gives

$$y_k^{(i_k)} = \bar{G}^{(i_k)} s_k^{(i_k)}. \quad (3.3)$$

From (2.9) and $p_k^{(i_k)} = -[B_k^{(i_k, i_k)}]^{-1} g_k^{(i_k)}$, we have

$$s_k^{(i_k)} = \alpha_k p_k^{(i_k)} = -\alpha_k [B_k^{(i_k, i_k)}]^{-1} g_k^{(i_k)}. \quad (3.4)$$

Multiplying on the left by $B_k^{(i_k, i_k)}$ yields

$$B_k^{(i_k, i_k)} s_k^{(i_k)} = -\alpha_k g_k^{(i_k)}, \implies \alpha_k g_k^{(i_k)} = -B_k^{(i_k, i_k)} s_k^{(i_k)}. \quad (3.5)$$

i. Lower bound on α_k

Using (3.3) and the second Wolfe condition (2.12), we obtain

$$(s_k^{(i_k)})^\top \bar{G}^{(i_k)} s_k^{(i_k)} = (s_k^{(i_k)})^\top y_k^{(i_k)} \geq -(1 - c_2) (g_k^{(i_k)})^\top s_k^{(i_k)}.$$

On the other hand, from (3.5),

$$(s_k^{(i_k)})^\top B_k^{(i_k, i_k)} s_k^{(i_k)} = -\alpha_k (g_k^{(i_k)})^\top s_k^{(i_k)}.$$

Combining these gives

$$(1 - c_2) (s_k^{(i_k)})^\top B_k^{(i_k, i_k)} s_k^{(i_k)} \leq \alpha_k (s_k^{(i_k)})^\top \bar{G}^{(i_k)} s_k^{(i_k)}.$$

Dividing by $(s_k^{(i_k)})^\top \bar{G}^{(i_k)} s_k^{(i_k)}$ yields

$$\alpha_k \geq (1 - c_2) \frac{(s_k^{(i_k)})^\top B_k^{(i_k, i_k)} s_k^{(i_k)}}{(s_k^{(i_k)})^\top \bar{G}^{(i_k)} s_k^{(i_k)}}.$$

Since $\nabla^2 f(x) \preceq M I$ implies $\bar{G}^{(i_k)} \preceq M I$, we get

$$(s_k^{(i_k)})^\top \bar{G}^{(i_k)} s_k^{(i_k)} \leq M \|s_k^{(i_k)}\|^2,$$

thus

$$\alpha_k \geq \frac{(1 - c_2)}{M} \frac{(s_k^{(i_k)})^\top B_k^{(i_k, i_k)} s_k^{(i_k)}}{\|s_k^{(i_k)}\|^2}. \quad (3.6)$$

ii. Upper bound on α_k

Using Taylor's theorem on f between x_k and $x_{k+1} = x_k + \alpha_k u_{i_k}(p_k^{(i_k)})$, we have

$$f(x_{k+1}) - f(x_k) = \nabla f(x_k)^\top s_k^{(i_k)} + \frac{1}{2} (s_k^{(i_k)})^\top \nabla^2 f(\xi_k) s_k^{(i_k)},$$

for some ξ_k on the segment. Rearrange (2.11) gives

$$c_1 \alpha_k (g_k^{(i_k)})^\top s_k^{(i_k)} \geq (g_k^{(i_k)})^\top s_k^{(i_k)} + \frac{1}{2} (s_k^{(i_k)})^\top \nabla^2 f(\xi_k) s_k^{(i_k)}.$$

Thus

$$\frac{1}{2} (s_k^{(i_k)})^\top \nabla^2 f(\xi_k) s_k^{(i_k)} \leq -(1 - c_1) (g_k^{(i_k)})^\top s_k^{(i_k)} = (1 - c_1) \frac{(s_k^{(i_k)})^\top B_k^{(i_k, i_k)} s_k^{(i_k)}}{\alpha_k},$$

where the last step again uses (3.5). Solving for α_k yields

$$\alpha_k \leq 2(1 - c_1) \frac{(s_k^{(i_k)})^\top B_k^{(i_k, i_k)} s_k^{(i_k)}}{(s_k^{(i_k)})^\top \nabla^2 f(\xi_k) s_k^{(i_k)}}.$$

Since $\nabla^2 f(\xi_k) \succeq m I$ by strong convexity, we have

$$(s_k^{(i_k)})^\top \nabla^2 f(\xi_k) s_k^{(i_k)} \geq m \|s_k^{(i_k)}\|^2, \quad (3.7)$$

thus

$$\alpha_k \leq \frac{2(1 - c_1)}{m} \frac{(s_k^{(i_k)})^\top B_k^{(i_k, i_k)} s_k^{(i_k)}}{\|s_k^{(i_k)}\|^2}. \quad (3.8)$$

Let

$$c'_1 = \frac{(1 - c_2)}{M}, \quad c'_2 = \frac{2(1 - c_1)}{m}.$$

Then (3.6) and (3.8) imply

$$c'_1 \frac{(s_k^{(i_k)})^\top B_k^{(i_k, i_k)} s_k^{(i_k)}}{\|s_k^{(i_k)}\|^2} \leq \alpha_k \leq c'_2 \frac{(s_k^{(i_k)})^\top B_k^{(i_k, i_k)} s_k^{(i_k)}}{\|s_k^{(i_k)}\|^2},$$

completing the proof of (3.1). \square

4 BFGS Update and Trace

In this section, we analyze the structure and properties of the BFGS update as applied to the active block in our block coordinate framework [4]. The BFGS update modifies the Hessian approximation for the selected block using curvature information derived from the previous and current iterates. A key aspect of our analysis is to study how the update affects the trace and determinant of the block Hessian submatrix.

For general Broyden's algorithms, Hessian approximation B_k is updated by

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k} + \phi(s_k^T B_k s_k) v_k v_k^T \quad (4.1)$$

However, for block BFGS update, $\phi = 0$, so a simplified version is:

$$B_{k+1}^{(i_k, i_k)} = B_k^{(i_k, i_k)} - \frac{B_k^{(i_k, i_k)} s_k^{(i_k)} (s_k^{(i_k)})^T B_k^{(i_k, i_k)}}{(s_k^{(i_k)})^T B_k^{(i_k, i_k)} s_k^{(i_k)}} + \frac{y_k^{(i_k)} (y_k^{(i_k)})^T}{(y_k^{(i_k)})^T s_k^{(i_k)}}. \quad (4.2)$$

Taking the trace on both sides, we get

$$\begin{aligned} \text{tr}(B_{k+1}^{(i_k, i_k)}) &= \text{tr}(B_k^{(i_k, i_k)}) - \text{tr}\left(\frac{B_k^{(i_k, i_k)} s_k^{(i_k)} (s_k^{(i_k)})^T B_k^{(i_k, i_k)}}{(s_k^{(i_k)})^T B_k^{(i_k, i_k)} s_k^{(i_k)}}\right) + \text{tr}\left(\frac{y_k^{(i_k)} (y_k^{(i_k)})^T}{(y_k^{(i_k)})^T s_k^{(i_k)}}\right). \\ &= \text{tr}(B_k^{(i_k, i_k)}) - \frac{(s_k^{(i_k)})^T (B_k^{(i_k, i_k)})^2 s_k^{(i_k)}}{(s_k^{(i_k)})^T B_k^{(i_k, i_k)} s_k^{(i_k)}} + \frac{\|y_k^{(i_k)}\|^2}{(y_k^{(i_k)})^T s_k^{(i_k)}}. \end{aligned} \quad (4.3)$$

We now examine how to bound the two terms on the right-hand side.

Bound on $\frac{\|y_k^{(i_k)}\|^2}{(y_k^{(i_k)})^T s_k^{(i_k)}}$

Lemma 4.1. *The following inequality holds:*

$$\frac{\|y_k^{(i_k)}\|^2}{(y_k^{(i_k)})^T s_k^{(i_k)}} \leq M \quad (4.4)$$

Proof. From (3.2), let $\bar{G}^{(i_k), 1/2} \bar{G}^{(i_k), 1/2} = \bar{G}^{(i_k)}$. Then set

$$z_k^{(i_k)} = \bar{G}^{(i_k), 1/2} s_k^{(i_k)}. \quad (4.5)$$

Rewrite the ratio:

$$\frac{\|y_k^{(i_k)}\|^2}{(y_k^{(i_k)})^T s_k^{(i_k)}} = \frac{(s_k^{(i_k)})^T \bar{G}^{(i_k), 2} s_k^{(i_k)}}{(s_k^{(i_k)})^T \bar{G}^{(i_k)} s_k^{(i_k)}} = \frac{(z_k^{(i_k)})^T \bar{G}^{(i_k)} z_k^{(i_k)}}{(z_k^{(i_k)})^T z_k^{(i_k)}} \leq M.$$

□

Bound on $\frac{\|B_k^{(i_k, i_k)} s_k^{(i_k)}\|^2}{(s_k^{(i_k)})^\top B_k^{(i_k, i_k)} s_k^{(i_k)}}$

Lemma 4.2. *The following inequality holds:*

$$\frac{\|B_k^{(i_k, i_k)} s_k^{(i_k)}\|^2}{(s_k^{(i_k)})^\top B_k^{(i_k, i_k)} s_k^{(i_k)}} \geq \frac{\alpha_k}{c'_2 \cos^2(\theta_k^{(i_k)})}. \quad (4.6)$$

Proof. From (3.5), we have:

$$\|B_k^{(i_k, i_k)} s_k^{(i_k)}\|^2 = (\alpha_k)^2 \|g_k^{(i_k)}\|^2.$$

and

$$(s_k^{(i_k)})^\top B_k^{(i_k, i_k)} s_k^{(i_k)} = -\alpha_k (g_k^{(i_k)})^\top s_k^{(i_k)}.$$

Thus,

$$\frac{\|B_k^{(i_k, i_k)} s_k^{(i_k)}\|^2}{(s_k^{(i_k)})^\top B_k^{(i_k, i_k)} s_k^{(i_k)}} = \alpha_k \frac{\|g_k^{(i_k)}\|^2}{|(g_k^{(i_k)})^\top s_k^{(i_k)}|}.$$

From (2.13),

$$(g_k^{(i_k)})^\top s_k^{(i_k)} = -\|g_k^{(i_k)}\| \|s_k^{(i_k)}\| \cos \theta_k^{(i_k)} \Rightarrow \frac{\|B_k^{(i_k, i_k)} s_k^{(i_k)}\|^2}{(s_k^{(i_k)})^\top B_k^{(i_k, i_k)} s_k^{(i_k)}} = \alpha_k \frac{\|g_k^{(i_k)}\|}{\|s_k^{(i_k)}\| \cos \theta_k^{(i_k)}}. \quad (4.7)$$

From (3.7) and (4.7),

$$\begin{aligned} \frac{1}{2} m \|s_k^{(i_k)}\|^2 &\leq -(1 - c_1) (g_k^{(i_k)})^\top s_k^{(i_k)} \\ &\Rightarrow \frac{1}{2} m \|s_k^{(i_k)}\|^2 \leq (1 - c_1) \|g_k^{(i_k)}\| \|s_k^{(i_k)}\| \cos \theta_k^{(i_k)} \\ &\Rightarrow \|s_k^{(i_k)}\| \leq \frac{2(1 - c_1)}{m} \|g_k^{(i_k)}\| \cos \theta_k^{(i_k)} \\ &\Rightarrow \|s_k^{(i_k)}\| \leq c'_2 \|g_k^{(i_k)}\| \cos \theta_k^{(i_k)} \\ &\Rightarrow \frac{1}{\|s_k^{(i_k)}\| \cos \theta_k^{(i_k)}} \geq \frac{1}{c'_2 \|g_k^{(i_k)}\| \cos^2 \theta_k^{(i_k)}}. \end{aligned}$$

Thus,

$$\frac{\|B_k^{(i_k, i_k)} s_k^{(i_k)}\|^2}{(s_k^{(i_k)})^\top B_k^{(i_k, i_k)} s_k^{(i_k)}} \geq \frac{\alpha_k}{c'_2 \cos^2(\theta_k^{(i_k)})}.$$

□

Hence, from (4.4) and (4.6), we have

$$\text{tr}(B_{k+1}^{(i_k, i_k)}) \leq \text{tr}(B_k^{(i_k, i_k)}) + M - \frac{\alpha_k}{c'_2 \cos^2(\theta_k^{(i_k)})} \quad (4.8)$$

With the information obtained so far, we now analyze the block-angle $\cos \theta_k^{(i_k)}$. Recall from (3.5) that

$$g_k^{(i_k)} = -\frac{1}{\alpha_k} B_k^{(i_k, i_k)} s_k^{(i_k)} \quad \text{and} \quad \|g_k^{(i_k)}\| = \frac{\|B_k^{(i_k, i_k)} s_k^{(i_k)}\|}{\alpha_k}.$$

Then, by the definition (2.13) of the block cosine,

$$\cos \theta_k^{(i_k)} = \frac{\frac{1}{\alpha_k} (s_k^{(i_k)})^\top B_k^{(i_k, i_k)} s_k^{(i_k)}}{\frac{1}{\alpha_k} \|B_k^{(i_k, i_k)} s_k^{(i_k)}\| \|s_k^{(i_k)}\|} = \frac{(s_k^{(i_k)})^\top B_k^{(i_k, i_k)} s_k^{(i_k)}}{\|B_k^{(i_k, i_k)} s_k^{(i_k)}\| \|s_k^{(i_k)}\|}. \quad (4.9)$$

We can factor this expression as

$$\cos \theta_k^{(i_k)} = \left(\frac{\|s_k^{(i_k)}\|}{\|B_k^{(i_k, i_k)} s_k^{(i_k)}\|} \right) \left(\frac{(s_k^{(i_k)})^\top B_k^{(i_k, i_k)} s_k^{(i_k)}}{\|s_k^{(i_k)}\|^2} \right).$$

It is easy to lower-bound each factor by known information: By Lemma 3.1, we have

$$\frac{(s_k^{(i_k)})^\top B_k^{(i_k, i_k)} s_k^{(i_k)}}{\|s_k^{(i_k)}\|^2} \geq \frac{\alpha_k}{c'_2}. \quad (4.10)$$

Also, for any symmetric positive definite matrix, we have

$$\|B_k^{(i_k, i_k)} s_k^{(i_k)}\| \leq \|B_k^{(i_k, i_k)}\| \|s_k^{(i_k)}\| \leq \text{tr}(B_k^{(i_k, i_k)}) \|s_k^{(i_k)}\|.$$

Hence,

$$\frac{\|s_k^{(i_k)}\|}{\|B_k^{(i_k, i_k)} s_k^{(i_k)}\|} \geq \frac{1}{\text{tr}(B_k^{(i_k, i_k)})}. \quad (4.11)$$

Multiplying the two lower bounds from (4.10) and (4.11), we obtain

$$\cos \theta_k^{(i_k)} \geq \frac{\alpha_k}{c'_2 \text{tr}(B_k^{(i_k, i_k)})}, \quad (4.12)$$

thus $\cos \theta_k^{(i_k)}$ can be small only if α_k is small or $\text{tr}(B_k^{(i_k, i_k)})$ is large. Therefore, we will have to consider two cases: (1) suppose the stepsize α_k are bounded away from zero; (2) suppose the stepsize α_k tend to zero.

5 Global Convergence

Inequality (4.12) shows that the block cosine can be small only if either the stepsize α_k is small or the trace $\text{tr}(B_k^{(i_k, i_k)})$ is large. Consequently, following [4], we distinguish two cases:

- (I) The stepsizes α_k tend to zero;
- (II) The stepsizes α_k are bounded away from zero.

5.1 Random block sampling

At iteration k , a single block index $i_k \in \{1, \dots, n_B\}$ is drawn i.i.d. uniformly at each iteration, independently of the past (i.e., conditionally on the filtration $\mathcal{F}_k := \sigma(x_0, \dots, x_k, B_0, \dots, B_k)$). For a fixed block i , define the block-activity indicator

$$\mathbf{1}_k^{(i)} := \mathbf{1}\{i_k = i\}, \quad N_i(K) := \sum_{k=0}^K \mathbf{1}_k^{(i)},$$

the number of times block i is updated up to iteration K .

5.2 α_k tend to zero

In this case we show that the stepsize cannot be arbitrarily small on average. This is accomplished via a determinant argument adapted to the Block BFGS update. Recall that the BFGS update for the active block is

$$B_{k+1}^{(i_k, i_k)} = B_k^{(i_k, i_k)} - \frac{B_k^{(i_k, i_k)} s_k^{(i_k)} (s_k^{(i_k)})^\top B_k^{(i_k, i_k)}}{(s_k^{(i_k)})^\top B_k^{(i_k, i_k)} s_k^{(i_k)}} + \frac{y_k^{(i_k)} (y_k^{(i_k)})^\top}{(y_k^{(i_k)})^\top s_k^{(i_k)}}.$$

and by applying the Sherman-Morrison Determinant formula, we obtain

$$\det(B_{k+1}) = \det(B_k) \frac{(y_k^{(i_k)})^\top s_k^{(i_k)}}{(s_k^{(i_k)})^\top B_k^{(i_k, i_k)} s_k^{(i_k)}}. \quad (5.1)$$

Note that when $(s_k^{(i_k)})^\top B_k^{(i_k, i_k)} s_k^{(i_k)}$ is small relative to $(y_k^{(i_k)})^\top s_k^{(i_k)}$, the determinant increases, reflecting the fact that the small curvature of our model is corrected, thus increasing some eigenvalues.

Lemma 5.1. *There is a constant $c_4 \in (0, 1)$ and N_0 such that for all $k \geq N_0$*

$$\prod_{j=1}^k \alpha_j \geq c_4^k. \quad (5.2)$$

Proof. By the curvature condition (2.12), we have

$$(y_k^{(i_k)})^\top s_k^{(i_k)} \geq (1 - c_2) \left[-g_k^{(i_k)} \right]^\top s_k^{(i_k)}.$$

Moreover, from (3.5), we deduce that

$$(s_k^{(i_k)})^\top B_k^{(i_k, i_k)} s_k^{(i_k)} = \alpha_k \left[-g_k^{(i_k)} \right]^\top s_k^{(i_k)}.$$

Dividing the two inequalities gives

$$\frac{(y_k^{(i_k)})^\top s_k^{(i_k)}}{(s_k^{(i_k)})^\top B_k^{(i_k, i_k)} s_k^{(i_k)}} \geq \frac{1 - c_2}{\alpha_k}. \quad (5.3)$$

Substituting this bound into (5.1) yields

$$\begin{aligned} \det(B_{k+1}) &= \det(B_k) \frac{(y_k^{(i_k)})^\top s_k^{(i_k)}}{(s_k^{(i_k)})^\top B_k^{(i_k, i_k)} s_k^{(i_k)}} \\ &\geq \det(B_k) \frac{1 - c_2}{\alpha_k}. \end{aligned} \quad (5.4)$$

Iterating this inequality from $j = 1$ to k gives

$$\det(B_{k+1}) \geq \det(B_1) \prod_{j=1}^k \frac{1 - c_2}{\alpha_j}.$$

That is,

$$\prod_{j=1}^k \frac{1}{\alpha_j} \leq \frac{1}{(1 - c_2)^k} \frac{\det(B_{k+1})}{\det(B_1)}.$$

For Block BFGS only block i_k changes at iteration k , so

$$\text{tr}(B_{k+1}) - \text{tr}(B_k) = \text{tr}(B_{k+1}^{(i_k, i_k)}) - \text{tr}(B_k^{(i_k, i_k)}).$$

Hence, by summing over $k = 1, 2, \dots$ on the block trace recursion (4.8) yields

$$\text{tr}(B_{k+1}) \leq \text{tr}(B_1) + k M - \sum_{j=1}^k \frac{\alpha_j}{c'_2 \cos^2 \theta^{(i_j)}}. \quad (5.5)$$

Hence,

$$\det(B_{k+1}) \leq \left(\frac{\text{tr}(B_1) + kM}{n} \right)^n.$$

Substituting into the previous inequality gives

$$\prod_{j=1}^k \frac{1}{\alpha_j} \leq \frac{1}{(1 - c_2)^k \det(B_1)} \left(\frac{\text{tr}(B_1) + kM}{n} \right)^n.$$

Taking reciprocals yields

$$\prod_{j=1}^k \alpha_j \geq (1 - c_2)^k \frac{\det(B_1)}{\left(\frac{\text{tr}(B_1) + kM}{n} \right)^n}$$

Since $\det(B_1)$, n , M , and $(1 - c_2)$ are independent of k , we may combine them into a single constant $c_4 \in (0, 1)$ and N_0 such that for all $k \geq N_0$

$$\prod_{j=1}^k \alpha_j \geq c_4^k. \quad (5.6)$$

□

Per-block determinant growth. For the active block, the determinant update refines to

$$\det(B_{k+1}^{(i_k, i_k)}) = \det(B_k^{(i_k, i_k)}) \frac{(y_k^{(i_k)})^\top s_k^{(i_k)}}{(s_k^{(i_k)})^\top B_k^{(i_k, i_k)} s_k^{(i_k)}} \geq \det(B_k^{(i_k, i_k)}) \frac{1 - c_2}{\alpha_k}. \quad (5.7)$$

For a fixed block i , define the active- i index sets

$$\mathcal{K}_i(K) := \{ 0 \leq k \leq K : i_k = i \}, \quad \mathcal{K}_i := \{ k \geq 0 : i_k = i \}, \quad N_i(K) := |\mathcal{K}_i(K)|.$$

Let $n_i := |I_i|$ denote the size of block i . Iterating (5.7) over the updates in $\mathcal{K}_i(K)$ gives

$$\det(B_{K+1}^{(i, i)}) = \det(B_0^{(i, i)}) \prod_{k \in \mathcal{K}_i(K)} \frac{(y_k^{(i)})^\top s_k^{(i)}}{(s_k^{(i)})^\top B_k^{(i, i)} s_k^{(i)}} \geq \det(B_0^{(i, i)}) \prod_{k \in \mathcal{K}_i(K)} \frac{1 - c_2}{\alpha_k},$$

where the inequality follows from (5.3). Next, apply AM–GM to the $n_i \times n_i$ SPD matrix $B_{K+1}^{(i, i)}$, and then invoke the per-block trace bound—obtained by summing (4.8) over the active- i indices $\mathcal{K}_i(K)$ and dropping the nonpositive term:

$$\det(B_{K+1}^{(i, i)}) \leq \left(\frac{1}{n_i} \text{tr}(B_{K+1}^{(i, i)}) \right)^{n_i} \leq \left(\frac{1}{n_i} [\text{tr}(B_0^{(i, i)}) + M N_i(K)] \right)^{n_i}. \quad (5.8)$$

Combining the two displays and rearranging,

$$\prod_{k \in \mathcal{K}_i(K)} \alpha_k \geq (1 - c_2)^{N_i(K)} \frac{\det(B_0^{(i,i)})}{\left(\frac{1}{n_i} [\text{tr}(B_0^{(i,i)}) + M N_i(K)]\right)^{n_i}}.$$

Since the denominator grows only polynomially in $N_i(K)$ whereas $(1 - c_2)^{N_i(K)}$ decays geometrically, there exist $c_4^{(i)} \in (0, 1)$ and K_0 such that, for all $K \geq K_0$,

$$\prod_{\substack{0 \leq k \leq K \\ i_k = i}} \alpha_k \geq (c_4^{(i)})^{N_i(K)}. \quad (5.9)$$

Consequently, for any $\varepsilon \in (0, c_4^{(i)})$, there exists a subsequence of active- i indices $\{k_\ell^{(i)}\}_{\ell \geq 1}$ with $\alpha_{k_\ell}(i) \geq \varepsilon$ for all ℓ .

5.3 α_k bounded away from zero

Assume there exists $\alpha_{\min} > 0$ and an infinite subsequence $\mathcal{K} \subset \mathbb{N}$ such that $\alpha_k \geq \alpha_{\min}$ for all $k \in \mathcal{K}$. We show, *per block*, that the cosine cannot vanish.

Per-block trace recursion. Fix a block i . Sum the block-trace inequality (B.8) only over the updates of block i , i.e., over the active- i index set,

$$\text{tr}(B_{K+1}^{(i,i)}) \leq \text{tr}(B_0^{(i,i)}) + M N_i(K) - \sum_{\substack{0 \leq k \leq K \\ i_k = i}} \frac{\alpha_k}{c_2' \cos^2 \theta_k^{(i)}}. \quad (5.10)$$

Theorem 5.1 (Global convergence of Block BFGS). *Under the standing assumptions stated earlier and i.i.d. uniform block sampling, the iterates $\{x_k\}$ converge almost surely to the unique minimizer x^* . Moreover, $f(x_k) \downarrow f(x^*)$.*

Proof. **Fix a block i** and choose $\varepsilon \in (0, c_4^{(i)})$ from (5.9). Define the α -large active- i set

$$\mathcal{K}_i^{\geq \varepsilon} := \{k \geq 0 : i_k = i, \alpha_k \geq \varepsilon\}.$$

By contradiction, we consider the following disjunction:

(I) $N_i(K) \not\rightarrow \infty$.

(II) $\cos \theta_{k_\ell^{(i)}}^{(i)} \rightarrow 0$ along $\{k_\ell^{(i)}\}_{\ell \geq 1}$.

(I): Let $E_k^{(i)} := \{i_k = i\}$; the events $E_k^{(i)}$ are independent with $\mathbb{P}(E_k^{(i)}) = 1/n_B$. Since $\sum_{k=0}^{\infty} \mathbb{P}(E_k^{(i)}) = \infty$, the second Borel–Cantelli lemma yields

$$\mathbb{P}(E_k^{(i)} \text{ i.o.}) = 1, \quad \text{i.e., } N_i(K) \rightarrow \infty \text{ almost surely.}$$

Thus (I) cannot occur.

(II): Because $N_i(K) \rightarrow \infty$ a.s. and $\varepsilon \in (0, c_4^{(i)})$, the product bound (5.9) implies that $\mathcal{K}_i^{\geq \varepsilon}$ is infinite a.s.; enumerate it as $\{k_\ell^{(i)}\}_{\ell \geq 1}$. Suppose, for contradiction, that

$$\cos \theta_{k_\ell^{(i)}}^{(i)} \rightarrow 0 \quad (\ell \rightarrow \infty).$$

Evaluate (5.10) at $K_\ell := k_\ell^{(i)}$. Since

$$\{k \in \mathbb{N} : 0 \leq k \leq K_\ell, i_k = i\} \quad \text{contains the first } \ell \text{ indices of } \{k_t^{(i)}\}_{t \geq 1},$$

we obtain

$$\text{tr}(B_{K_\ell+1}^{(i,i)}) \leq \text{tr}(B_0^{(i,i)}) + M N_i(K_\ell) - \frac{1}{c_2'} \sum_{t=1}^{\ell} \frac{\alpha_{k_t^{(i)}}}{\cos^2 \theta_{k_t^{(i)}}^{(i)}}. \quad (5.11)$$

Because $\alpha_{k_t^{(i)}} \geq \varepsilon$ and $\cos \theta_{k_t^{(i)}}^{(i)} \rightarrow 0$, the summands $\alpha_{k_t^{(i)}} / \cos^2 \theta_{k_t^{(i)}}^{(i)}$ tend to $+\infty$, hence

$$\sum_{t=1}^{\ell} \frac{\alpha_{k_t^{(i)}}}{\cos^2 \theta_{k_t^{(i)}}^{(i)}} = +\infty \quad (\ell \rightarrow \infty),$$

contradicting boundedness of the left-hand side of (5.10), and therefore (II) cannot occur.

From the failure of (I) and (II) we conclude that, for this block i , there exist $\alpha_{\min}^{(i)} > 0$, $\delta_i > 0$, and an infinite subsequence of active- i indices $\{k_\ell^{(i)}\} \subset \mathcal{K}_i$ such that

$$\alpha_{k_\ell^{(i)}} \geq \alpha_{\min}^{(i)} \quad \text{and} \quad \cos \theta_{k_\ell^{(i)}}^{(i)} \geq \delta_i \quad \text{for all } \ell,$$

and moreover $N_i(K) \rightarrow \infty$ almost surely. Because there are finitely many blocks, intersecting the corresponding probability-one events over $i = 1, \dots, n_B$ yields an event Ω with $\mathbb{P}(\Omega) = 1$ on which the conclusion above holds for *every* block.

For Any Block. Recall p_k denote the embedded search direction ($s_k = \alpha_k p_k$ with p_k supported on i_k). By the Armijo–Wolfe line search, we know ∇f is M-Lipschitz on \mathcal{D} , hence [4]:

$$\sum_{k=0}^{\infty} \frac{(\nabla f(x_k)^\top p_k)^2}{\|p_k\|^2} < \infty. \quad (5.12)$$

Since

$$\nabla f(x_k)^\top p_k = \nabla_{i_k} f(x_k)^\top p_k^{(i_k)} = -\|\nabla_{i_k} f(x_k)\| \|p_k^{(i_k)}\| \cos \theta_k^{(i_k)}$$

and

$$\|p_k\| = \|p_k^{(i_k)}\|,$$

(5.12) becomes

$$\sum_{k=0}^{\infty} \|\nabla_{i_k} f(x_k)\|^2 \cos^2 \theta_k^{(i_k)} < \infty.$$

Along the subsequence $\{k_\ell\}$ we have $\cos \theta_{k_\ell}^{(i)} \geq \delta$, hence

$$\sum_{\ell} \|\nabla_i f(x_{k_\ell})\|^2 \leq \frac{1}{\delta^2} \sum_{\ell} \|\nabla_i f(x_{k_\ell})\|^2 \cos^2 \theta_{k_\ell}^{(i)} < \infty,$$

so $\|\nabla_i f(x_{k_\ell})\| \rightarrow 0$.

Armijo implies $f(x_{k+1}) \leq f(x_k)$, so $\{f(x_k)\}$ is nonincreasing and bounded below by $f(x^*)$. Any accumulation point \bar{x} of $\{x_k\}$ satisfies $\nabla_i f(\bar{x}) = 0$ for the block i above. If some block j admits a similar subsequence, we get $\nabla_j f(\bar{x}) = 0$ as well; iterating across the finitely many blocks and extracting a diagonal subsequence yields $\nabla f(\bar{x}) = 0$. By strong convexity, $\bar{x} = x^*$. Therefore $f(x_k) \downarrow f(x^*)$ and $x_k \rightarrow x^*$ a.s.

Full Generalization By contradiction, assume that there exists $\varepsilon > 0$ with

$$\|\nabla f(x_k)\| \geq \varepsilon \quad \text{for all } k. \quad (5.13)$$

define the maximum block

$$J_k \in \arg \max \{ \|\nabla_1 f(x_k)\|, \|\nabla_2 f(x_k)\|, \dots, \|\nabla_{n_B} f(x_k)\| \},$$

so that

$$\|\nabla_{J_k} f(x_k)\| \geq \frac{\varepsilon}{\sqrt{n_B}} \quad \text{for all } k. \quad (5.14)$$

Recall $i_k \in \{1, 2, \dots, n_B\}$ is the active block at iteration k , drawn i.i.d. uniformly and independently of the filtration $\mathcal{F}_k := \sigma(x_0, B_0, \dots, x_k, B_k)$, and from (4.8) that the BFGS update on the active block satisfies

$$\text{tr}(B_{k+1}^{(i_k, i_k)}) \leq \text{tr}(B_k^{(i_k, i_k)}) + M - \frac{\alpha_k}{c'_2 \cos^2(\theta_k^{(i_k)})}.$$

We will construct an infinite index set on which:

- (i) α_k is bounded away from zero;
- (ii) i_k equals the maximum block J_k ;
- (iii) $\cos \theta_k^{(i_k)}$ is bounded away from 0.

(i) Large α_k occur i.o. By Lemma 5.1 (product bound), there exists $c_4 \in (0, 1)$ and K_0 such that $\prod_{j=1}^K \alpha_j \geq c_4^K$ for all $K \geq K_0$. In particular, for any fixed $\underline{\alpha} \in (0, c_4)$, the we obtain an index set

$$\mathcal{L}(\underline{\alpha}) := \{k \geq 0 : \alpha_k \geq \underline{\alpha}\}.$$

Suppose, to the contrary, that $\mathcal{L}(\underline{\alpha})$ is finite. Then there exists $K_1 \in \mathbb{N}$ such that

$$\alpha_k < \underline{\alpha} \quad \text{for all } k \geq K_1.$$

Define the finite product

$$C_0 := \prod_{j=1}^{K_1-1} \alpha_j > 0.$$

For any $K \geq K_1$, we can split the product as

$$\prod_{j=1}^K \alpha_j = \left(\prod_{j=1}^{K_1-1} \alpha_j \right) \left(\prod_{j=K_1}^K \alpha_j \right) \leq C_0 \underline{\alpha}^{K-K_1+1},$$

because each factor in the second product is strictly less than $\underline{\alpha}$. On the other hand, by Lemma 5.1 we have

$$\prod_{j=1}^K \alpha_j \geq c_4^K \quad \text{for all } K \geq N_0.$$

Combining the two bounds, for all $K \geq \max\{K_1, N_0\}$ we obtain

$$c_4^K \leq C_0 \underline{\alpha}^{K-K_1+1}.$$

Taking logarithms and dividing by K yields

$$\log c_4 \leq \frac{\log C_0}{K} + \left(1 - \frac{K_1 - 1}{K}\right) \log \underline{\alpha}.$$

Letting $K \rightarrow \infty$ gives

$$\log c_4 \leq \log \underline{\alpha}.$$

Since $0 < \underline{\alpha} < c_4 < 1$, we have $\log \underline{\alpha} < \log c_4$, which contradicts the last inequality. Hence our assumption was false, and $\mathcal{L}(\underline{\alpha})$ must be infinite.

(ii) large-step J_k occurs i.o. Fix $\underline{\alpha} \in (0, c_4)$ and enumerate $\mathcal{L}(\underline{\alpha})$ as $k_1 < k_2 < \dots$. For each m , the random variable i_{k_m} is uniform on $\{1, 2, \dots, n_B\}$ and independent of all other i_k and of J_{k_m} (which is a function of x_{k_m} and hence determined by the past). Therefore

$$\mathbb{P}(E_m \mid J_{k_m}) = \mathbb{P}(i_{k_m} = J_{k_m} \mid J_{k_m}) = \frac{1}{n_B},$$

so $\mathbb{P}(E_m) = \frac{1}{n_B}$ for all m . Since the block choices $\{i_k\}$ are independent across k , the events $\{E_m\}_{m \geq 1}$ are independent. Thus

$$\sum_{m=1}^{\infty} \mathbb{P}(E_m) = \sum_{m=1}^{\infty} \frac{1}{n_B} = \infty,$$

and the Borel–Cantelli lemma implies that E_m occurs infinitely often almost surely. Equivalently, the index set

$$\mathcal{K}_{\alpha}^{\max} := \{k \geq 0 : \alpha_k \geq \underline{\alpha}, i_k = J_k\}$$

is infinite almost surely.

(iii) $\cos \theta_k^{(i_k)}$ is bounded away from zero on $\mathcal{K}_{\alpha}^{\max}$ Fix $\underline{\alpha} \in (0, c_4)$ and let $\mathcal{K}_{\alpha}^{\max}$ be as in (ii). Pick any $\tau > M$ and define

$$\Omega_k := \frac{\alpha_k}{c'_2 \cos^2 \theta_k^{(i_k)}}, \quad k \geq 0.$$

Split the large-step max-block index set into

$$G := \{k \in \mathcal{K}_{\alpha}^{\max} : \Omega_k \leq \tau\}, \quad B := \{k \in \mathcal{K}_{\alpha}^{\max} : \Omega_k > \tau\}.$$

Suppose, for contradiction, that G is finite. Then B is infinite. For each $k \in B$, we have $\Omega_k > \tau$ and $\alpha_k \geq \underline{\alpha}$, so by the trace recursion

$$\text{tr}(B_{k+1}^{(i_k, i_k)}) \leq \text{tr}(B_k^{(i_k, i_k)}) + M - \Omega_k \leq \text{tr}(B_k^{(i_k, i_k)}) + M - \tau.$$

Since $\tau > M$, set $\eta_0 := \tau - M > 0$ and rewrite this as

$$\text{tr}(B_{k+1}^{(i_k, i_k)}) \leq \text{tr}(B_k^{(i_k, i_k)}) - \eta_0 \quad \text{for all } k \in B.$$

Summing over the infinitely many $k \in B$ forces $\text{tr}(B_k^{(i_k, i_k)}) \rightarrow -\infty$ along the subsequence $k \in B$, contradicting positive definiteness of $B_k^{(i_k, i_k)}$. Hence B must be finite and G infinite. For any $k \in G$, we have $\alpha_k \geq \underline{\alpha}$ and $\Omega_k \leq \tau$, hence

$$\cos^2 \theta_k^{(i_k)} = \frac{\alpha_k}{c'_2 \Omega_k} \geq \frac{\underline{\alpha}}{c'_2 \tau} =: \delta > 0.$$

Let $\mathcal{K}^* := G$. Then $\mathcal{K}^* \subseteq \mathcal{K}_{\alpha}^{\max}$ is infinite and satisfies the desired cosine lower bound.

Zoutendijk contradiction. By our arguments in (i), (ii) and (iii), there exists an infinite index set \mathcal{K}^* and constants $\underline{\alpha} > 0$, $\delta > 0$ such that

$$\alpha_k \geq \underline{\alpha}, \quad i_k = J_k, \quad \cos^2 \theta_k^{(i_k)} \geq \delta \quad \text{for all } k \in \mathcal{K}^*.$$

On \mathcal{K}^* , we also have by (5.14)

$$\|\nabla_{i_k} f(x_k)\| = \|\nabla_{J_k} f(x_k)\| \geq \frac{\varepsilon}{\sqrt{n_B}}.$$

Recall the Zoutendijk inequality (5.12):

$$\sum_{k=0}^{\infty} \frac{(\nabla f(x_k)^\top p_k)^2}{\|p_k\|^2} = \sum_{k=0}^{\infty} \|\nabla_{i_k} f(x_k)\|^2 \cos^2 \theta_k^{(i_k)} < \infty.$$

Along the infinite index set \mathcal{K}^* , each summand satisfies

$$\|\nabla_{i_k} f(x_k)\|^2 \cos^2 \theta_k^{(i_k)} \geq \left(\frac{\varepsilon}{\sqrt{n_B}}\right)^2 \delta =: c_0 > 0,$$

so the subseries $\sum_{k \in \mathcal{K}^*} \|\nabla_{i_k} f(x_k)\|^2 \cos^2 \theta_k^{(i_k)}$ diverges, contradicting (5.12). Therefore the assumption (5.13) is false, and we conclude

$$\liminf_{k \rightarrow \infty} \|\nabla f(x_k)\| = 0.$$

□

5.4 Global Convergence Under Relaxed Assumptions

In this subsection, we prove global convergence for the block-coordinate BFGS algorithm under a relaxed assumption on the smooth part f . In particular, we assume:

Assumption 5.1. *The function f is twice continuously differentiable, convex, and bounded below. Moreover, its Hessian is uniformly bounded on the level set \mathcal{D} , i.e., there exists a constant $M > 0$ such that*

$$\|\nabla^2 f(x)\| \leq M, \quad \forall x \in \mathcal{D}.$$

Note that Assumption 5.1 is strictly weaker than strong convexity. However, the convexity of f together with the Wolfe line search guarantees sufficient descent. We now show the global convergence under the new assumption.

Theorem 5.2 (Relaxed global convergence for block BFGS). *Assume 5.1 and Armijo–Wolfe. Then*

$$\liminf_{k \rightarrow \infty} \|\nabla f(x_k)\| = 0.$$

If, in addition, $\{x_k\}$ is bounded, there exists a subsequence $\{x_{k_\ell}\}$ with $x_{k_\ell} \rightarrow x_$, where x_* is a global minimizer, $\nabla f(x_*) = 0$, and $f(x_k) \downarrow f_*$.*

Proof. Assume that there exist $\gamma > 0$ and K_0 with $\|g_k\| \geq \gamma$ for all $k \geq K_0$. Let the embedded step be $s_k := u_{i_k}(s_k^{(i_k)})$. From (2.11), $g_k^\top s_k \rightarrow 0$.

Using (3.5), $B_k^{(i_k, i_k)} s_k^{(i_k)} = -\alpha_k g_k^{(i_k)}$. Embedding gives

$$B_k s_k = -\alpha_k u_{i_k}(g_k^{(i_k)}) \quad \text{and} \quad s_k^\top B_k s_k = -\alpha_k g_k^\top s_k.$$

Define

$$\tilde{\eta}_k := -\frac{\|g_k\|^2}{-g_k^\top s_k} (< 0), \quad \lambda_k := \alpha_k,$$

so the negative term in (4.3) equals $-\|B_k s_k\|^2 / (s_k^\top B_k s_k) = \alpha_k \tilde{\eta}_k = \lambda_k \tilde{\eta}_k$. Since $\|g_k\| \geq \gamma$ and $g_k^\top s_k \rightarrow 0$, we have $\tilde{\eta}_k \rightarrow -\infty$.

Summing (4.3) from K_0 to K and using (4.4),

$$\text{tr}(B_{K+1}) \leq \text{tr}(B_{K_0}) + M(K - K_0 + 1) + \sum_{k=K_0}^K \lambda_k \tilde{\eta}_k. \quad (5.15)$$

By Lemma 5.1, there exist $c_4 \in (0, 1)$, N_0 such that for any $\varepsilon \in (0, c_4)$ there is a subsequence $\{k_\ell\}$ with $\lambda_{k_\ell} = \alpha_{k_\ell} \geq \varepsilon$. Thin it so that $\tilde{\eta}_{k_\ell} \leq -\ell$. Then (5.15) gives

$$\text{tr}(B_{K+1}) \leq \text{tr}(B_{K_0}) + M(K - K_0 + 1) - \varepsilon \sum_{\ell: k_\ell \leq K} \ell \longrightarrow -\infty,$$

contradicting $B_{K+1} \succ 0$. Hence $\liminf_k \|g_k\| = 0$. □

If $\{x_k\}$ is bounded, Armijo implies $f(x_k)$ is monotone and bounded below, so $f(x_k) \rightarrow f_\star$. Choose k_ℓ with $\|g_{k_\ell}\| \rightarrow 0$ and pass to a further subsequence (still k_ℓ) with $x_{k_\ell} \rightarrow x_\star$. Continuity gives $\nabla f(x_\star) = 0$; convexity implies x_\star is a global minimizer. Thus $f(x_k) \downarrow f_\star$.

6 Global Convergence for Non-smooth Functions

6.1 Smooth Neighborhood for Non-smoothness

Having established global convergence for the smooth case of the block BFGS algorithm under the assumption that the smooth part f is twice continuously differentiable on the entire level set \mathcal{D} , we now extend the analysis to the non-smooth setting. In this section we analyze a block BFGS algorithm applied to a composite convex objective. Our goal is to prove that, under suitable assumptions, the sequence generated by the algorithm converges in function value to the global minimum. We rely on techniques outlined in [5], which adapts Powell's convergence results to handle non-smoothness.

Recall the composite function from (1.1), where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is smooth and strongly convex and $g : \mathbb{R}^n \rightarrow \mathbb{R}$ is convex. And then, we assume the level set defined in (2.2) to be compact. Then, the following assumptions are made based on Assumption 2.1:

Assumption 6.1 (Non-smooth Component). *The function $g : \mathbb{R}^n \rightarrow \mathbb{R}$ is convex. Moreover, there exists an open set $V \subset U$ (with U as in Assumption 2.1) such that:*

1. *All accumulation points of the iterates $\{x_k\} \subset \mathcal{D}$ lie in V .*
2. *The composite function $F(x) = f(x) + g(x)$ is twice continuously differentiable on V .*

$$3. \inf_{x \in V} F(x) = \min F.$$

Assumption 6.2. *The block-coordinate BFGS algorithm uses an Armijo–Wolfe line search on F and i.i.d. uniform block sampling: $i_k \in \{1, \dots, n_B\}$ are i.i.d. uniform and independent of the past $\mathcal{F}_k = \sigma(x_0, \dots, x_k, B_0, \dots, B_k)$.*

Theorem 6.1 (Global convergence in the non-smooth strongly convex case). *Under Assumptions 6.1–6.2, the block BFGS iterates satisfy, almost surely,*

$$F(x_k) \downarrow F_\star := \min F \quad \text{and} \quad x_k \rightarrow x_\star,$$

where x_\star is the unique minimizer of F .

Proof. Construct smooth neighborhood. By compactness of \mathcal{D} and Assumption 6.1(i), the set of accumulation points of $\{x_k\}$ is nonempty, compact, and contained in the open set V . Hence there exists a compact set Ω with $\text{accum}(\{x_k\}) \subset \Omega \subset \text{int}(V)$. Therefore, there is K_0 such that $x_k \in \Omega$ for all $k \geq K_0$.

Following [5], shrink V slightly and intersect with a level set to obtain a compact set on which F is C^2 and strongly convex; then apply the smooth convex extension tool to obtain a function $F_e \in C^2(U_e)$ on a convex open neighborhood $U_e \supset \text{conv}(\Omega)$ such that

$$F_e(x) = F(x), \quad \nabla F_e(x) = \nabla F(x) \quad \forall x \in \Omega,$$

and $\nabla^2 F_e(x)$ is positive definite and continuous on U_e . Because $\text{conv}(\Omega)$ is compact, there exist constants $0 < \hat{m} \leq \hat{M} < \infty$ with

$$\hat{m}I \preceq \nabla^2 F_e(x) \preceq \hat{M}I \quad \forall x \in \text{conv}(\Omega).$$

Moreover, $\min F_e = \min F$ and every minimizer lies in $V \subset U_e$.

Fix $k \geq K_0$. Since $x_k, x_{k+1} \in \Omega \subset U_e$ and $F_e = F$, $\nabla F_e = \nabla F$ on Ω , the accepted stepsize α_k satisfies the Armijo–Wolfe conditions for F_e if and only if it does for F . Hence the realized pair

$$s_k = u_{i_k}(s_k^{(i_k)}), \quad y_k = \nabla F(x_{k+1}) - \nabla F(x_k) = \nabla F_e(x_{k+1}) - \nabla F_e(x_k)$$

is the same for F and F_e . Consequently, the block BFGS update computed from (s_k, y_k) is identical under F and F_e . Therefore, $\{x_k\}_{k \geq K_0}$ is *also* a block-BFGS sequence for the smooth, strongly convex function F_e with the same i.i.d. block sampling.

Since F_e is C^2 with $\hat{m}I \preceq \nabla^2 F_e \preceq \hat{M}I$ on $\text{conv}(\Omega)$ and the line segments $[x_k, x_{k+1}] \subset \text{conv}(\Omega)$, the hypotheses of our smooth strong-convexity result (Theorem 5.1) hold for F_e on the tail under the same i.i.d. sampling. Hence, almost surely,

$$F_e(x_k) \downarrow \min F_e \quad \text{and} \quad x_k \rightarrow x_e^\star \quad (k \rightarrow \infty).$$

Because $F_e = F$ on Ω and $\min F_e = \min F$, we obtain $F(x_k) = F_e(x_k) \downarrow \min F =: F_\star$ and $x_k \rightarrow x_e^\star =: x_\star$. Uniqueness follows from strong convexity of F . \square

6.2 A Weaker Smoothness Condition

In many practical problems—especially when the non-smooth component is prominent—the full smoothness on a neighborhood of the whole level set may fail. Following Section 4 in [5], we assume only that on each fixed “value band” above the minimum there exists a smooth convex extension that agrees with the objective. This allows us to reuse the smooth block-BFGS argument on the realized sequence without altering the steps taken by the algorithm.

Assumption 6.3 (Local Smooth Extension). *Let $F : \mathbb{R}^n \rightarrow \mathbb{R}$ be convex with compact level set $L_0 := \{x : F(x) \leq F(x_0)\}$ and $F_\star := \min F$. For every $\delta > 0$ there exist*

1. *a convex open neighborhood $U_\delta \supset L_0$, and*
2. *a function $F_\delta \in C^2(U_\delta)$ that is convex on U_δ and satisfies*

$$F_\delta(x) = F(x) \quad \text{whenever } F(x_0) \geq F(x) \geq F_\star + \delta.$$

Theorem 6.2 (Global Convergence Under Local Extension). *Assume 6.3 and the Armijo–Wolfe line search with i.i.d. uniform block sampling (as in Section 5). Then, almost surely,*

$$F(x_k) \downarrow F_\star.$$

Proof. Suppose by contradiction that $F(x_k) \not\downarrow F_\star$. Then there exists $\delta > 0$ such that

$$F(x_k) \geq F_\star + 2\delta \quad \text{for all } k. \tag{6.1}$$

By Assumption 6.3, pick U_δ and a C^2 convex F_δ on U_δ with $F_\delta(x) = F(x)$ on the band $\{x : F_\star + \delta \leq F(x) \leq F(x_0)\}$. Since L_0 is compact and U_δ is open with $L_0 \subset U_\delta$, the set L_0 is contained in a compact $K_\delta \Subset U_\delta$. By continuity of $\nabla^2 F_\delta$ on K_δ , there exists $M_\delta < \infty$ with $\|\nabla^2 F_\delta(x)\| \leq M_\delta$ for all $x \in K_\delta$.

Claim: Along the realized sequence, the block steps, Wolfe acceptance, and BFGS updates are identical for F and F_δ . Indeed, by (6.1) each x_k lies in the band where $F_\delta = F$; hence $F_\delta(x_k) = F(x_k)$ and $\nabla F_\delta(x_k) = \nabla F(x_k)$ for all k . Therefore the accepted stepsize α_k satisfies Armijo–Wolfe for F_δ iff it does for F , and with $s_k := u_{i_k}(s_k^{(i_k)})$ we have

$$y_k = \nabla F(x_{k+1}) - \nabla F(x_k) = \nabla F_\delta(x_{k+1}) - \nabla F_\delta(x_k),$$

so the block BFGS update computed from (s_k, y_k) is the same for both problems.

Consequently, the realized sequence $\{x_k\}$ is also a block-BFGS sequence for the smooth convex function F_δ on K_δ with bounded Hessian. The hypotheses of Theorem 5.2, our relaxed smooth result, therefore hold for F_δ : the level set $\{x : F_\delta(x) \leq F_\delta(x_0)\}$ is contained in K_δ (hence compact), $\|\nabla^2 F_\delta\| \leq M_\delta$ on it, and Armijo–Wolfe holds along the realized steps. Thus, almost surely,

$$F_\delta(x_k) \downarrow \min F_\delta.$$

Finally, since $F_\delta = F$ on the band, $F_\delta(x_k) = F(x_k)$ for all k by (6.1). Moreover, by continuity of F and convexity, there exists y with $F(y) = F_\star + \delta$, whence $F_\delta(y) = F(y)$ and so $\min F_\delta \leq F_\delta(y) = F_\star + \delta$. Therefore $F(x_k) = F_\delta(x_k) \downarrow \min F_\delta \leq F_\star + \delta$, contradicting (6.1).

Hence $F(x_k) \downarrow F_\star$ almost surely. □

If F is strongly convex, uniqueness of the minimizer implies $x_k \rightarrow x_\star$.

Corollary 6.1 (Block BFGS Analogue of Corollary 3.4 from [5]). *Suppose F is semialgebraic, strongly convex on an open semialgebraic convex set $U \supset L_0$, and has compact level sets. Let $V \subset U$ be the (open) locus where F is C^2 . If x_0 is drawn from a distribution absolutely continuous w.r.t. Lebesgue measure on U , and the algorithm uses Armijo–Wolfe with i.i.d. uniform block sampling, then almost surely either*

- (i) $F(x_k) \rightarrow \min F$ (success), or
- (ii) a subsequence $\{x_{k_j}\}$ converges to a point where F is not smooth and not minimized (failure).

In semialgebraic geometry V has full measure, so with probability one the realized iterates and line segments avoid the nonsmooth set; then Theorem 6.1 (or Theorem 6.2 when the extension exists) yields (i). If (i) fails, any limit point must be outside V and nonoptimal, giving (ii). This is the block-coordinate analogue of [5, Cor. 3.4].

6.3 Examples

In this section, we illustrate the theory on two concrete convex objectives: a 2D ℓ_1 model, and the Overlapping Group LASSO. Throughout, the block index i_k is sampled i.i.d. uniformly, and the line search is Armijo–Wolfe as specified earlier.

Corollary 6.2 (Two-block Block BFGS on 2D ℓ_1). *Let $F(u, v) = |u| + |v|$ on \mathbb{R}^2 , with two coordinate blocks $I_1 = \{u\}$ and $I_2 = \{v\}$. Assume i.i.d. uniform block sampling and an Armijo–Wolfe line search applied to F . Then, almost surely,*

$$F(x_k) \downarrow 0 \quad \text{and} \quad x_k = (u_k, v_k) \rightarrow (0, 0).$$

Proof. The level set $\{F \leq F(x_0)\}$ is compact and convex. For every $\delta > 0$, we can construct a convex C^2 function F_δ that agrees with F on the “value band” $\{(u, v) : \delta \leq |u| + |v| \leq F(x_0)\}$ and is C^2 on a convex open neighborhood of the level set (Assumption 6.3). Hence the hypotheses of Theorem 6.2 hold, and we conclude $F(x_k) \downarrow \min F = 0$ almost surely.

Because the level set $\{F \leq F(x_0)\}$ is compact and $\arg \min F = \{(0, 0)\}$ is a singleton, every accumulation point of $\{x_k\}$ must be $(0, 0)$; hence the whole sequence converges: $x_k \rightarrow (0, 0)$. \square

Corollary 6.3 (Overlapping Group LASSO). *Consider*

$$F(x) = \frac{1}{2} \|Ax - b\|^2 + \lambda \sum_{g \in \mathcal{G}} w_g \|x_{(g)}\|,$$

with $A \in \mathbb{R}^{m \times n}$, $\lambda > 0$, weights $w_g > 0$, and overlapping groups \mathcal{G} . Assume:

1. the level set $\mathcal{D} := \{x : F(x) \leq F(x_0)\}$ is compact;
2. blocks are sampled i.i.d. uniformly (independently of the past);
3. the line search is the composite Armijo–Wolfe (Armijo on F , curvature on $f(x) = \frac{1}{2} \|Ax - b\|^2$);
4. F satisfies Assumption 6.3.

Then, almost surely,

$$F(x_k) \downarrow F_\star := \min F,$$

and, moreover, $\mathbb{E}[F(x_k)] \downarrow F_\star$.

Proof. By Assumption 6.3, for each $\delta > 0$ there exist an open convex $U_\delta \supset \mathcal{D}$ and a convex C^2 function F_δ on U_δ such that $F_\delta(x) = F(x)$ whenever $F_\star + \delta \leq F(x) \leq F(x_0)$. Applying Theorem 6.2 to the realized sequence, we conclude $F(x_k) \downarrow F_\star$ almost surely.

Along each path the sequence $\{F(x_k)\}$ is nonincreasing by Armijo and bounded below by F_\star , and we just proved $F(x_k) \rightarrow F_\star$ a.s. Since $0 \leq F(x_k) - F_\star \leq F(x_0) - F_\star$ for all k , the bounded convergence theorem yields $\mathbb{E}[F(x_k)] \rightarrow F_\star$. \square

7 Experiments with Leastsquares + Overlapping GroupLASSO

We consider the following composite optimization problem:

$$\min_{x \in \mathbb{R}^n} F(x) = \underbrace{\frac{1}{2} \|Ax - b\|^2}_{f(x)} + \underbrace{\left(\lambda_1 \|x\|_1 + \lambda_2 \sum_{i=1}^m w_i \|x_{G_i}\|_2 \right)}_{g(x)},$$

where

- $A \in \mathbb{R}^{p \times n}$ and $b \in \mathbb{R}^n$ define the least-squares loss term,
- $G = \{G_1, G_2, \dots, G_m\}$ is a collection of index subsets corresponding to *overlapping groups* of variables in x ,
- $\|x_{G_i}\|_2$ denotes the Euclidean norm of the subvector of x indexed by G_i ,
- $\lambda_1, \lambda_2 > 0$ are regularization parameters, and $w_i > 0$ is a weight for group G_i .

This problem consists of a smooth component $f(x)$ and a non-smooth component $g(x)$. Although each group norm is non-smooth at zero, it is differentiable when $x_{G_i} \neq 0$. We solve this problem by treating the smooth part with a quasi-Newton strategy while incorporating subgradient information for the non-smooth component. The algorithm is terminated when either (i) the cumulative change in x across all blocks falls below a threshold (set to $\varepsilon = 10^{-6}$) or (ii) a maximum number of iterations is reached.

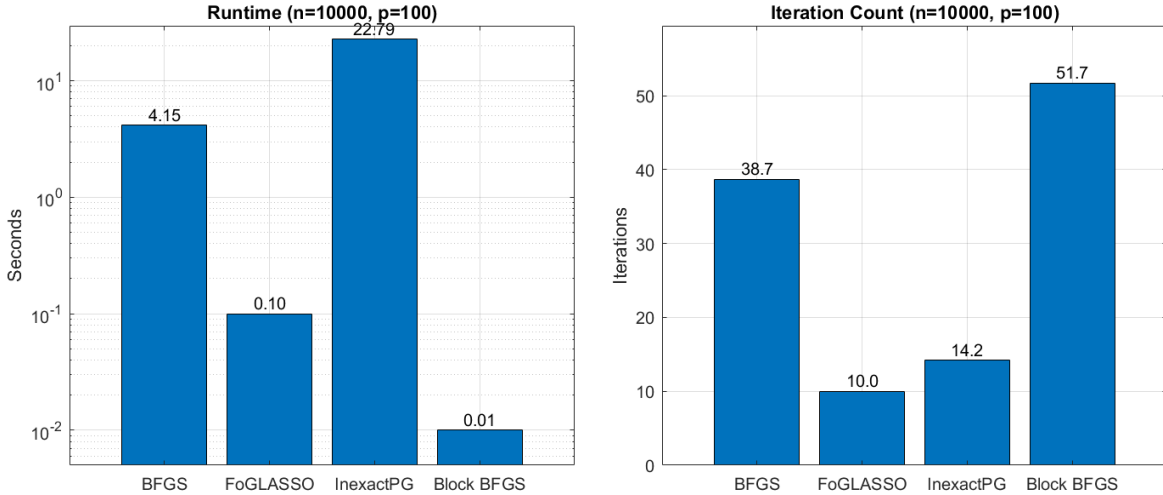


Figure 4: Scaling comparison of BFGS, FoGLASSO, InexactPG, and Block BFGS on an overlapping GroupLasso least-squares instance ($n = 10,000$, $p = 100$).

Recall the comparison from Figure 3, we now add the Block BFGS to this problem as shown in Figure 4. Its runtime is about 0.01 s, compared with 4.15 s for full BFGS and 0.10 s for FoGLASSO. Thus, Block BFGS is roughly 415 \times faster than full BFGS and 10 \times faster than FoGLASSO, highlighting how block-diagonal curvature updates sharply reduce per-iteration cost while maintaining strong progress per step. In the following sections, we introduce the formal computational experiments.

7.1 Simulated Experiments

Dataset Generation and Experimental Protocol: For our simulated experiments, datasets are generated as follows: given the number of samples n , the number of features p , and the number of groups m , we partition the n features into overlapping groups. A sparse ground truth vector x^* is constructed by randomly activating a subset (typically 20%) of groups and assigning them random nonzero coefficients. The design matrix $A \in \mathbb{R}^{p \times n}$ is drawn from a multivariate Gaussian distribution. We compute the noise-free response Ax^* , and then add Gaussian noise to obtain the observed vector b . In these simulated experiments, we use a least-squares loss combined with GroupLasso regularization. (For our experiments on LIBSVM datasets, we employ logistic regression, so the experimental setup is different.)

To ensure robust statistical evaluation, each algorithm is run 10 times using seeds 1 to 10. This repetition allows us to compute both the average runtime per iteration and the overall average runtime across 10 independent runs, thereby reducing the impact of random initialization and noise variability.

We vary the dimensions of A and b to explore different regimes, including settings where the number of samples is less than the number of features ($p > n$). One notable observation is that the *Block BFGS* method sometimes fails after several hundred iterations when the dataset is underdetermined. This failure can be attributed to the fact that, in such cases, the Hessian matrix $A^\top A$ becomes rank deficient or nearly singular. Since the Block BFGS update relies on curvature information extracted from this Hessian, an insufficient sample size causes the approximated Hessian to lose its positive definiteness. In particular, small eigenvalues lead to numerical instabilities in the quasi-Newton update, eventually destabilizing the algorithm. In contrast, first-order methods such as *InexactPG* and *FoGLASSO* [2] are less dependent on curvature information and tend to be more robust in underdetermined settings, though they may exhibit slower convergence.

Results of Simulated Experiments Tables 1 and 2 summarize the experimental results in terms of the average runtime (in seconds) and runtime per iteration, computed over 10 runs:

Table 1: Solution Accuracy for Simulated Datasets

n	p	Block BFGS		BFGS		FoGLASSO		InexactPG	
		Max Error	Mean Error	Max Error	Mean Error	Max Error	Mean Error	Max Error	Mean Error
10	10	0.308	0.398	0.308	0.398	0.308	0.399	0.308	0.398
10	100	0.607	0.745	0.607	0.745	0.357	0.437	0.357	0.437
10	1000	0.111	0.237	0.120	0.243	0.114	0.235	0.114	0.235
100	10	1.009	1.009	1.008	1.098	1.019	1.114	1.019	1.114
100	100	1.241	1.824	1.242	1.825	0.984	1.064	0.984	1.064
100	1000	0.855	0.895	0.855	0.895	0.860	0.896	0.860	0.896
100	10000	0.111	0.237	0.685	0.708	2.403	2.729	2.403	2.729
1000	10	0.198	0.198	3.150	3.289	3.159	3.304	3.159	3.304
1000	100	1.073	1.073	3.017	3.112	3.175	3.276	3.174	3.180
1000	1000	4.160	4.160	2.544	2.969	3.129	3.235	3.129	3.233
1000	10000	0.325	0.331	0.743	0.754	0.316	0.321	0.315	0.316
10000	10	0.005	0.005	0.003	0.005	0.003	0.005	0.003	0.005
10000	100	0.092	0.092	0.011	0.013	0.010	0.012	0.010	0.012
10000	1000	0.031	0.031	0.033	0.034	0.033	0.034	0.033	0.034
10000	10000	0.399	0.440	0.396	0.435	0.444	0.501	0.511	0.534

Table 2: Runtime and Iteration Count for Simulated Datasets

n	p	m	Block BFGS			BFGS		FoGLASSO		InexactPG	
			# Blocks	RunTime	# Iters	RunTime	# Iters	RunTime	# Iters	RunTime	# Iters
10	10	4	2	0.02	31.1	0.14	41.8	0.44	7.6	1.20	10.2
10	100	10	3	0.08	155.5	4.17	83.1	19.14	113.4	131.11	11.9
10	1000	150	10	8.89	843.3	403.63	933.8	1960.47	3739.6	4451.53	15.3
100	10	4	3	0.01	21.8	0.16	27.9	0.33	6.4	2.23	10.1
100	100	10	3	0.09	278.6	18.01	86.4	0.14	86.6	676.71	50.1
100	1000	150	4	55.02	4662.2	361.18	816.4	108880.11	21940.4	96401.15	77
100	10000	1000	10	46659.11	47584.7	639785.56	2475.17	320305.87	39486.1	682905.56	182.4
1000	10	4	2	0.01	13.3	0.31	16.3	0.05	8.0	2.99	15.6
1000	100	10	6	0.04	30.3	5.95	56.1	0.07	16.7	77.88	22.3
1000	1000	150	8	11.93	1686.8	84.74	270.4	6.90	300.3	48.91	13.9
1000	10000	1000	20	20423.46	47213.6	243311.12	783.4	23683.81	40859.1	491808.14	140.0
10000	10	4	2	0.02	9.7	0.07	14.8	0.04	7.1	1.86	12.7
10000	100	10	2	0.01	51.7	4.15	38.7	0.10	10.0	22.79	14.2
10000	1000	150	12	6.55	100.6	40.43	117.5	14.53	23.2	28540.04	40.9
10000	10000	1000	20	8449.07	7424.4	290364.52	1205.6	12989.23	16027.2	221076.19	112.7

Solution Accuracy. As seen in Table 1, Block BFGS delivers accuracy competitive with full BFGS and exceeds it in several regimes. It matches or improves upon BFGS on 13 of 15 instances and is numerically indistinguishable on most small-to-medium settings; on higher-dimension problems it can be substantially tighter—for example, at $(n=100, p=10,000)$ the mean/max errors are 0.111/0.237 for Block BFGS versus 0.685/0.708 for BFGS. Relative to the proximal-gradient baselines, Block BFGS is consistently competitive and often markedly more accurate at high dimensionality—for instance, at $(n=100, p=10,000)$ FoGLASSO and InexactPG yield mean/max error at 2.403/2.729 versus 0.111/0.237 for Block BFGS. Taken together with its runtime advantages, these

results indicate that Block BFGS achieves reliable, BFGS-level (or better) accuracy while avoiding the variability of the proximal methods, reinforcing its suitability for large-scale structured optimization.

Runtime and Iteration Efficiency. According to Table 2, Block-BFGS attains the shortest runtime in 14/15 cases against FoGLASSO, InexactPG, and BFGS; the only exception is $(n=1000, p=1000)$, where FoGLASSO is faster (6.90 s vs. 11.93 s, i.e., Block-BFGS is $1.73\times$ slower, -72.8%). Relative to BFGS, Block-BFGS is faster in all 15/15 settings and typically by one-two orders of magnitude—for example, at $(n=10, p=1000)$ the runtime drops from 403.6 s to 8.89 s ($\sim 45\times$), and at $(n=10,000, p=10,000)$ from 290,365 s to 8,449 s ($\sim 34\times$). A direct comparison with BFGS clarifies why Block-BFGS remains faster even when it takes more steps: each Block-BFGS iteration updates only a small coordinate block and reuses within-block curvature, thereby avoiding the full $p \times p$ quasi-Newton update and dense search-direction computation required by BFGS. For instance, at $(n=100, p=100)$ Block-BFGS uses 278.6 block steps versus 86.4 BFGS iterations yet runs in 0.092 s versus 18.01 s. Among the two proximal-gradient methods, FoGLASSO is substantially faster (and in some instances faster than BFGS), but it offers no runtime advantage over Block-BFGS—especially at high dimensionality. For example, at $(n=100, p=10,000)$ FoGLASSO takes 320,306 s versus 46,659 s for Block-BFGS, a difference of $\approx 2.74 \times 10^5$ seconds (~ 76 hours).

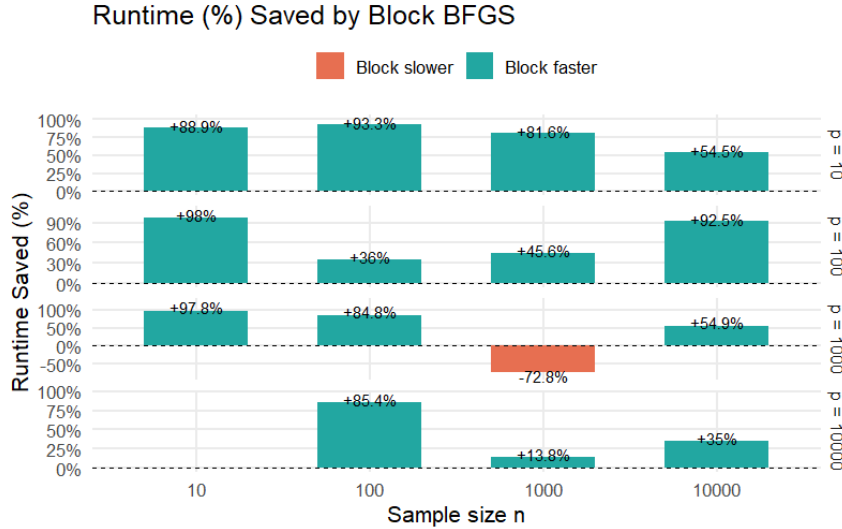


Figure 5: Savings of Block-BFGS relative to the best competing method (BFGS, FoGLASSO, or InexactPG) on each instance. Bars show $\% \text{ savings} = 100(1 - t_{\text{Block}}/t_{\text{BestOther}})$. Panels are grouped by feature count p ; within each panel, bars are indexed by sample size n . Positive values indicate Block-BFGS is faster; the single negative bar corresponds to $(n=1000, p=1000)$.

7.2 Number of Blocks vs. Runtime

Table 1 and Table 2 display the optimal performance of Block BFGS versus three competitive methods for various problems. We now examine how the number of blocks B affects runtime and solution quality. Two settings are considered: (i) $n=100, p=1000, m=150$ groups with tolerance 10^{-9} ; and (ii) $n=2000, p=1000, m=200$ groups with tolerance 10^{-6} .

Table 3: Effect of block count on runtime and solution quality ($n=100$, $p=1000$, $m=150$, $\text{tol} = 10^{-9}$).

Blocks	Blocks/m ratio	Time (s)	Iters	$\ Ax_{\text{final}} - b\ $ (mean)	$\ x_{\text{final}} - x^*\ $ (mean)	$\ x_{\text{final}} - x^*\ $ (max)
2	0.013	273.2637	16990.5	0.8552	44.2368	56.2936
4	0.027	185.4530	9448.6	0.8552	44.2369	56.2939
5	0.033	148.6696	9901.7	0.8826	44.2370	56.2938
6	0.040	207.3646	10853.7	0.8552	44.2369	56.2937
8	0.053	111.6893	10698.1	0.8552	44.2368	56.2937
15	0.100	417.2210	16298.6	0.8552	44.2368	56.2937
23	0.153	574.9661	15651.5	0.8552	44.2368	56.2937
45	0.300	2912.3897	32636.4	0.8552	44.2370	56.2904
75	0.500	1885.3600	15500.2	0.8552	44.2373	56.2959
120	0.800	3799.7954	29754.2	0.8552	44.2372	56.2940

Table 4: Effect of block count on runtime and solution quality ($n=2000$, $p=1000$, $m=200$, $\text{tol} = 10^{-6}$).

Blocks	Blocks/m ratio	Time (s)	Iters	$\ Ax_{\text{final}} - b\ $ (mean)	$\ x_{\text{final}} - x^*\ $ (mean)	$\ x_{\text{final}} - x^*\ $ (max)
2	0.01	44.14	564.8	3.2197	0.0984	0.1053
4	0.02	14.94	307.3	3.2197	0.0984	0.1053
6	0.03	11.24	225.9	3.2197	0.0984	0.1053
8	0.04	9.12	189.7	3.2197	0.0984	0.1053
10	0.05	9.12	174.6	3.2197	0.0984	0.1053
12	0.06	8.55	149.8	3.2197	0.0984	0.1053
15	0.075	9.83	140.5	3.2197	0.0984	0.1053
18	0.09	9.97	129.5	3.2197	0.0984	0.1053
20	0.10	8.31	123.4	3.2197	0.0984	0.1053
25	0.125	9.76	117.3	3.2197	0.0984	0.1053
30	0.15	9.56	116.7	3.2197	0.0984	0.1053
36	0.18	12.77	124.5	3.2197	0.0984	0.1053
45	0.225	16.25	111.0	3.2197	0.0984	0.1053
50	0.25	16.24	105.8	3.2197	0.0984	0.1053
80	0.40	29.03	123.9	3.2197	0.0984	0.1053
100	0.50	28.47	102.7	3.2197	0.0984	0.1053
120	0.60	42.35	130.9	3.2197	0.0984	0.1053
150	0.75	59.44	166.6	3.2197	0.0984	0.1053

In both regimes, solution quality is essentially invariant to B : the residual $\|Ax_{\text{final}} - b\|$, the distance to the planted solution $\|x_{\text{final}} - x^*\|$, $\|x_{\text{final}}\|$, and the final objective remain stable across all block counts. Runtime, however, is *not* monotone in B . For $n=100$, the fastest configuration is $B=8$ (Table 3), while very small B (few, very large blocks) and very large B (many tiny blocks) both increase runtime. For $n=2000$, the minimum runtime occurs near $B=20$ (Table 4); iteration counts continue to drift slightly with B , but per-iteration cost grows when blocks become too small or too large, yielding a clear U-shaped trade-off. Practically, a *moderate* number of blocks (here, roughly B/G between 5% and 10%) provides the best accuracy-efficiency balance.

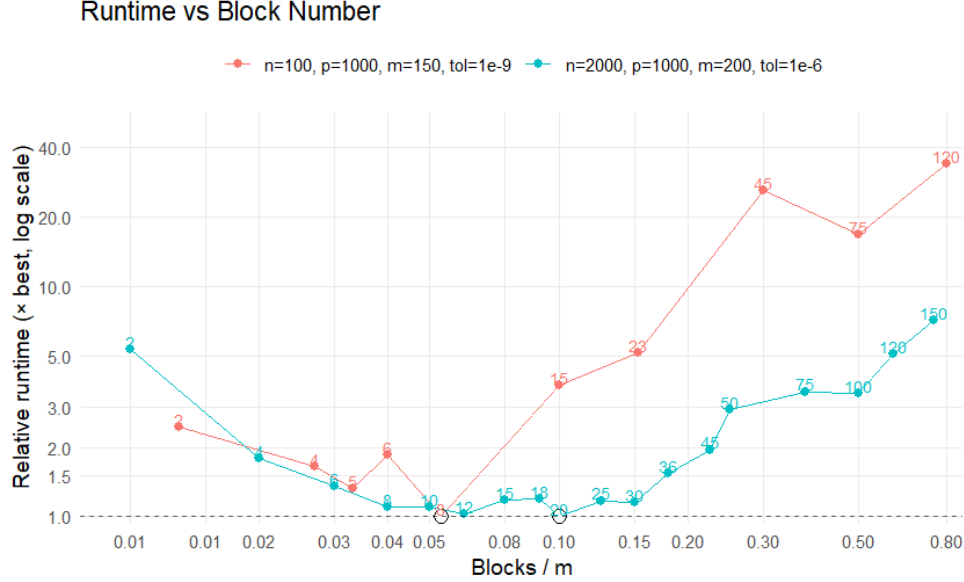


Figure 6: Runtime vs. block ratio (Blocks/ m) normalized by each experiment’s best runtime. Curves show relative runtime (\times best; log scale on both axes), overlaid for $n=100, p=1000, m=150$ and $n=2000, p=1000, m=200$. Hollow markers denote the minimizing block count for each experiment; numeric labels give the actual number of blocks. The dashed line at $1\times$ marks the per-experiment optimum.

7.3 Gene Expression Experiments

We evaluate the proposed algorithm on the breast cancer gene expression dataset originally from [10], and used in [2]. The dataset consists of 8,141 genes measured on 295 tumors, and is organized into two settings: (i) *pathways* from MSigDB canonical pathways: 639 groups are formed, and the average group size is 23.7 genes [12]; (ii) *edges* from the network in [11], yielding 42,594 overlapping size-2 groups. The objective is identical to the simulated experiments, and for both pathway and edge settings, we standardize features by columnwise z -scoring of A (gene expression), leaving b (labels) unchanged.

Protocol and hyperparameters. We run Block BFGS with Wolfe parameters $c_1 = 10^{-3}$, $c_2 = 0.3$, tolerance $\varepsilon = 10^{-8}$. We initialize at $x_0 = \mathbf{0}$ and use the provided weights and groups. For both edge and pathways, we run 10 independent trials with seeds 1-10.

- **Edges:** $A = X_{\text{edge}}, b = y, G = G_{\text{edge}}, w = w_{\text{edge}}, (\lambda_1, \lambda_2) = (10^{-3}, 10^{-1})$.
- **Pathways:** $A = X_{\text{path}}, b = y, G = G_{\text{path}}, w = w_{\text{path}}, (\lambda_1, \lambda_2) = (10^{-3}, 1)$.

For each run we record the runtime, number of iterations, the final iterate x , and the objective history $F(x_k)$. In the tables below, we report (i) *solution accuracy* and (ii) *runtime/efficiency* summaries across the repeated trials for the *edge* and *pathway* groupings.

Results On the *edge* groups (Table 5), a moderate block count yields the best efficiency. With 10 blocks, Block BFGS attains $F = 33.1074$, essentially matching FoGLASSO (gap $\approx 5 \times 10^{-4}$, and in fact slightly lower), while significantly reducing runtime from 1.68×10^5 s (FoGLASSO) to 2.43×10^4 s ($\approx 6.9\times$ faster). The estimated parameter is also closer to the FoGLASSO solution ($\|x_{\text{final}} - x_{\text{FoGLASSO}}\| = 0.1593$), in contrast to other algorithms. L-BFGS is the fastest in absolute

Table 5: Breast cancer gene expression (edge groups, 42,594 pairs): performance summary.

Method	Blocks	Iterations	Runtime	$\ x_{\text{final}} - x_{\text{FoGLASSO}}\ $	$F_{\text{final}} - F_{\text{FoGLASSO}}$
Block BFGS	5	25582	16769.86	0.1809	0.9144
Block BFGS	10	27221	24337.61	0.1593	0.0005
Block BFGS	15	22679	28995.83	0.1647	0.0006
Block BFGS	20	32729	52654.99	0.1702	0.0006
Block BFGS	25	28497	58196.33	0.1786	0.0006
Block BFGS	30	25634	63709.73	0.1744	0.0006
Block BFGS	50	112681	432996.02	0.1792	0.0006
L-BFGS	N.A.	5316	727.25	0.4542	0.0733
BFGS	N.A.	250	549710.83	0.7106	0.0724
FoGLASSO	N.A.	3217	168075.74	N.A.	N.A.
InexactPG	N.A.	45.9	22934.20	0.9597	7.4170

Table 6: Breast cancer gene expression (pathway groups, 637 groups): performance summary.

Method	Blocks	Iterations	Runtime	$\ x_{\text{final}} - x_{\text{FoGLASSO}}\ $	$f_{\text{final}} - F_{\text{FoGLASSO}}$
Block BFGS	5	2798	3056.13	0.0097	0.0007
Block BFGS	10	1727	463.36	0.0099	0.0007
Block BFGS	15	1571	222.14	0.0097	0.0007
Block BFGS	20	1414	134.44	0.0098	0.0007
Block BFGS	25	1144	134.08	0.0098	0.0007
Block BFGS	30	1117	187.32	0.0097	0.0007
Block BFGS	40	1026	226.40	0.0099	0.0007
Block BFGS	50	1020	437.33	0.0097	0.0007
Block BFGS	100	1616	615.55	0.0097	0.0007
Block BFGS	200	2409	1998.70	0.0097	0.0007
L-BFGS	N.A.	2397	20.04	0.3944	1.5076
BFGS	N.A.	5942	91417.53	0.0098	0.0007
FoGLASSO	N.A.	634	173745.33	N.A.	N.A.
InexactPG	N.A.	37	66371.84	0.1332	0.3629

time (7.27×10^2 s) but settles at a higher objective ($F = 33.18$, +0.22% vs. FoGLASSO). Full BFGS reaches a slightly worse objective ($F = 33.32$) at prohibitive cost (5.50×10^5 s). Using too few blocks (e.g., 5) speeds computations but degrades the objective ($F = 34.02$); using many blocks (e.g., 50) does not improve F but inflates runtime (to 4.33×10^5 s).

On the *pathway* groups (Table 6), Block BFGS with 15-25 blocks converges to the FoGLASSO solution quality *three orders of magnitude faster*. For example, with 20 blocks, we obtain $F \approx 64.79533$ (gap $\approx 7 \times 10^{-4}$) and $\|x_{\text{final}} - x_{\text{FoGLASSO}}\| \approx 0.0098$ in 1.34×10^2 s, compared to 1.74×10^5 s for FoGLASSO ($\sim 1,300\times$ speedup). Full BFGS attains the same objective level but requires 9.14×10^4 s ($\sim 680\times$ slower than Block BFGS). InexactPG is both slower (6.64×10^4 s) and less accurate ($F = 65.16$, +0.56%; distance 0.1332). L-BFGS is again fastest (2.00×10^1 s) but with a noticeably higher objective ($F = 66.30$, +2.33%) and a much larger parameter deviation (0.3944). Increasing blocks far beyond 25 yields no meaningful accuracy gains but raises runtime.

7.4 Post-Processing via FoGLASSO Warm-Starts

Across both gene-expression and simulated experiments, *Block BFGS* attains FoGLASSO-level solutions at far lower cost than FoGLASSO, full BFGS, or InexactPG. Although *L-BFGS* is the fastest in runtime, its solution consistently deviates further from FoGLASSO. To compare solution quality across methods more directly, we *warm-start FoGLASSO* from the terminal iterate of each BFGS variant. For each method $M \in \{\text{Block BFGS}, \text{BFGS}, \text{L-BFGS}\}$, we rerun FoGLASSO with the same hyperparameters and stopping criterion (tolerance 10^{-9} before and after), but initialize

at $x_0 = x_{\text{final}}^{(M)}$. If $x_{\text{final}}^{(M)}$ is already near the FoGLASSO solution, the warm-started run should require few iterations and produce negligible changes. We assess post-processing quality by the *parameter change* $\|x_{\text{post}}^{(M)} - x_{\text{final}}^{(M)}\|$ and the *objective change* $F(x_{\text{post}}^{(M)}) - F(x_{\text{final}}^{(M)})$, together with the warm-start iterations and runtime. Because the model induces group sparsity, we also compare recovered group support: letting $\mathcal{Z}(x)$ be the set of groups with zero norm in x and \mathcal{G} the collection of all groups, we report $\text{Agreement}\% = 100 \times |\{g : \mathbf{1}_{\{g \in \mathcal{Z}(x_{\text{post}}^{(M)})\}} = \mathbf{1}_{\{g \in \mathcal{Z}(x_{\text{FoGLASSO}})\}}\}|/|\mathcal{G}|$ and $\text{OnlyFoZero}\% = 100 \times |\mathcal{Z}(x_{\text{FoGLASSO}}) \setminus \mathcal{Z}(x_{\text{post}}^{(M)})|/|\mathcal{G}|$.

Table 7: FoGLASSO warm-starts (simulated data; $n=100$, $p=1000$, groups = 150; tol = 10^{-9}).

Method	Runtime (s)	Iterations	$\ x_{\text{post}}^{(M)} - x_{\text{final}}^{(M)}\ $ (mean)	$F(x_{\text{post}}^{(M)}) - F(x_{\text{final}}^{(M)})$ (mean)	OnlyFoZero (%)	Agreement (%)
Block BFGS	369.0993	21.4	2.42×10^{-12}	4.26×10^{-14}	0.5333	99.22
BFGS	372.5878	21.1	2.42×10^{-12}	4.26×10^{-14}	0.5333	99.22
L-BFGS	2410.9961	156.2	2.51×10^{-10}	3.64×10^{-12}	1.8660	96.66

Post-processing: Simulated Experiments For the simulated setting, warm-starting FoGLASSO from *Block BFGS* (or full BFGS) requires only ≈ 21 iterations and $\approx 3.7 \times 10^2$ s, with changes at numerical precision: $\|x_{\text{post}}^{(M)} - x_{\text{final}}^{(M)}\| \approx 2.4 \times 10^{-12}$ and $F(x_{\text{post}}^{(M)}) - F(x_{\text{final}}^{(M)}) \approx 4.3 \times 10^{-14}$. Support metrics are likewise aligned (OnlyFoZero $\approx 0.53\%$, Agreement $\approx 99.2\%$). By contrast, warm-starting from *L-BFGS* takes substantially longer (156 iterations and $\approx 3.41 \times 10^3$ s) and corrects a noticeably larger discrepancy ($\|x_{\text{post}}^{(M)} - x_{\text{final}}^{(M)}\| \approx 2.5 \times 10^{-10}$, F change $\approx 3.6 \times 10^{-12}$), with lower support agreement (96.66%) and more missed zeros (OnlyFoZero = 1.866%). These results confirm that Block BFGS arrives extremely close to the FoGLASSO solution—both in parameters and in group support—while L-BFGS endpoints deviate more and require substantial correction despite their standalone speed. In fact, the total runtime for L-BFGS plus post-processing is $16.515 + 2410.9961 = 2427.5111$ s, which still exceeds the combined time for Block BFGS plus post-processing ($111.6893 + 369.0993 = 480.7886$ s, block=8).

Table 8: FoGLASSO warm-starts on gene-expression data. *Iters* is the number of FoGLASSO iterations taken from each warm start. **Total Runtime** = (algorithm runtime) + (warm-start runtime).

Dataset	Method	Iters	Runtime (s)	Total Runtime (s)	$\ x_{\text{final}} - x_{\text{prev}}\ $ (mean)	$F_{\text{PG}} - F_{\text{prev}}$ (final)	OnlyFoZero (%)	Agreement (%)
Edge	Block BFGS	20	40814.83	65152.44	0.1593	33.1074	8.0281	89.64
	Block BFGS	50	47280.52	71618.13	0.1592	33.1074	2.45	97.05
	BFGS	20	50754.29	600465.12	0.5706	33.2928	57.2003	83.64
	BFGS	50	53432.14	603142.97	0.5689	33.2710	2.45	97.05
	L-BFGS	120	52304.18	53031.42	0.7028	33.1760	27.7234	81.56
	L-BFGS	200	55510.22	56237.46	0.6904	33.1701	24.43	82.32
Pathway	Block BFGS	2	3652.48	3874.62	0.0097	0.0006	21.1823	93.25
	Block BFGS	5	4711.62	4933.76	0.0097	0.0007	1.4778	99.53
	Block BFGS	10	6155.50	6377.64	0.0097	0.0007	0.4926	99.84
	Block BFGS	15	7652.49	7874.62	0.0097	0.0007	0	100.00
	BFGS	2	3614.51	95032.04	0.0098	0.0006	21.1823	92.62
	BFGS	5	4830.48	96248.01	0.0098	0.0007	0.9852	99.69
	BFGS	10	6357.08	97774.61	0.0098	0.0007	0	100.00
	BFGS	15	7672.34	99089.87	0.0098	0.0007	0	100.00
	L-BFGS	50	16652.05	16672.08	0.2448	0.4636	23.6453	84.93
	L-BFGS	100	32085.20	32105.24	0.0883	0.0484	15.2709	94.66
	L-BFGS	150	45621.52	45641.56	0.0327	0.0084	2.9556	99.06

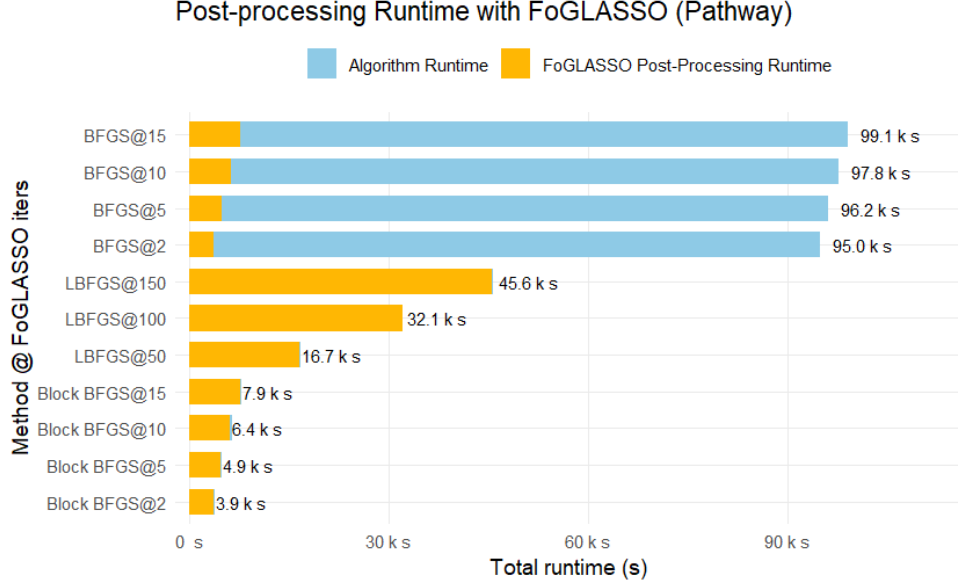


Figure 7: Pathway dataset: total runtime decomposition for BFGS plus FoGLASSO. Bars show the warm-start algorithm time (blue) and the FoGLASSO post-processing time (yellow); labels use **Method@iters**. Block-BFGS attains the same final accuracy after post-processing as the other warm starts but with substantially lower total runtime, whereas L-BFGS requires large FoGLASSO clean-up despite a short initial runtime.

Post-processing: Gene Expression Experiment From Table 8 and Figure 7, we can see that on the *Pathway* dataset, warm-starting FoGLASSO from *Block BFGS* requires only 5-15 iterations and makes vanishing adjustments ($\|x_{\text{post}} - x_{\text{final}}\| \approx 9.7 \times 10^{-3}$, $F(x_{\text{post}}) - F(x_{\text{final}}) \approx 7 \times 10^{-4}$). The recovered group support essentially matches FoGLASSO (Agreement = 99.53% \rightarrow 100%; OnlyFoGLASSOZero = 1.4778% \rightarrow 0% as the warm-start iterations increase). Crucially, when we compare *total runtime* (algorithm runtime + post-processing), Block BFGS finishes in 4.93×10^3 - 7.87×10^3 s, which is an order of magnitude faster than full BFGS (9.62×10^4 - 9.91×10^4 s) and also substantially faster than L-BFGS (1.67×10^4 - 3.21×10^4 s). Although L-BFGS is fast in isolation, its endpoints need 50-100 warm-start iterations and sizeable corrections ($\|x_{\text{post}} - x_{\text{final}}\| = 2.448 \times 10^{-1} \rightarrow 8.83 \times 10^{-2}$, objective change = 0.4636 \rightarrow 0.0484), with markedly lower support agreement (84.93-94.66%) and many more missed zero groups (OnlyFoGLASSOZero = 23.6453% \rightarrow 15.2709%). Full BFGS attains FoGLASSO-quality solutions after post-processing but incurs prohibitive total runtime due to its expensive pre-run. Overall, Block BFGS delivers FoGLASSO-level accuracy—both numerically and in group support—at the lowest overall cost once post-processing is accounted for.

The same pattern holds on *Edge*. Block BFGS warm-starts with 20-50 iterations finish in 6.52×10^4 – 7.16×10^4 s with improving support (Agreement 89.64% \rightarrow 97.05%, OnlyFoGLASSOZero 8.03% \rightarrow 2.45%). BFGS ultimately attains comparable agreement at 50 iterations, but its total runtime is 6.03×10^5 s, roughly 8–9 \times slower than Block BFGS; with only 20 iterations its clean-up is even more expensive (6.00×10^5 s; OnlyFoGLASSOZero 57.2%). L-BFGS has small algorithm times but remains inaccurate even after 120-200 iterations (Agreement 81.56%-82.32%; OnlyFoGLASSOZero 27.72%-24.43%), so matching the Block-/full-BFGS accuracy would require many more FoGLASSO steps and a larger total runtime than reported.

8 Experiments with Logistic Regression+Overlapping GroupLASSO

We also evaluate the performance of the Block BFGS algorithm on several real-world classification problems from the LIBSVM repository [8]. In these experiments, the smooth component of the objective function is based on logistic regression rather than least-squares. Specifically, for a given dataset the loss function is defined as

$$f(x) = \sum_{i=1}^N \log\left(1 + \exp(-b_i(A_{i,:}x))\right),$$

where $A \in \mathbb{R}^{N \times n}$ is the feature matrix, $b \in \{-1, 1\}^N$ are the binary class labels (with any label originally 0 converted to -1), and N denotes the number of samples. The computational setup for these experiments followed the idea from [3] with adjustments. Each of 8 datasets is loaded using the `libsvmread` function in MATLAB. The feature matrix is converted to a full matrix and the labels are adjusted so that they are in the set $\{-1, 1\}$. Overlapping groups are then constructed by partitioning the feature indices into a fixed number (in our experiments, 10) of groups. Each group is formed by selecting a block of consecutive features, with consecutive groups overlapping by a quarter of the group size. This grouping strategy is designed to capture feature correlations and to reflect realistic scenarios in high-dimensional classification problems.

The overall regularized objective function is given by

$$F(x) = f(x) + \lambda_1 \|x\|_1 + \lambda_2 \sum_{g \in G} w_g \|x_g\|,$$

where λ_1 and λ_2 are regularization parameters (set to 0.1 and 1, respectively) and the group weights w_g are set uniformly to one. The Block BFGS algorithm is then applied with EBLs line search ($c1 = 10^{-3}$ and $c2 = 0.3$), and a tolerance of 10^{-6} . We record performance metrics such as the overall runtime, iteration count, final objective value, misclassification rate, and logistic loss for each dataset. Also, the LIBSVM datasets span a wide range of dimensions. This variety allows us to assess the scalability and robustness of the Block BFGS method in realistic, high-dimensional classification scenarios.

Results of LIBSVM Experiments Tables 9 and 10 summarize the experimental results in terms of logistic loss, the average runtime (in seconds), and runtime per iteration:

Table 9: Runtime and Iteration Count for LIBSVM Datasets

Dataset	Block BFGS			FoGLASSO			InexactPG			BFGS		
	# Iters	RunTime	Runtime/Iter	# Iters	RunTime	Runtime/Iter	# Iters	RunTime	Runtime/Iter	# Iters	RunTime	Runtime/Iter
a9a	156	30.9802	0.1986	310	5.2872	0.0171	40	1.7809	0.0445	96	25.3251	0.2638
colon-cancer	905	58.4914	0.3323	176	92.4293	0.5252	43	110.3279	2.5658	430	810.5214	1.8849
duke	7	3.3119	0.4731	1369	1248.6311	0.9121	51	330.01	6.47	1107	2650.17	2.39
leukemia	904	937.4091	1.03696	139	199.7885	1.4373	40	351.30	8.76	1034	2411.06	2.33
mushrooms	295	11.0477	0.0374	121	22.8852	0.1891	41	0.3494	0.0085	236	12.6167	0.0535
w8a	410	337.9433	0.8243	710	46.0110	0.0648	39	5.2049	0.1335	182	40.3435	0.2217

Solution Accuracy. Block BFGS achieves logistic loss values that are generally competitive with those obtained by BFGS and FoGLASSO. For instance, on the a9a dataset, Block BFGS obtains an average loss of about 0.322, matching the performance of the full-matrix method and closely tracking FoGLASSO. Although in some cases—such as on the colon-cancer dataset—FoGLASSO presents marginally lower loss values, Block BFGS still delivers reliable and high-quality solutions across all tested datasets.

Table 10: Logistic Loss for LIBSVM Datasets

Dataset	Block BFGS		FoGLASSO		InexactPG		BFGS	
	Total Loss	Average Loss	Total Loss	Average Loss	Total Loss	Average Loss	Total Loss	Average Loss
a9a	10511.2410	0.3219	10517.6898	0.3230	17151.2897	0.5267	10511.2367	0.3228
colon-cancer	3.8928	0.0628	1.3063	0.0301	30.8809	0.4981	1.8903	0.0304
duke	0.7103	0.0115	1.9964	0.0454	4.4505	0.1001	0.7100	0.0122
leukemia	0.9929	0.0261	0.5755	0.0151	4.0100	0.1100	0.9900	0.0302
mushrooms	1.5939	0.0002	1.6024	0.0002	1768.8982	0.2177	1.5932	0.0001
w8a	5585.8749	0.1123	5615.0068	0.1129	23075.0890	0.4639	5585.8582	0.1122

Runtime and Iteration Efficiency In terms of computational efficiency, Block BFGS requires a moderate number of iterations. While its per-iteration cost is somewhat higher than that of FoGLASSO (which benefits from simpler proximal updates), the overall runtime remains competitive, particularly on larger datasets like w8a where curvature information helps accelerate convergence. Datasets with lower sample sizes (e.g., colon-cancer or leukemia) necessitate more iterations, yet Block BFGS consistently converges in a reasonable total runtime compared to the alternatives.

Overall, Block BFGS strikes a solid balance between leveraging second-order information and maintaining computational efficiency. Its performance on LIBSVM datasets demonstrates robustness across different problem scales and highlights its potential for handling high-dimensional structured classification tasks. Although iteration counts and runtime may vary with data characteristics, the method consistently delivers competitive solution quality, making it a promising approach for real-world applications.

9 Conclusion

We proposed Block BFGS, a block-coordinate-descent adaptation of BFGS for composite objectives with overlapping groups. The method refreshes curvature only on the active block, maintains a block-diagonal Hessian approximation, and employs a composite directional-derivative Wolfe rule to navigate non-smoothness without proximal operators. On the theory side, we proved global convergence under standard assumptions and extended the guarantees to a milder smooth-extension setting on neighborhoods containing the iterates, using angle and trace/determinant controls to prevent degeneracy. Empirically, across least-squares and logistic models with structured sparsity, Block BFGS consistently matches the solution quality of proximal-gradient baselines while delivering substantial runtime savings, particularly when curvature is informative and blocks are moderately sized.

9.1 Limitations

Block BFGS inherits sensitivity to line-search quality and may be less advantageous when blocks are extremely small, overlap is dense enough to negate block locality, or curvature is poorly conditioned within blocks. Our analysis focuses on convex composites; extending guarantees to broadly nonconvex settings remains open.

9.2 Outlook

The evidence in this paper indicates that *Block BFGS* is especially effective when

1. problems exhibit clear block structure (e.g., group or overlapping penalties) so that within-block curvature accelerates progress;

2. dimensions are medium-to-large, making full BFGS prohibitive while first-order methods underuse curvature;
3. objective evaluations dominate cost, allowing block-local quasi-Newton updates to amortize their overhead.

These observations motivate several extensions:

1. **Limited-memory and sketching.** Develop per-block L-BFGS/sketched updates to cut storage and Hessian-approximation cost without sacrificing curvature quality.
2. **Adaptive blocks and screening.** Learn block selection and block sizes on the fly; integrate safe/strong screening to tighten active sets and reduce unnecessary updates.
3. **Stochastic and distributed variants.** Combine block quasi-Newton steps with variance-reduced gradients and asynchronous, multi-node implementations for streaming and large-scale settings.

Together, these directions aim to preserve quasi-Newton accuracy while further reducing per-iteration cost and broadening Block BFGS’s applicability.

References

- [1] A. S. Lewis and M. L. Overton. (2012). Nonsmooth Optimization via Quasi-Newton Methods. *Mathematical Programming*, 134(1-2), 499-525. <https://link.springer.com/article/10.1007/s10107-012-0514-2>.
- [2] L. Yuan, J. Liu, and J. Ye. (2011). Efficient Methods for Overlapping Group Lasso. In *Advances in Neural Information Processing Systems (NIPS 2011)*.
- [3] Y. Dai and D. P. Robinson. (2022). Inexact Proximal-Gradient Methods with Support Identification. *arXiv:2211.02214*.
- [4] R. H. Byrd, J. Nocedal, and Y.-X. Yuan. (1987). Global Convergence of a Class of Quasi-Newton Methods on Convex Problems. *SIAM Journal on Numerical Analysis*, 24(5), 1152-1171. <https://doi.org/10.1137/0724077>.
- [5] J. Guo and A. S. Lewis. (2018). Nonsmooth Variants of Powell’s BFGS Convergence Theorem. *SIAM Journal on Optimization*, 28(1), 284-301. <https://doi.org/10.1137/17M1121883>.
- [6] C. G. Broyden. (1970). The Convergence of a Class of Double-rank Minimization Algorithms. *IMA Journal of Applied Mathematics*, 6(1), 76-90.
- [7] P. Tseng. (2001). Convergence of a Block Coordinate Descent Method for Nondifferentiable Minimization. *Journal of Optimization Theory and Applications*, 109(3), 475-494.
- [8] C.-C. Chang and C.-J. Lin. (2011). LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3), Article 27.
- [9] M. Yan. (2014). Extension of Convex Function. *Journal of Convex Analysis*, 21, 965-987.
- [10] M. J. Van de Vijver, et al. (2002). A gene-expression signature as a predictor of survival in breast cancer. *The New England Journal of Medicine*, 347(25):1999-2009.
- [11] H. Y. Chuang, E. Lee, Y. T. Liu, D. Lee, and T. Ideker. (2007). Network-based classification of breast cancer metastasis. *Molecular Systems Biology*, 3:140.
- [12] A. Subramanian, et al. (2005). Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences*, 102(43):15545-15550.
- [13] Wright, S. and Recht, B. (2022). Optimization for Data Analysis. *Cambridge University Press*. <https://doi.org/10.1017/9781009004282>.
- [14] W. Wang, H. Yuan, J. Han, and W. Liu. (2022). PCLassoLog: A protein complex-based, group Lasso-logistic model for cancer classification and risk protein complex discovery. *Computational and Structural Biotechnology Journal*, 21, 365-377. <https://doi.org/10.1016/j.csbj.2022.12.005>.
- [15] S. J. Wright. (2015). Coordinate descent algorithms. *Mathematical Programming*, 151, 3-34. <https://doi.org/10.1007/s10107-015-0892-3>

A Notation Dictionary

Symbol	Meaning	Type / Dimension
Data, dimensions, sets		
n	number of samples (rows of A)	integer
p	number of features/variables (columns of A)	integer
m	number of (possibly overlapping) groups	integer
n_B	number of coordinate blocks	integer
A	design matrix	$\mathbb{R}^{n \times p}$
b	response / labels	\mathbb{R}^n or $\{-1, 1\}^n$
x	parameter vector	\mathbb{R}^p
\mathcal{I}_i	index set of block i ; $ \mathcal{I}_i = p_i$	subset of $\{1, \dots, p\}$
\mathcal{G}	group index sets (may overlap)	$G_\gamma \subset \{1, \dots, p\}$
$\{G_1, \dots, G_m\}$		
\mathcal{H}_i	groups affected by block \mathcal{I}_i	$\{\gamma : G_\gamma \cap \mathcal{I}_i \neq \emptyset\}$
w_γ	group weight for G_γ	$w_\gamma > 0$
\mathcal{D}	level set $\{x : F(x) \leq F(x_0)\}$	subset of \mathbb{R}^p
$[p]$	shorthand for $\{1, \dots, p\}$	set
Objective and derivatives		
F	composite objective $F(x) = f(x) + g(x)$	$\mathbb{R}^p \rightarrow \mathbb{R}$
f	smooth convex loss (LS or logistic)	$\mathbb{R}^p \rightarrow \mathbb{R}$
g	convex (possibly non-smooth) penalty	$\mathbb{R}^p \rightarrow \mathbb{R}$
λ_1, λ_2	ℓ_1 and group penalties	nonnegative scalars
$\nabla f(x)$	gradient of f	\mathbb{R}^p
$\nabla_i f(x)$	block gradient on \mathcal{I}_i	\mathbb{R}^{p_i}
$\nabla^2 f(x)$	Hessian of f	$\mathbb{R}^{p \times p}$
$\partial g(x)$	subdifferential of g	set in \mathbb{R}^p
$\partial_i g(x)$	block subdifferential on \mathcal{I}_i	set in \mathbb{R}^{p_i}
$Dg(x; d)$	one-sided directional derivative of g at x along d	scalar
m, M	curvature bounds for f on \mathcal{D}	$0 < m \leq M$
Group penalty helpers		
$\psi_\gamma(z)$	group term on G_γ (e.g., $\lambda w_\gamma \ z\ _2$)	$\mathbb{R}^{ G_\gamma } \rightarrow \mathbb{R}$
$a_\gamma, b_\gamma, c_\gamma$	cache for $\ x_{G_\gamma} + \alpha p_{G_\gamma}\ _2 = \sqrt{a_\gamma + b_\gamma \alpha + c_\gamma \alpha^2}$	scalars
Block structure and operators		
u_i	embedding: inserts a block vector into \mathbb{R}^p	$u_i : \mathbb{R}^{p_i} \rightarrow \mathbb{R}^p$
B_k	(typically block-diagonal) SPD metric at iter. k	$\mathbb{R}^{p \times p}$
$B_k^{(i,i)}$	block- i submatrix (updated only when $i = i_k$)	$\mathbb{R}^{p_i \times p_i}$
Per-iteration quantities (active block i_k)		
i_k	active block index at iteration k	$\{1, \dots, n_B\}$
x_k	iterate before updating block i_k	\mathbb{R}^p
$h_k^{(i)}$	composite residual $h_k^{(i)} \in \nabla_i f(x_k) + \partial_i g(x_k)$	\mathbb{R}^{p_i}
$p_k^{(i)}$	search direction on block i	$-[B_k^{(i,i)}]^{-1} h_k^{(i)}$
α_k	accepted stepsize for block move	scalar > 0
$s_k^{(i)}$	block step $= \alpha_k p_k^{(i)}$	\mathbb{R}^{p_i}
$y_k^{(i)}$	smooth curvature diff. $= \nabla_i f(x_{k+1}) - \nabla_i f(x_k)$	\mathbb{R}^{p_i}
<i>continued on next page</i>		

Symbol	Meaning	Type / Dimension
$\theta_k^{(i)}$	angle between $-\nabla_i f(x_k)$ and $s_k^{(i)}$	$\cos \theta_k^{(i)} = \frac{(s_k^{(i)})^\top (-\nabla_i f(x_k))}{\ s_k^{(i)}\ \ \nabla_i f(x_k)\ }$
π_k	random permutation of blocks in a sweep	permutation of $\{1, \dots, n_B\}$
S_k	sum of block step norms in a sweep	scalar
Line search (two variants used)		
c_1, c_2	Wolfe constants with $0 < c_1 < c_2 < 1$	scalars
$\phi(\alpha)$	1D restriction $\phi(\alpha) = F(x_k + \alpha u_{i_k}(p_k^{(i_k)}))$	scalar fn. of α
$\phi'_+(\alpha)$	one-sided derivative of ϕ at α	scalar
Composite Wolfe (impl.)	accept $\alpha > 0$ if $\phi(\alpha) \leq \phi(0) + c_1 \alpha \phi'_+(0)$ and $\phi'_+(\alpha) \geq c_2 \phi'_+(0)$	rule on F
Hybrid Wolfe (proofs)	Armijo on F , curvature on f : $F(x_k + \alpha u_i p) \leq F(x_k) + c_1 \alpha \nabla_i f(x_k)^\top p$, $\nabla_i f(x_k + \alpha u_i p)^\top p \geq c_2 \nabla_i f(x_k)^\top p$	mixed rule
Quasi-Newton updates		
BFGS (block i)	$B_{k+1}^{(i,i)} = B_k^{(i,i)} - \frac{B_k^{(i,i)} s s^\top B_k^{(i,i)}}{s^\top B_k^{(i,i)} s} + \frac{y y^\top}{y^\top s}$ (when $y^\top s > 0$)	SPD update
$\text{tr}(\cdot), \det(\cdot)$	trace and determinant	scalars
Cached LS quantities		
w_k	cached smooth image Ax_k	\mathbb{R}^n
V_i	$A_{i,\mathcal{I}_i}^\top b$ (constant in block i)	\mathbb{R}^{p_i}
<i>Note</i>	w_k (cache) vs. w_γ (group weight) — meaning is by context	—
Sampling / counting		
$\mathbf{1}_k^{(i)}$	indicator $\mathbf{1}\{i_k = i\}$	$\{0, 1\}$
$N_i(K)$	number of updates of block i up to K	integer
Angles, stepsize, and summability (analysis)		
α_{\min}	positive lower bound on stepsizes on a subsequence (if exists)	scalar
Zoutendijk sum	$\sum_k \frac{(\nabla f(x_k)^\top p_k)^2}{\ p_k\ ^2} < \infty$ under Wolfe	series
Miscellaneous		
ε	termination tolerance (e.g., $\sum_i \ s_k^{(i)}\ $)	scalar
$\ \cdot\ , \langle \cdot, \cdot \rangle$	Euclidean norm and inner product	—