bAdag: an adaptive block coordinate gradient method for smooth nonconvex functions

Giovanni Seraghiti*†

Abstract

A new Block Coordinate Gradient (BCG) method, dubbed bAdag, for smooth, nonconvex minimization problem is proposed; it falls in the class of Objective Function Free Optimization (OFFO) methods, and it is based on the AdaGrad algorithm. At each iteration, our method computes an adaptive step size based on the cumulative sum of block gradients, instead of full gradients as in AdaGrad-type methods. We prove ergodic, sublinear convergence rates for the bAdag algorithm when minimizing a smooth, possibly nonconvex objective under the (block) Lipschitz continuity assumption on the gradient. Our theory covers three widely popular block selection strategies: the Cyclic (C) rule, Uniform Random selection (UR), and the greedy Gauss-Southwell (GS) rule. We also extend our algorithm and its convergence theory to box-constrained smooth functions. We validate the proposed algorithms through synthetic and real-world experiments.

Keywords Block coordinate gradient, Adaptive gradient, Convergence rates, Nonconvex optimization

1 Introduction

We are interested in solving the following unconstrained optimization problem

$$\min_{x\in\mathbb{R}^N} f(x), \tag{1.1}$$

with variables $x = (x_1, \dots, x_N) \in \mathbb{R}^N$, and where f is a continuously differentiable, possibly nonconvex function. In large-scale optimization, a common iterative approach is Block Coordinate Descent (BCD), where, at iteration k, the objective function is minimized with respect to a subset of variables $\{x_i\}_{i\in b_{\ell_k}}$ and $b_{\ell_k} \subseteq \{1 \dots N\}$, while the others remain fixed. Among the BCD schemes, a widely-used approach in smooth settings is the Block Coordinate Gradient (BCG) method, which uses the following update:

$$x^{k+1} = x^k - \alpha_k g_k^{(\ell_k)}, \tag{1.2}$$

where x_k is the k-th iterate, $g_k \overset{\text{def}}{=} \nabla f(x_k)$ denotes the gradient of f at x_k, and $g_k^{(\ell_k)}$ is the vector containing the partial derivatives corresponding to the indices in b_{ℓ_k} and zero elsewhere, and α_k is the step size. Due to its simplicity, this approach has been applied in a variety of contexts, especially in applications where the cost of one iteration of gradient descent is at least as expensive as updating all the components in the BCG scheme. Such objective functions are widely popular and include quadratic functions and logistic loss [67, 69, 72].

BCG methods are often classified based on the criteria for selecting the block to optimize at each iteration. Among the widely-known criteria, we identify three main categories:

*University of Mons, Rue de Houdain 9, 7000 Mons, Belgium. Email: giovanni.seraghiti@umons.ac.be

†Dipartimento di Ingegneria Industriale, Università degli Studi di Firenze, Viale Morgagni 40/44, 50134 Firenze, Italia. GS acknowledge the support by the European Union (ERC consolidator, eLinoR, no 101085607). The work of GS was partially supported by INdAM-GNCS through Progetti di Ricerca.

1. greedy strategies, such as the *Gauss-Southwell* (GS) rule, which selects the block corresponding to the largest gradient norm;

2. random strategies, where the block is chosen according to some probability distribution, for example, *Uniformly at Random* (UR);

3. *Cyclic* (C) rules, updating the blocks of variables cyclically.

We further discuss the different types of BCG methods and their convergence in Section 2.

While the criteria for selecting the block to optimize at each iteration have been extensively investigated, surprisingly, much less effort has been devoted to the choice of the step α_k and how it affects the convergence of the BCG method. To the best of our knowledge, the majority of the contributions on BCG either consider constant step sizes or employ line-search strategies, which require several function evaluations at each iteration. However, gradient methods directly evaluating the objective function exhibit slow convergence when the function is expensive to compute or suffer from poor accuracy in noisy scenarios. A possible alternative are Objective Function Free Optimization (OFFO) methods that never evaluate the objective function. One of the reasons of their popularity is that, when dealing with noisy functions, they are more robust than gradient methods requiring function evaluations [32]. Adaptive gradient methods are relevant examples of OFFO algorithms that rely on the history of the past gradient values to compute the step size, or the step direction, or both at every iteration. Prominent examples are AdaGrad [28, 64], Adam [50], RMSprop [85], ADADELTA [97], Shampoo [43], and Muon [48], with a unified framework covering several of the mentioned algorithms recently proposed in [37]. Adaptive gradient methods represent state-of-the-art algorithms to solve large machine learning problems where the cost of computing the objective function is sometimes prohibitive. Among these methods, AdaGrad has gain particular attention due to its flexibility and the simplicity arguments used in its convergence theory. Despite the popularity of adaptive gradient methods in standard first-order optimization, very few works apply adaptive step size selection strategies in the context of BCG, and the convergence rate of such algorithms is still unexplored [59].

Outline and Contributions We discuss relevant contributions on BCG in Section 2; we describe the bAdag framework in Section 3 and its convergence rate analysis in Section 4. In particular, we prove a general nonmonotone descent lemma (Lemma 4.1), and we derive a unified convergence rate analysis (Theorem 4.5) that covers GS, UR, and C block selection strategies. In Section 5, we extend our algorithm and prove new convergence rates for box-constrained problems. Finally, we show exhaustive numerical results both on convex and nonconvex problems in Section 6.

Our main contributions can be summarized as follows:

1. We introduce bAdag: a novel adaptive BCG method inspired by AdaGrad. To our knowledge, bAdag is the first adaptive BCG algorithm with provable nonasymptotic convergence rates in the nonconvex setting covering GS, UR, and C block selection rules, while they are usually analyzed separately in the literature.

2. We establish an ergodic sublinear convergence rate for the bAdag algorithm by proving that, under the GS and UR selection rules, the average gradient norm and its expectation respectively, decrease at a rate of $\mathcal{O}\left(\frac{1}{\sqrt{k+1}}\right)$, where k denotes the iteration index. Furthermore, for the cyclic variant of bAdag, we derive a sublinear convergence rate of $\mathcal{O}\left(\frac{1}{\sqrt{T+1}}\right)$, where T represents the complete cycle in which all blocks of variables are updated once.

3. We extend the bAdag algorithm to box-constrained minimization and we establish similar rates to the unconstrained case.

Finally, we illustrate our findings with exhaustive numerical results on several applications, including least square and logistic regression, and robust Nonnegative Matrix Factorization (NMF).

Notation We denote $v_{i,k}$ the i-th component of a vector $v_k \in \mathbb{R}^n$. The gradient of f is denoted as $g \stackrel{\text{def}}{=} \nabla f$; thus, at the k-th iteration, the gradient evaluated at the current iterate x_k is $g_k = g(x_k)$ and its i-th component is $g_{i,k}$. We introduce a partition of the variable into disjoint blocks denoted $\{1, \dots, N\} = \bigcup_{\ell=1}^{d} b_\ell$, where $b_\ell \subseteq \{1, \dots, N\}$, $b_\ell \cap b_s = \emptyset$ for $\ell \neq s$. We denote $g_{i,k}^{(\ell)}$ the block gradient containing only the components in b_ℓ and being zero elsewhere, that is,

$$g_{i,k}^{(\ell)} = \begin{cases} g_{i,k} & i \in b_\ell \\ 0 & \text{otherwise.} \end{cases} \tag{1.3}$$

Unless specified otherwise, $\|\cdot\|$ is the Euclidean norm on $\mathbb{R}^n$. Since in this paper we deal with nonsynmptotic rates, given two sequences $\{\alpha_k\}$ and $\{\beta_k\}$ of non-negative reals, we write with a slight abuse of notation, $\alpha_k = \mathcal{O}(\beta_k)$ if there exists a finite constant C such that $\alpha_k \leq C\beta_k$ for every $k \geq 1$.

2 Related works

Despite that BCD approaches have long been known, they have recently regained interest in the study of large-scale problems. They can be classified into two categories: exact and inexact algorithms. The former is characterized by solving each block subproblem to global optimality and convergence results often rely on convexity or relaxed per-block convexity assumptions on the objective function [8, 13, 39]. While in the latter, the block subproblems are solved approximately; thus, the BCG method is an example of inexact scheme.

The convergence properties of BCG have been extensively studied in the convex setting, and the corresponding theory is by now well established, see [92, 80] for review articles on the topic. Among all possible block selection strategies, randomized BCG has attracted the largest body of contributions due to the simplicity of the arguments employed in its convergence analysis. In particular, for randomized (accelerated) Coordinate Gradient (CG), Nesterov established in [67] nonasymptotic sublinear convergence rates in the convex case and linear convergence rates in the strongly convex setting. Following the work of Nesterov, sharper rates were derived in [56, 4, 68], by introducing different sampling techniques. In the nonsmooth setting, a first sublinear rate was established for an ℓ_1 regularized finite sum minimization problem in [79]. Later, the result of Nesterov was extended in [75] and further refined in [61] to Block Coordinate Proximal Gradient (BCPG) for the sum of a smooth and a nonsmooth, block-separable, convex function. Subsequent works have established convergence rates for randomized BCG algorithms in specialized convex settings, including parallelizable extensions [66] and formulations involving nonseparable objective functions [3].

Despite the analysis of cyclic BCG being more involved than that of randomized BCG, Luo and Tseng proved in [62] an asymptotic linear convergence rate for cyclic CG for strictly convex and twice differentiable functions. Saha and Tewari obtained in [76] the first nonasymptotic result with a sublinear rate in the convex case under an isotonicity assumption on the gradient. Later, Beck and Tetruashvili in [11] proved sublinear convergence of cyclic BCG for a general smooth and convex function. Their bounds were refined in [83] for the sum of a quadratic and a block-separable, nonsmooth term, and for general convex functions, employing second order information. A similar composite setting was also considered in [58]. Other relevant contributions focus on the quadratic case, see [54, 84, 91]. Moreover, cyclic accelerated BCPG was extended to the class of Minty variational inequalities with monotone Lipschitz operators, encompassing convex composite optimization as well as convex-concave min-max problems. By introducing a generalized Lipschitz condition with respect to a Mahalanobis norm, the authors achieved the significant improvement of reducing the dependence of the convergence rate on the number of blocks by a square-root factor [81]. To the best of our knowledge, the first accelerated BCPG method with adaptive step sizes was recently proposed by Wei et al. [89]. The authors considered the same class of Minty variational inequalities with monotone Lipschitz operators, where the step sizes are adaptively selected according to local curvature information. Their framework is also

parallelizable since it utilizes operator information delayed by a full cycle. Similar to the method proposed in this paper, their algorithm does not require prior knowledge of the Lipschitz constant nor the use of a line-search procedure. Nevertheless, their approach differs substantially from ours: it is not an OFFO algorithms, in fact the local estimates of the Lipschitz constant rely on explicit function evaluations. Moreover, their framework does not cover general nonconvex functions. Greedy selection rules have been shown to have better convergence rates than randomized rule when minimizing smooth strongly convex [70] as well as general convex functions [82]. On the other hand, the per-iteration computational cost of GS is usually higher than that of random strategies, since it requires the evaluation of the full gradient. Thus, random and cyclic rules are often preferred in practice unless greedy strategies can be cheaply evaluated at every iteration, see [82, 57, 70]. Nevertheless, a sublinear rate of convergence for the BCG method with GS-like block selection rules has been derived for convex problems [86, 23]. Another relevant contribution covering both greedy and cyclic rules is the block successive upper-bound minimization (BSUM) framework, which provides a unified convergence rate analysis for both the BCG and the BCPG methods for convex objectives [74, 47].

Coordinate descent in convex setting with nonasyntotic rates guarantees					
	Algorithm	Block rule	Nonsmooth term	Adaptive step	Additional assumptions
Nesterov [67]	CG/ACG	R	/	no	/
Tee, Sidford [56]	ACG	R	/	no	strongly convex f
Allen-Zhu, al. [4]	ACG	R	/	no	/
Nesterov, Stich [68]	ACG	R	/	no	/
Shalev-Shwartz, Tewari [79]	PCG	R	$\|\cdot\|_1$	no	f in a finite-sum form
Richtárik, Takáč [75]	BCPG	R	separable	no	/
Lu, Xiao [61]	BCPG/ABCG	R	separable	no	/
Saha, Tewari [76]	PCG	C	$\|\cdot\|_1$	no	isotonicity of ∇f
Beck, Tetruashvili [11]	BCG/ABCG	C	/	no	/
Sun, Hong [83]	CG	C	/	no	twice differentiable f
Sun, Ye [84]	CG	C	/	no	quadratic f
Lee, Wright [54]	CG	C	/	no	quadratic f
Li et al. [58]	BCPG	C	strongly convex	no	twice differentiable f
Song and Diakonikolas [81][(1)]	ABCPG	C	separable	no	/
Wei et al. [89][(1)]	ABCPG	C	separable	yes	/
Razaviyayn et al. [47]	BCPG[(1)]	C/Gr	separable	no	/
Tseng, Yun [86]	BCG	Gr	separable	no	/
Dhillon et al. [23]	PCG	Gr	separable	no	/

Table 1: Summary of contributions establishing nonasymptotic convergence rates for BCG and its variant in the convex setting. Notation in the first column: Coordinate Gradient (CG), Accelerated CG (ACG), Proximal CG (PCG), Block CG (BCG), Accelerated BCG (ABCG), Block Coordinate Proximal Gradient (BCPG); in the second column: Random (R), Cyclic (C), Greedy (Gr). The third columns shows if composite (smooth+nonsmooth) optimization problems of the form $f + h$ are considered and, if so, the type of nonsmooth term h. The last column contains additional assumptions considered in the paper.

Relevant contributions providing nonasymptotic convergence rates for BCG and related algorithms in the convex setting are summarized in Table 1. Moreover, several works analyze the more general setting of composite problems of the form $f + h$, where usually f is smooth and h is nonsmooth; thus we include them in the table specifying, in the third column, the additional assumptions for the nonsmooth term h. Due to space limitations, the table does not cover specialized settings such as the strongly convex or constrained, which are possibly separately analyzed

[(1)]Their framework analyzes general monotone variational inequalities; thus, covering some structured nonconvex problems such as convex-concave min-max optimization.

by some of the contribution. Similarly, since all the contributions also assume (block) Lipschitz continuity of the gradient or generalized variants we omit this information from Table 1.

The literature on the non-asymptotic convergence of BCG in the nonconvex setting is much more limited, while most existing results focus on asymptotic convergence, see [7, 60, 94, 74, 16, 95, 52]. Among the relevant contributions investigating nonasymptotic convergence rates in the nonconvex setting, Csiba and Richtárik in [21] proposed a unified analysis for an arbitrary block-selection rule, under the Polyak-Łojasiewicz (PL) or Weakly PL (WPL) properties, proving a linear convergence and a sublinear rate, respectively. For randomized BCG, Patrascu and Necoara in [71] established a sublinear convergence rate for BCPG in a more general setting, specifically when minimizing the sum of a nonconvex, smooth function and a convex, separable, and possibly nondifferentiable function. Sublinear convergence rates with high probability were also derived in [65] for (stochastic) random BCG for smooth but nonseparable functions. To our knowledge, for cyclic (proximal) BCG, the only nonasymptotic convergence guarantee in general nonconvex setting is given by Cai et al. in [17]. In particular, they analyze a composite minimization problem where the smooth part consists of a finite sum of functions and the possibly nonsmooth part is block-separable. Under a non restrictive Lipschitz condition on the gradient with respect to a Mahalanobis norm (implied by the global Lipschitz continuity of the gradient), they could prove sublinear convergence of cyclic BCPG. For greedy BCG and BCPG, Nutini et al. in [69] considered different GS-like block selection rules proving sublinear and linear convergence in the general nonconvex setting and under the PL condition on the objective function, respectively.

In all the contributions listed above, the step size in the BCG update is chosen either by assuming some knowledge of the (block) Lipschitz constant, which might not be the case in practical applications, or by using a line-search strategy.

BCG global nonasymptotic rates for nonconvex functions					
	Block rule	Nonsmooth extension	Adaptive step	Additional assumptions	Rate for smooth objective
Csiba, Richtárik [21]	R/Gr	✓	no	f is PL	$\mathbb{E}[f(x^k) - f^*] = \mathcal{O}(\Theta^k)$
Csiba, Richtárik [21]	R/Gr	✓	no	f is WPL	$\mathbb{E}[f(x^k) - f^*] = \mathcal{O}(\frac{1}{k})$
Patrascu, Necoara [71]	R	✓	no	/	$\min_{j=1,\dots,k} \mathbb{E}[\|\nabla f(x^j)\|^2] = \mathcal{O}\left(\frac{1}{k}\right)$
Necoara, Chorobura [65]	R	✗	no	/	$\min_{j=1,\dots,k} \|\nabla f(x^j)\|^2 = \mathcal{O}\left(\frac{1}{k}\right)$ w.h.p
Nutini et al. [69]	Gr	✓	no	f is PL	$f(x^k) - f^* = \mathcal{O}(\Theta^k)$
Nutini et al. [69]	Gr	✓	no	/	$\min_{j=1,\dots,k} \|\nabla f(x^j)\|^2 = \mathcal{O}\left(\frac{1}{k}\right)$
Cai et al. [17]	C	✓	no	f in finite sum form	$\min_{j=1,\dots,k} \|\nabla f(x^j)\|^2 = \mathcal{O}\left(\frac{1}{k}\right)$
This paper	R	✗	yes	/	$\mathbb{E}\left[\text{average}_{j=1,\dots,k} \|\nabla f(x^j)\|\right] = \mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$
This paper	C/Gr	✗	yes	/	$\text{average}_{j=1,\dots,k} \|\nabla f(x^j)\| = \mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$

Table 2: Summary of contributions establishing nonasymptotic convergence rates for the BCG for smooth and nonconvex functions. Notation in the first column: Random (R), Cyclic (C), Greedy (Gr). The fourth column contains additional assumptions made on the objective function with notation: Polyak–Łojasiewicz (PL), Weakly PL (WPL). All the contributions also assume (block) Lipschitz continuity of the gradient or other closely related assumptions which are not shown in this table. The last column contains the type of nonasymptotic convergence result, where f^* denotes the optimal function value and with high probability is abbreviated to w.h.p. The norm used in the rate is not specified and it may vary depending on the contribution; thus, we refer to the original references for more details.

Recently, Liu et al. showed in [59] the potential of combining BCG and adaptive step sizes. They proposed a block coordinate version of stochastic Adam but the convergence theory of their method has not been investigated. Moreover, a unified convergence theory for adaptive first-order methods has been very recently proposed in [38][(2)] in the related context of stochastic

[(2)] During the final preparation of this manuscript, we became aware of the recent preprint [38]. The two works were developed independently and in parallel. While they propose similar approaches, their purposes are substantially different: deterministic BCG for this paper and stochastic, parallel and distributed optimization for [38]. The two contributions were carried out without knowledge of each other. We cite it here for completeness.

asynchronous gradient methods with delayed operators. We summarized the contributions in the nonconvex setting in Table 2. For a more detailed discussion and comparison among convergence rates, see Section 4.1.

Other relevant contributions in the BCG literature succeeded in relaxing the hypothesis of block-Lipschitz continuity of the gradient by introducing weaker hypothesis, often based on Bregman distance [44, 29, 30, 45, 46]; or considered the more restrictive setting of alternate minimization, where the number of blocks is fixed at two [6, 15, 10, 24]. Analyzing these rather different settings is out of the scope of this work, which focuses on the smooth, block-Lipschitz continuous, and possibly nonconvex case. Therefore, these contributions do not appear in Table 1 and 2.

3 The bAdag framework

We proceed now by introducing the bAdag algorithm (Algorithm 3.1). Let $\{1, \dots, N\} = \bigcup_{\ell=1}^{d} b_\ell$ be a partition of the variables into blocks which we assume remains fixed within the iterations. Let also $g_k^{(\ell_k)}$ the block gradient defined in (1.3). We consider three block selection rules in our analysis:

- **GS**: pick the block b_{ℓ_k} with the largest gradient norm, corresponding to

$$\|g_k^{(\ell_k)}\| = \text{argmax}_{\ell=1,\dots,d} \|g_k^{(\ell)}\|.$$

- **C**: pick the block b_{ℓ_k} in cyclic order; thus, every d iterations, all the blocks are updated once.
- **UR**: pick the block b_{ℓ_k} uniformly at random, with a probability of $1/d$.

One can notice that the GS rule requires the evaluation of the full gradient at each iteration; thus, being in general more expensive than the UR and the cyclic rule. We are now ready to introduce the bAdag algorithm.

The definition of the adaptive step size used in (3.4) at step 3 of Algorithm 3.1 follows the same principle of the AdaGrad algorithm, with the crucial difference that only the weights in the selected block are updated. This means that the algorithm does not accumulate the entire history of the gradient but only the history of the gradient in the blocks selected by the BCD scheme. Indeed, using (3.4) and the notation in (1.3), one verifies that

$$w_{i,k} = \sqrt{\varsigma + \sum_{j=0}^{k} \left(g_{i,j}^{(\ell_j)}\right)^2}, \quad \text{and} \quad s_{i,k} = -\frac{g_{i,k}^{(\ell_k)}}{w_{i,k}}. \tag{3.7}$$

It is clear from (1.3) and (3.7) that the step $s_{i,k}$ is non-zero only in the selected block b_{ℓ_k}; thus, the update of the iterate x_k in (3.6) only affects the component in the block, and it is of the same form as the one in (1.2).

4 Convergence

We now present the nonasymptotic convergence analysis of bAdag. We develop a unified framework that covers GS, UR, C. We first characterize the descent property of the algorithm by bounding the difference in the function values of two consecutive iterates in Lemma 4.1. Then, we provide a unified strategy to upper bound the average norm of the block gradient, independently from the block selection rule considered, and finally we derive specialized bounds on the average norm of the full gradient for each block selection rule (Theorem 4.5).

Let us introduce standard assumptions in the nonasymptotic convergence analysis of first order BCG algorithms.

Algorithm 3.1: bAdag

Step 0: Initialization. A starting point x_0, a constant $\varsigma \in (0,1)$, and the initial weight vector $w_{i,-1} = \sqrt{\varsigma}$ for $i = 1, \ldots, N$, a fixed partition $\{1, \ldots, N\} = \bigcup_{\ell=1}^{d} b_\ell$. Set $k = 0$.

Step 1: Choose the block. Select the block $b_{\ell_k} \subseteq \{1, \ldots, N\}$ to be updated, according to the GS, UR or C rule.

Step 2: Gradient If not already computed in Step 1, compute the gradient $g_{i,k}$ for $i \in b_{\ell_k}$.

Step 3: Optimization weights. Compute

$$\begin{aligned} w_{i,k} &= \sqrt{(w_{i,k-1})^2 + g_{i,k}^2} \qquad (i \in b_{\ell_k}), \\ w_{i,k} &= w_{i,k-1} \qquad (i \in \{1, \ldots, N\} \backslash b_{\ell_k}). \end{aligned} \tag{3.4}$$

Step 4: Compute the step. Compute

$$\begin{aligned} s_{i,k} &= -\frac{g_{i,k}}{w_{i,k}} \qquad (i \in b_{\ell_k}). \\ s_{i,k} &= 0 \qquad (i \in \{1, \ldots, N\} \backslash b_{\ell_k}). \end{aligned} \tag{3.5}$$

Step 5: New iterate. Define

$$x_{k+1} = x_k + s_k, \tag{3.6}$$

increment k by one and return to Step 1.

AS.1: the objective function f is continuously differentiable;

AS.2: there exists a constant f_{low} such that, for all x, $f(x) \geq f_{\text{low}}$;

AS.3: given a partition of the components of the iterate in blocks $\{1, \ldots, N\} = \bigcup_{\ell=1}^{d} b_\ell$, the gradient g is block Lipschitz continuous with block Lipschitz constant $L_\ell \geq 0$, that is

$$\|g^{(\ell)}(x) - g^{(\ell)}(x + s^{(\ell)})\| \leq L_\ell \|s^{(\ell)}\| \quad \text{for every } \ell = 1, \ldots, d, \tag{4.8}$$

for all $x \in \mathbb{R}^N$; where $s^{(\ell)} \in \mathbb{R}^N$ denotes a vector whose support is contained in b_ℓ. It follows from AS.3 that

$$f(x + s^{(\ell)}) \leq f(x) + g(x)^T s^{(\ell)} + \frac{L_\ell}{2} ||s^{(\ell)}||^2. \tag{4.9}$$

We also define the Lipschitz continuity of the full gradient, which is only needed to prove convergence for cyclic bAdag.

AS.4: the gradient g is Lipschitz continuous with global Lipschitz constant $L \geq 0$:

$$\|g(x) - g(y)\| \leq L\|x - y\| \quad \text{for all } x, y \in \mathbb{R}^N.$$

Notice that AS.3 is implied by AS.4 by chosing $L_\ell = L$, for every $\ell = 1, \ldots, d$.

Moreover, the UR rule introduces randomness in the algorithm; thus, the sequence generated by bAdag is a random process. As a consequence, the convergence analysis for the bAdag with the UR rule is in expectation. Specifically, at iteration k, the expectation conditioned to knowing the

history of the past iterate $x_0, \dots, x_{k-1}$ will be denoted by the symbol $\mathbb{E}_k[\cdot]$. Moreover, when the index ℓ_k of the block to optimize at each iteration is chosen uniformly at random, it holds

$$\mathbb{E}_k[\|g_k^{(\ell_k)}\|] = \frac{1}{d}\sum_{\ell=1}^{d}\|g_k^{(\ell)}\|. \tag{4.10}$$

Furthermore, the block gradient $g_k^{(\ell_k)}$ satisfies $\mathbb{E}_k[g_k^{(\ell_k)}] = 1/d\sum_{\ell=1}^{d} g_k^{(\ell)} = 1/d\, g_k$, which means that the block gradient is an unbiased estimator of the true gradient, except for a constant factor d. Thus, the analysis of bAdag with the UR rule is partially inspired by that of the stochastic AdaGrad algorithm in [33]. Nevertheless, there are some crucial differences:

a) we do not consider any rescaling in our algorithm; therefore, the theory in [33] does not apply when the stochastic estimator of the gradient is $g_k^{(\ell_k)}$;

b) Contrary to [33], we do not assume boundedness of the gradient in our analysis.

We are now ready to state the descent lemma for the bAdag algorithm. The proof of the lemma is provided only for the UR rule since that for GS and C follows from the same idea by removing expected values. In fact, both the rules do not introduce any randomness in the algorithm; thus, all the quantities in expectation are deterministic.

Lemma 4.1 *Suppose that AS.1 - AS.3 hold. Then, according to the block selection rules, we have that, for all $j \geq 0$,*

$$\textbf{\textit{(GS/C) :}} \quad f(x_{j+1}) \leq f(x_j) - \sum_{i=1}^{N}\frac{(g_{i,j}^{(\ell_j)})^2}{w_{i,j}} + \frac{L_{\ell_j}}{2}\sum_{i=1}^{N}\frac{(g_{i,j}^{(\ell_j)})^2}{w_{i,j}^2}, \tag{4.11}$$

$$\textbf{\textit{(UR) :}} \quad \mathbb{E}_j\left[f(x_{j+1})\right] \leq f(x_j) - \mathbb{E}_j\left[\sum_{i=1}^{N}\frac{(g_{i,j}^{(\ell_j)})^2}{w_{i,j}}\right] + \frac{L_{\ell_j}}{2}\mathbb{E}_j\left[\sum_{i=1}^{N}\frac{(g_{i,j}^{(\ell_j)})^2}{w_{i,j}^2}\right]. \tag{4.12}$$

Proof. Using Assumptions AS.1 and AS.3, the definition of the step in (3.5), and the inequality in (4.9), we derive that

$$\begin{aligned}
\mathbb{E}_j\left[f(x_{j+1})\right] &\leq f(x_j) + \mathbb{E}_j\left[g_j^T s_j\right] + \frac{L_{\ell_j}}{2}\mathbb{E}_j\left[||s_j||^2\right] \\
&= f(x_j) + \mathbb{E}_j\left[\sum_{i\in b_{\ell_j}} g_{i,j}s_{i,j}\right] + \frac{L_{\ell_j}}{2}\mathbb{E}_j\left[\sum_{i\in b_{\ell_j}} s_{i,j}^2\right] \\
&= f(x_j) - \mathbb{E}_j\left[\sum_{i\in b_{\ell_j}}\frac{g_{i,j}^2}{w_{i,j}}\right] + \frac{L_{\ell_j}}{2}\mathbb{E}_j\left[\sum_{i\in b_{\ell_j}}\frac{g_{i,j}^2}{w_{i,j}^2}\right] \\
&= f(x_j) - \mathbb{E}_j\left[\sum_{i=1}^{N}\frac{(g_{i,j}^{(\ell_j)})^2}{w_{i,j}}\right] + \frac{L_{\ell_j}}{2}\mathbb{E}_j\left[\sum_{i=1}^{N}\frac{(g_{i,j}^{(\ell_j)})^2}{w_{i,j}^2}\right],
\end{aligned} \tag{4.13}$$

which concludes the lemma. □

We introduce two technical lemmas that are crucial to prove our main convergence result.

Lemma 4.2 ([12]) *Suppose that, for $u > 0$ and $\gamma_1, \gamma_2 > 0$,*

$$\gamma_1 u \leq \gamma_2 \log(u) + \gamma_3.$$

Then,

$$u \leq \frac{2\gamma_3}{\gamma_1} + \frac{2\gamma_2}{\gamma_1}\left(\log\left(\frac{2\gamma_2}{\gamma_1}\right) - 1\right).$$

The next lemma is from [88, 22, 93] and we use it to bound the contribution of the quadratically decaying term in the descent lemma of the bAdag algorithm.

Lemma 4.3 *Let $\{c_k\}_{k\geq 0}$ be a non-negative sequence and $\xi > 0$. Then*

$$\sum_{j=0}^{k} \frac{c_j}{(\xi + \sum_{q=0}^{j} c_q)} \leq \log\left(1 + \frac{1}{\xi}\sum_{j=0}^{k} c_j\right).$$

The proof of the lemma can be found in [32, Lemma 3.1].

The following lemma aims at providing a lower bound for the first summation in (4.11) and (4.12) which linearly decreases with the weights $w_{i,j}$.

Lemma 4.4 *Assume* bAdag *is applied to problem* (1.1)*, then*

$$\sqrt{\varsigma}\sqrt{1 + \frac{1}{\varsigma}\sum_{j=0}^{k} \|g_j^{(\ell_j)}\|^2} - \sqrt{\varsigma} \leq \sum_{j=0}^{k}\sum_{i=1}^{N} \frac{(g_{i,j}^{(\ell_j)})^2}{w_{i,j}}.$$

Proof. Since the sequence of $w_{i,j}$ in (3.7) is increasing in j, $w_{i,j} \leq w_{i,k}$ for every $j = 0, \ldots, k$. Then

$$w_{i,j} \leq w_{i,k} = \sqrt{\varsigma + \sum_{q=0}^{k} (g_{i,q}^{(\ell_q)})^2} \leq \sqrt{\varsigma + \sum_{q=0}^{k} \|g_q^{(\ell_q)}\|^2}, \tag{4.14}$$

from which we obtain

$$\sum_{j=0}^{k} \frac{\|g_j^{(\ell_j)}\|^2}{\sqrt{\varsigma + \sum_{q=0}^{k} \|g_q^{(\ell_q)}\|^2}} \leq \sum_{j=0}^{k}\sum_{i=1}^{N} \frac{(g_{i,j}^{(\ell_j)})^2}{w_{i,j}}, \tag{4.15}$$

Furthermore, we have

$$\begin{aligned}
\sum_{j=0}^{k} \frac{\|g_j^{(\ell_j)}\|^2}{\sqrt{\varsigma + \sum_{q=0}^{k} \|g_q^{(\ell_q)}\|^2}} =& \frac{\sum_{j=0}^{k} \|g_j^{(\ell_j)}\|^2}{\sqrt{\varsigma + \sum_{q=0}^{k} \|g_q^{(\ell_q)}\|^2}} + \frac{\varsigma}{\sqrt{\varsigma + \sum_{q=0}^{k} \|g_q^{(\ell_q)}\|^2}} - \frac{\varsigma}{\sqrt{\varsigma + \sum_{q=0}^{k} \|g_q^{(\ell_q)}\|^2}} \\
=& \sqrt{\varsigma + \sum_{j=0}^{k} \|g_j^{(\ell_j)}\|^2} - \frac{\varsigma}{\sqrt{\varsigma + \sum_{q=0}^{k} \|g_q^{(\ell_q)}\|^2}} \\
\geq& \sqrt{\varsigma + \sum_{j=0}^{k} \|g_j^{(\ell_j)}\|^2} - \sqrt{\varsigma}, \\
=& \sqrt{\varsigma} \cdot \sqrt{1 + \frac{1}{\varsigma}\sum_{j=0}^{k} \|g_j^{(\ell_j)}\|^2} - \sqrt{\varsigma},
\end{aligned}$$

which, combined with (4.15), concludes the proof. □

We are now ready to state our main result, characterizing the nonsymptotic rate of convergence of bAdag.

Theorem 4.5 *Suppose that AS.1–AS.3 hold and that* bAdag *is applied to problem (1.1). Let $\varsigma > 0$, d the number of blocks, and $\Gamma_0 \stackrel{\text{def}}{=} f(x_0) - f_{\text{low}}$. Then,*

- ***GS:***

$$\underset{j\in\{0,\ldots,k\}}{\text{average}} \|g_j\| \leq \sqrt{d}\frac{\Theta}{\sqrt{k+1}}. \tag{4.16}$$

- ***C:*** *if, in addition, AS.4 holds, then*

$$\operatorname*{average}_{t\in\{0,\dots,T\}} \|g_{td}\| \leq \sqrt{2\left(1+d\frac{L^2}{\varsigma^2}\right)}\frac{\Theta}{\sqrt{T+1}}. \tag{4.17}$$

- ***UR:***

$$\mathbb{E}\left[\operatorname*{average}_{j\in\{0,\dots,k\}} \|g_j\|\right] \leq d\frac{\Theta}{\sqrt{k+1}}, \tag{4.18}$$

with

$$\Theta \stackrel{\text{def}}{=} 2\Gamma_0 + 2\sqrt{\varsigma} + 2N\bar{L}\left[\log\left(\frac{2N\bar{L}}{\sqrt{\varsigma}}\right)-1\right], \tag{4.19}$$

where $\bar{L} = \max_{\ell=1,\dots,d} L_\ell$.

Proof. Since the descent lemma for the GS/C (deterministic) and UR rule (stochastic) differ, we analyze separately these two cases. However, the proof follows the same strategy, with some more technical steps needed in the UR selection case since expected values are involved.

GS/C: Starting from the descent lemma in (4.11) and summing for $j = 0, \dots, k$,

$$f(x_0) - f(x_{k+1}) \geq \sum_{j=0}^{k}\sum_{i=1}^{N}\frac{(g_{i,j}^{(\ell_j)})^2}{w_{i,j}} - \frac{\bar{L}}{2}\sum_{j=0}^{k}\sum_{i=1}^{N}\frac{(g_{i,j}^{(\ell_j)})^2}{w_{i,j}^2},$$

where $\bar{L} = \max_{\ell=1,\dots,d} L_\ell$. Rearranging the terms and adding the definition of Γ_0,

$$\sum_{j=0}^{k}\sum_{i=1}^{N}\frac{(g_{i,j}^{(\ell_j)})^2}{w_{i,j}} \leq \Gamma_0 + \frac{\bar{L}}{2}\sum_{j=0}^{k}\sum_{i=1}^{N}\frac{(g_{i,j}^{(\ell_j)})^2}{w_{i,j}^2}. \tag{4.20}$$

We first focus on computing an upper bound of the right hand side. In particular, we use the reformulation of $w_{i,j}$ in (3.7) and Lemma 4.3 with $c_j = (g_{i,j}^{(\ell_j)})^2$ and $\xi = \zeta$, yielding

$$\begin{aligned}\sum_{i=1}^{N}\sum_{j=0}^{k}\frac{(g_{i,j}^{(\ell_j)})^2}{w_{i,j}^2} &= \sum_{i=1}^{N}\sum_{j=0}^{k}\frac{(g_{i,j}^{(\ell_j)})^2}{\varsigma + \sum_{q=0}^{j}(g_{i,q}^{(\ell_q)})^2} \leq \sum_{i=1}^{N}\log\left(1+\frac{1}{\varsigma}\sum_{j=0}^{k}(g_{i,j}^{(\ell_j)})^2\right)\\ &\leq \sum_{i=1}^{N}\log\left(1+\frac{1}{\varsigma}\sum_{j=0}^{k}\|g_j^{(\ell_j)}\|^2\right) = 2N\log\left(\sqrt{1+\frac{1}{\varsigma}\sum_{j=0}^{k}\|g_j^{(\ell_j)}\|^2}\right).\end{aligned} \tag{4.21}$$

Combining (4.20) and (4.21), gives

$$\sum_{j=0}^{k}\sum_{i=1}^{N}\frac{(g_{i,j}^{(\ell_j)})^2}{w_{i,j}} \leq \Gamma_0 + N\bar{L}\log\left(\sqrt{1+\frac{1}{\varsigma}\sum_{j=0}^{k}\|g_j^{(\ell_j)}\|^2}\right). \tag{4.22}$$

We now focus on the left hand side introducing the lower bound provided by Lemma 4.4, which gives:

$$\sqrt{\varsigma}\sqrt{1+\frac{1}{\varsigma}\sum_{j=0}^{k}\|g_j^{(\ell_j)}\|^2} \leq \sqrt{\varsigma} + \Gamma_0 + N\bar{L}\log\left(\sqrt{1+\frac{1}{\varsigma}\sum_{j=0}^{k}\|g_j^{(\ell_j)}\|^2}\right).$$

We can now apply Lemma 4.2 with

$$u = \sqrt{1+\frac{1}{\varsigma}\sum_{j=0}^{k}\|g_j^{(\ell_j)}\|^2}, \quad \gamma_1 = \sqrt{\varsigma}, \quad \gamma_2 = N\bar{L}, \quad \gamma_3 = \sqrt{\varsigma} + \Gamma_0,$$

obtaining

$$\sqrt{1+\frac{1}{\varsigma}\sum_{j=0}^{k}\|g_j^{(\ell_j)}\|^2} \leq \frac{2(\sqrt{\varsigma}+\Gamma_0)}{\sqrt{\varsigma}} + \frac{2N\bar{L}}{\sqrt{\varsigma}}\left[\log\left(\frac{2N\bar{L}}{\sqrt{\varsigma}}\right)-1\right]. \tag{4.23}$$

Simple calculations yields

$$\sqrt{\sum_{j=0}^{k}\|g_j^{(\ell_j)}\|^2} \leq \sqrt{\varsigma}\sqrt{1+\frac{1}{\varsigma}\sum_{j=0}^{k}\|g_j^{(\ell_j)}\|^2} \leq 2(\sqrt{\varsigma}+\Gamma_0) + 2N\bar{L}\left[\log\left(\frac{2N\bar{L}}{\sqrt{\varsigma}}\right)-1\right]. \tag{4.24}$$

Let $\Theta = 2\sqrt{\varsigma} + 2\Gamma_0 + 2N\bar{L}\left[\log\left(\frac{2N\bar{L}}{\sqrt{\varsigma}}\right)-1\right]$, dividing by $\sqrt{k+1}$, and using the inequality

$$\frac{1}{k+1}\sum_{j=0}^{k}\|g_j^{(\ell_j)}\| \leq \frac{1}{\sqrt{k+1}}\sqrt{\sum_{j=0}^{k}\|g_j^{(\ell_j)}\|^2}, \tag{4.25}$$

we have

$$\underset{j\in\{0,\dots,k\}}{\text{average}}\|g_j^{(\ell_j)}\| \leq \sqrt{\underset{j\in\{0,\dots,k\}}{\text{average}}\|g_j^{(\ell_j)}\|^2} \leq \frac{\Theta}{\sqrt{k+1}}. \tag{4.26}$$

The next step is to derive a bound for the full gradient norm rather than the norm of the block gradient. This bound depends on the block selection criteria. Thus, we now consider two different cases corresponding to the GS rule or the Cyclic rule.

Case 1: GS. At every iteration, the block b_{ℓ_j} corresponding to the largest gradient norm is chosen; thus, it trivially follows that for every j

$$\|g_j\| = \sqrt{\sum_{\ell=1}^{d}\|g_j^{(\ell)}\|^2} \leq \sqrt{d\|g_j^{(\ell_j)}\|^2} = \sqrt{d}\|g_j^{(\ell_j)}\|. \tag{4.27}$$

Thus, from (4.26),

$$\underset{j\in\{0,\dots,k\}}{\text{average}}\|g_j\| \leq \sqrt{d}\ \underset{j\in\{0,\dots,k\}}{\text{average}}\|g_j^{(\ell_j)}\| \leq \frac{\sqrt{d}\Theta}{k+1}, \tag{4.28}$$

hence obtaining (4.16).

Case 2: C. Let us first observe that in the cyclic rule, every d iterations, all the blocks are updated the same number of times. In order to simplify the passages that follow, we reparametrize the iteration counter by grouping iterations into groups of d elements. Equivalently, we write $k = Td+\ell$, where $\ell = 0,\dots,d-1$. Consequently, all the components in x_{td}, for $t = 0,1,\dots,T$, have been updated the same number of times. Following the same reasoning presented in [11, Lemma 3.3], it holds the following

$$x_{td+\ell} = x_{td} - \sum_{s=1}^{\ell-1}\frac{g_{td+s}^{(s)}}{w_{td+s}}. \tag{4.29}$$

Using (4.29) and the Lipschitz continuity of f in AS.4, the fact that $\varsigma \leq w_{td+\ell}$ for every t and ℓ, we have

$$\begin{aligned}
\|g_{td}^{(\ell)}\|^2 &\leq \left(\|g_{td}^{(\ell)} - g_{td+\ell}^{(\ell)}\| + \|g_{td+\ell}^{(\ell)}\|\right)^2 \leq 2\|g_{td}^{(\ell)} - g_{td+\ell}^{(\ell)}\|^2 + 2\|g_{td+\ell}^{(\ell)}\|^2 \\
&\leq 2\|g_{td} - g_{td+\ell}\|^2 + 2\|g_{td+\ell}^{(\ell)}\|^2 \leq 2L^2\|x_{td} - x_{td+\ell}\|^2 + 2\|g_{td+\ell}^{(\ell)}\|^2 \\
&\leq 2L^2\left\|\sum_{s=1}^{\ell-1}\frac{g_{td+s}^{(s)}}{w_{td+s}}\right\|^2 + 2\|g_{td+\ell}^{(\ell)}\|^2 \leq 2L^2\sum_{s=1}^{\ell-1}\left\|\frac{g_{td+s}^{(s)}}{w_{td+s}}\right\|^2 + 2\|g_{td+\ell}^{(\ell)}\|^2 \\
&\leq \frac{2L^2}{\varsigma^2}\sum_{s=1}^{\ell-1}\|g_{td+s}^{(s)}\|^2 + 2\|g_{td+\ell}^{(\ell)}\|^2.
\end{aligned} \tag{4.30}$$

Then, summing $\ell = 1, \dots, d$ and using (4.30), we have

$$\begin{aligned} \|g_{td}\|^2 = \sum_{\ell=1}^{d} \|g_{td}^{(\ell)}\|^2 &\leq 2 \sum_{\ell=1}^{d} \left(1 + (d-\ell)\frac{L^2}{\varsigma^2}\right) \|g_{td+\ell}^{(\ell)}\|^2. \\ &\leq 2\left(1 + d\frac{L^2}{\varsigma^2}\right) \sum_{\ell=1}^{d} \|g_{td+\ell}^{(\ell)}\|^2. \end{aligned} \tag{4.31}$$

Summing over the number of cycles $t = 0, 1, \dots, T$, and since summing both for $t = 0, 1, \dots, T$ and for $\ell = 0, 1, \dots, d$ is equivalent to summing for $j = 0, 1, \dots, k$, then

$$\sum_{t=0}^{T} \|g_{td}\|^2 \leq 2\left(1 + d\frac{L^2}{\varsigma^2}\right) \sum_{t=0}^{T} \sum_{\ell=1}^{d} \|g_{td+\ell}^{(\ell)}\|^2 = 2\left(1 + d\frac{L^2}{\varsigma^2}\right) \sum_{j=0}^{k} \|g_j^{(\ell_j)}\|^2, \tag{4.32}$$

which, taking the square root and using (4.25), gives

$$\frac{1}{T+1} \sum_{t=0}^{T} \|g_{td}\| \leq \frac{1}{\sqrt{T+1}} \sqrt{\sum_{t=0}^{T} \|g_{td}\|^2} \leq \frac{1}{\sqrt{T+1}} \sqrt{2\left(1 + d\frac{L^2}{\varsigma^2}\right) \sum_{j=0}^{k} \|g_j^{(\ell_j)}\|^2}. \tag{4.33}$$

Finally, combining (4.33) with (4.24), we obtain

$$\underset{t=0,\dots,T}{\text{average}} \|g_{td}\| \leq \sqrt{2\left(1 + d\frac{L^2}{\varsigma^2}\right)} \cdot \frac{\Theta}{\sqrt{T+1}}, \tag{4.34}$$

which is the thesis in (4.17).

UR: We now prove the rate of convergence of the bAdag algorithm when the UR rule is used. Starting from (4.12) and AS.3, summing for $j = 0, \dots, k$, taking the expectation

$$f(x_0) - \mathbb{E}\left[\mathbb{E}_k\left[f(x_{k+1})\right]\right] \geq \mathbb{E}\left[\sum_{j=0}^{k} \mathbb{E}_j\left[\sum_{i=1}^{N} \frac{(g_{i,j}^{(\ell_j)})^2}{w_{i,j}}\right]\right] - \frac{\bar{L}}{2}\mathbb{E}\left[\sum_{j=0}^{k} \mathbb{E}_j\left[\sum_{i=1}^{N} \frac{(g_{i,j}^{(\ell_j)})^2}{w_{i,j}^2}\right]\right], \tag{4.35}$$

using the tower property, equation (4.35) simplifies to

$$\mathbb{E}\left[\sum_{j=0}^{k} \sum_{i=1}^{N} \frac{(g_{i,j}^{(\ell_j)})^2}{w_{i,j}}\right] \leq f(x_0) - f_{\text{low}} + \frac{\bar{L}}{2}\mathbb{E}\left[\sum_{j=0}^{k} \sum_{i=1}^{N} \frac{(g_{i,j}^{(\ell_j)})^2}{w_{i,j}^2}\right]. \tag{4.36}$$

We provide an upper bound for the right hand side in (4.36) by using the reformulation of $w_{i,j}$ in (3.7) and Lemma 4.3 with $c_j = (g_{i,j}^{(\ell_j)})^2$. This bound is derived similarly to the one in (4.21), that is

$$\begin{aligned} \mathbb{E}\left[\sum_{j=0}^{k} \sum_{i=1}^{N} \frac{(g_{i,j}^{(\ell_j)})^2}{w_{i,j}}\right] &\leq \Gamma_0 + \frac{N\bar{L}}{2}\mathbb{E}\left[\log\left(1 + \frac{1}{\varsigma}\sum_{j=0}^{k} \|g_j^{(\ell_j)}\|^2\right)\right], \\ &= \Gamma_0 + N\bar{L}\mathbb{E}\left[\log\left(\sqrt{1 + \frac{1}{\varsigma}\sum_{j=0}^{k} \|g_j^{(\ell_j)}\|^2}\right)\right], \\ &\leq \Gamma_0 + N\bar{L}\log\left(\mathbb{E}\left[\sqrt{1 + \frac{1}{\varsigma}\sum_{j=0}^{k} \|g_j^{(\ell_j)}\|^2}\right]\right), \end{aligned} \tag{4.37}$$

where we used the concavity of the logarithm and Jensen inequality to derive the last bound. Using Lemma 4.4 combined with (4.37),

$$\sqrt{\varsigma}\mathbb{E}\left[\sqrt{1+\frac{1}{\varsigma}\sum_{j=0}^{k}\|g_j^{(\ell_j)}\|^2}\right] \leq \sqrt{\varsigma}+\Gamma_0+N\bar{L}\log\left(\mathbb{E}\left[\sqrt{1+\frac{1}{\varsigma}\sum_{j=0}^{k}\|g_j^{(\ell_j)}\|^2}\right]\right).$$

Similarly to the GS and cyclic case we can apply Lemma 4.2 with

$$u=\mathbb{E}\left[\sqrt{1+\frac{1}{\varsigma}\sum_{j=0}^{k}\|g_j^{(\ell_j)}\|^2}\right], \quad \gamma_1=\sqrt{\varsigma}, \quad \gamma_2=N\bar{L}, \quad \gamma_3=\sqrt{\varsigma}+\Gamma_0,$$

leading to

$$\mathbb{E}\left[\sqrt{1+\frac{1}{\varsigma}\sum_{j=0}^{k}\|g_j^{(\ell_j)}\|^2}\right] \leq \frac{2(\sqrt{\varsigma}+\Gamma_0)}{\sqrt{\varsigma}}+\frac{2N\bar{L}}{\sqrt{\varsigma}}\left[\log\left(\frac{2N\bar{L}}{\sqrt{\varsigma}}\right)-1\right], \tag{4.38}$$

thus resulting in

$$\mathbb{E}\left[\sqrt{\sum_{j=0}^{k}\|g_j^{(\ell_j)}\|^2}\right] \leq \sqrt{\varsigma}\mathbb{E}\left[\sqrt{1+\frac{1}{\varsigma}\sum_{j=0}^{k}\|g_j^{(\ell_j)}\|^2}\right] \leq 2(\sqrt{\varsigma}+\Gamma_0)+2N\bar{L}\left[\log\left(\frac{2N\bar{L}}{\sqrt{\varsigma}}\right)-1\right]=\Theta. \tag{4.39}$$

Using (4.25) again

$$\mathbb{E}\left[\operatorname*{average}_{j\in\{0,\dots,k\}}\|g_j^{(\ell_j)}\|\right] \leq \mathbb{E}\left[\sqrt{\operatorname*{average}_{j\in\{0,\dots,k\}}\|g_j^{(\ell_j)}\|^2}\right] \leq \frac{\Theta}{\sqrt{k+1}}. \tag{4.40}$$

Now we need to derive a bound for the norm of the whole gradient g_j. Therefore, using (4.10) and the tower property, we have

$$\begin{aligned}\mathbb{E}\left[\operatorname*{average}_{j\in\{0,\dots,k\}}\|g_j^{(\ell_j)}\|\right] &= \frac{1}{k+1}\sum_{j=0}^{k}\mathbb{E}[\mathbb{E}_j[\|g_j^{(\ell_j)}\|]] = \frac{1}{d(k+1)}\sum_{j=0}^{k}\mathbb{E}\left[\sum_{\ell=1}^{d}\|g_j^{(\ell)}\|\right] \\ &\geq \frac{1}{d(k+1)}\sum_{j=0}^{k}\mathbb{E}\left[\left\|\sum_{\ell=1}^{d}g_j^{(\ell)}\right\|\right] = \frac{1}{d(k+1)}\sum_{j=0}^{k}\mathbb{E}\left[\|g_j\|\right] = \frac{1}{d}\mathbb{E}\left[\operatorname*{average}_{j\in\{0,\dots,k\}}\|g_j\|\right]\end{aligned} \tag{4.41}$$

Finally, combining (4.40) with (4.41), we obtain (4.18), that is,

$$\mathbb{E}\left[\operatorname*{average}_{j\in\{0,\dots,k\}}\|g_j\|\right] \leq \frac{d\Theta}{\sqrt{k+1}}.$$

hence concluding the proof. □

Let us comment the convergence of bAdag.

1. Similarly to the original AdaGrad algorithm, the nonmonotone behavior of bAdag can be deduced from Lemma 4.1. In fact, if the second summation in (4.11) or (4.12) is larger than the first, then the function increases from iteration j to iteration $j+1$. However, since the second term quadratically decreases with the optimization weights $w_{i,j}$, while the first term depends linearly on the weights, we expect the first to dominate the second in later stages of

the algorithm. This behavior is consistently different from the majority of BCG methods in the literature, where constant step sizes or line-search are used. In fact, they usually imply a sufficient decrease of the function and result in monotonic algorithms. The nonmonotone behavior makes bAdag very flexible but it also results in weaker convergence guarantees, see the discussion below for more details.

2. While both the rates for random and greedy BCG depend on the total number of iterations k, the rate of cyclic BCG is given in terms of full number of cycles T after which all the variables are updated. One can deduce that in a situation where the number of blocks d is large, T grows slower than k. Thus, our result for cyclic BCD is weaker than those for random and greedy BCG. Nevertheless, proving convergence rates depending on the full cycles T is common practice when the cyclic block selection rule is used, see for example [11, 17].

3. Our bounds have an explicit dependence on the number of blocks d. Ideally, one would like to remove it since it impacts the comparison between BCG and standard gradient method in favor of non-block approach. One possible strategy in random BCG, it has been shown that choosing the block according to a probability p_ℓ depending on the block Lipschitz constant L_ℓ achieves the goal [67, 4, 68]. However, in our setting, the Lipschitz constants are considered to be unknown. Alternatively, on might consider introducing alternative Lipschitz condition such as that proposed in [81, 17, 89] to achieve this goal for the C rule. We regard exploring this possibility in the bAdag framework as future research.

4. The constant Θ that appears in our bound also depends on the variable dimension N. This dependency is due to the coordinatewise structure of bAdag which, at each iteration, adaptively chooses a different step size for the update of each component in the selected block. In particular, the constant N comes from the strategy we used in (4.21) to bound the sum of quadratic terms in the descent lemma. We believe that the factor N in the estimate might be improved, at least to the size of the largest block, but it is not clear how to achieve this goal.

4.1 Comparison with previous works

The first difference between this work and those closely related in Table 2 is that we provide bounds on the average gradient norm (ergodic rate) that are slightly stronger than those in the minimum gradient norm typically established in the contributions investigating similar smooth and nonconvex settings. Nevertheless, convergence rates expressed in terms of the minimum or the average norm are of a similar nature and constitute a standard technique in nonconvex optimization and ergodic results are widely used in the analysis of AdaGrad-type algorithms, see [34, 32, 33, 73]. We also express our bounds in term of the gradient norm instead of the gradient norm square as done in the majority of the contributions in Table 2. To favor the comparison between the convergence rates analyzed in this section, we rewrite all the bounds in terms of the gradient norms.

Moreover, no sufficient decrease is imposed in our setting, and the block Lipschitz constants are assumed to be unknown, as it occurs in several practical applications. This yields larger constants in the convergence rate for bAdag with respect to BCG where the step size is constant or computed by a line-search. Nevertheless, bAdag remains more flexible since it does not use the block-Lipschitz constants or any function evaluations to choose the step size at each iteration.

We now compare the rate of bAdag with those in the literature in more detail. Starting from randomized BCG, our result is similar to that proposed by Patrascu and Necoara [71, Corollary 1] for constant step size BCG, that is,

$$\min_{j\in\{0,\dots,k\}} \|g_j\|_L^* \leq \sqrt{\frac{2d(f(x^0)-f^*)}{k+1}}, \quad \text{where} \quad \|g_j\|_L^* = \sqrt{\sum_{\ell=1}^{d} \frac{\|g_j^{(\ell)}\|^2}{L_\ell}}, \tag{4.42}$$

and f^* denotes the optimal function value. Their constant in the bound is smaller than ours, and they prove a $\sqrt{d}$ dependence on the number of blocks d, while our rate involves a larger factor d. For greedy BCG, Nutini and co-authors [70] consider a similar setting to the one analyzed in this paper. Their method uses an update, referred to as matrix update, of the form $x^{k+1} = x^k - H_{\ell_k}^{-1} g_k^{(\ell_k)}$, where H_k is a positive definite matrix that represents an upper bound for the Hessian of f in the twice differentiable case. However, constructing H_{ℓ_k} is not trivial in general and is problem dependent, making this approach less flexible. Nevertheless, convergence is established as:

$$\min_{j\in\{0,\dots,k\}} \max_{\ell=1,\dots,d} \|g_j^\ell\|_{H_\ell^{-1}} \leq \sqrt{2\frac{f(x^0)-f^*}{k+1}}, \quad \text{where} \quad \|\cdot\|_H = \sqrt{\langle H\cdot,\cdot\rangle}. \tag{4.43}$$

Despite considering the standard Euclidean norm, the bound in this paper is similar to the one in (4.43). In fact, both the bounds present a linear dependence on the number of blocks d and using the fact that $\|g_j\|^2 \leq d \max_{\ell=1,\dots,d} \|g_j^\ell\|^2$ we can deduce from (4.43) a bound on the full gradient norm, similar to the one proposed in Theorem 4.5. Nevertheless, the constant in our bound still remains larger since it involves an additional term due to the nonmonotone behavior of the algorithm.

In addition, for cyclic BCG, the contribution that is closer to our nonconvex setting is that by Cai et al. [17]. Specifically, they analyze a composite (smooth + nonsmooth) minimization problem, where the smooth term is assumed to be a finite sum of functions and the nonsmooth term is convex and separable. Therefore, their convergence results are given in terms of the distance of the subgradient from zero. We readapt it here to the smooth case, obtaining:

$$\min_{j\in\{0,\dots,k\}} \|g_j\| \leq \sqrt{4(1+\hat{L})\frac{(f(x^0)-f^*)}{T+1}},$$

where $\hat{L}$ is a generalized notion of a global Lipschitz constant with respect to a Mahalanobis norm; we refer to [17] for more details. Contrary to our bound, the constant of Cai et al. has the advantage of being independent of the number of blocks. Nevertheless, when only smooth functions are considered, their finite sum setting is more restrictive than ours which includes general convex functions. We also mention that the dependence on the number of blocks d in our bound is similar to that proposed by Beck and Tetruashvili in [11] for convex, smooth functions.

Finally, the bAdag algorithm includes the original AdaGrad as a special case when the number of blocks d is equal to 1. In fact, for greedy and random BCG, setting $d = 1$ in the convergence rates we recover a similar bound to that of the AdaGrad-like method presented in [12]. For cyclic BCG, our result is weaker since for $d = 1$, our bound has an additional factor $1 + L/\varsigma$ which does not appear in the original method. The same observation is discussed in [11], comparing their cyclic BCG with the standard gradient method in the convex case.

5 Extension to box-constrained problems

We now extend bAdag to box constrained optimization. Thus, we consider the following problem:

$$\min_x f(x) \quad \text{subject to} \quad x \in C = \{x \in \mathbb{R}^N \mid a_i \leq x_i \leq u_i\}, \tag{5.44}$$

where $a_i, u_i \in \mathbb{R} \cup \{-\infty; \infty\}$. Therefore, when $a_i = -\infty$ and $u_i = \infty$, the unconstrained setting is recovered as a special case. We also denote P_C the projection into the set C, which is easy to compute since $[P_C(x)]_i = P_{[a_i,u_i]}(x_i) = \max(a_i, \min(u_i, x_i))$. In what follows, we adapt the bAdag algorithm to this more general setting, extending the idea of Bellavia et al. in [12], where they consider stochastic AdaGrad for box-constrained problems.

Let us introduce the box-constrained version of bAdag in Algorithm 5.1.

Algorithm 5.1: box-constrained bAdag

Step 0: Initialization. Bound constrained vectors $a, u \in \mathbb{R}^N$, a starting point $x_0 \in C$, a constant $\varsigma \in (0,1)$, and the initial weight vector $w_{i,-1} = \sqrt{\varsigma}$ for $i = 1, \ldots, N$, a fixed partition $\{1, \ldots, N\} = \bigcup_{\ell=1}^{d} b_\ell$. Set $k = 0$.

Step 1: Choose the block. Select the block $b_{\ell_k} \subseteq \{1, \ldots, N\}$ to update according to the GS, UR or C rule.

Step 2: Gradient If not already computed in Step 1, compute the gradient $g_{i,k}$ for $i \in b_{\ell_k}$.

Step 3: Measure of optimality. Compute

$$\begin{aligned} v_{i,k} &= P_{[a_i,u_i]}(x_{i,k} - g_{i,k}) - x_{i,k} \qquad (i \in b_{\ell_k}), \\ v_{i,k} &= 0 \qquad (i \in \{1, \ldots, N\} \backslash b_{\ell_k}). \end{aligned}$$

Step 4: Optimization weights. Compute

$$\begin{aligned} \eta_{i,k} &= \sqrt{(\eta_{i,k-1})^2 + v_{i,k}^2} \qquad (i \in b_{\ell_k}), \\ \eta_{i,k} &= \eta_{i,k-1} \quad \text{and} \qquad (i \in \{1, \ldots, N\} \backslash b_{\ell_k}). \end{aligned} \tag{5.45}$$

Step 6: Compute the step. Set

$$\begin{aligned} s_{i,k} &= \frac{v_{i,k}}{\max(1, \eta_{i,k})} \qquad (i \in b_{\ell_k}), \\ s_{i,k} &= 0 \qquad (i \in \{1, \ldots, N\} \backslash b_{\ell_k}). \end{aligned} \tag{5.46}$$

Step 7: New iterate. Define

$$x_{k+1} = x_k + s_k,$$

increment k by one and return to Step 1.

Let us first introduce the direction we use for the update of the i-th component of the kth-iterate:

$$v_{i,k} = P_{[a_i,u_i]}(x_{i,k} - g_{i,k}) - x_{i,k}, \quad \text{for} \quad i \in b_{\ell_k}.$$

The vector v_k is also used to update the new adaptive weights η_k of the algorithm. In fact, $\|v_k\|$ is a standard optimality measure in constrained, first order optimization, see [20]. Of course, if there are no constraints ($C = \mathbb{R}^N$), then $\|v_k\| = \|g_k\|$. In particular, this change in the optimality measure also affects the GS-type block selection rule in the box-constrained case, which is still denoted Gauss-Southwell for simplicity, and, at iteration k, selects the block b_{ℓ_k} associated to the largest norm $\|v_k^{(\ell)}\|$ for $\ell = 1, \dots, d$.

We observe that the updates in Algorithm 5.1 remain feasible within the iterations. We show it by distinguishing between two cases based on the value of $s_{i,k}$ in (5.46). If $s_{i,k} = v_{i,k}$, it holds for every $i \in b_{\ell_k}$ that $x_{i,k+1} = P_{[a_i,u_i]}(x_{i,k} - g_{i,k})$ which is trivially feasible. On the other hand, if $s_{i,k} = v_{i,k}/\eta_{i,k}$ with $\eta_{i,k} > 1$, then

$$x_{i,k+1} = x_{i,k} + \frac{v_{i,k}}{\eta_{i,k}} = \left(1 - \frac{1}{\eta_{i,k}}\right) x_{i,k} + \frac{1}{\eta_{i,k}} P_{[a_i,u_i]}(x_{i,k} - g_{i,k}), \qquad \text{for } i \in b_{\ell_k},$$

which is a convex combination of points in the interval $[a_i, u_i]$, provided that the first iterate x_0 is in the feasible set C; thus, it is feasible. We used the property of box constraint to be convex in each component, which is crucial due to the coordinatewise nature of box-constrained bAdag algorithm.

Moreover, we clarify that when $a_i = -\infty$ and $u_i = \infty$, Algorithm 5.1 results in a slightly more conservative version of the bAdag algorithm, where the step size is set to the smaller value between 1 and $1/\eta_{i,k}$ for every $i \in b_{\ell_k}$. Nevertheless, as the discussion above confirms, the modification in the choice of the step size in (5.46) is necessary to ensure the feasibility of the iterate in the box-constrained case.

We now introduce the descent lemma for Algorithm 5.1. Similarly to Lemma 4.1, we prove the result just for the UR case since those for GS and C rule follow by removing the expectation in all the computations.

Lemma 5.1 *Suppose that AS.1 - AS.3 hold. If Algorithm 5.1 is applied to problem* (5.44)*, then, according to the block selection rules GS, UR, and C, we have that, for all* $j \geq 0$*,*

$$\textbf{\textit{(GS/C)}}: \quad f(x_{j+1}) \leq f(x_j) - \varsigma \sum_{i=1}^{N} \frac{(v_{i,j}^{(\ell_j)})^2}{\eta_{i,j}} + \frac{L_{\ell_j}}{2} \sum_{i=1}^{N} \frac{(v_{i,j}^{(\ell_j)})^2}{\eta_{i,j}^2}, \tag{5.47}$$

$$\textbf{\textit{(UR)}}: \quad \mathbb{E}_j\left[f(x_{j+1})\right] \leq f(x_j) - \varsigma \mathbb{E}_j\left[\sum_{i=1}^{N} \frac{(v_{i,j}^{(\ell_j)})^2}{\eta_{i,j}}\right] + \frac{L_{\ell_j}}{2} \mathbb{E}_j\left[\sum_{i=1}^{N} \frac{(v_{i,j}^{(\ell_j)})^2}{\eta_{i,j}^2}\right]. \tag{5.48}$$

Proof. Let us first observe that the non-expansiveness of the projection operator yields the inequality $|g_{i,j}| \geq |v_{i,j}|$. Therefore, we distinguish between two disjoint cases, based on the value of the step $s_{i,k}$. In the first case ($s_{i,k} = v_{i,k}$), we have for every $i \in b_{\ell_k}$

$$|g_{i,j} s_{i,j}| \geq v_{i,j}^2 = \eta_{i,j} \frac{v_{i,j}^2}{\eta_{i,j}} \geq \varsigma \frac{v_{i,j}^2}{\eta_{i,j}}. \tag{5.49}$$

In the second case ($s_{i,k} = v_{i,k}/\eta_{i,j}$), for every $i \in b_{\ell_k}$, it holds $|g_{i,j} s_{i,j}| \geq \frac{v_{i,j}^2}{\eta_{i,j}}$. Thus, recalling that $\varsigma \in (0, 1]$, and taking the minimum with (5.49), we get $|g_{i,j} s_{i,j}| \geq \varsigma(v_{i,j}^2/\eta_{i,j})$. Since, by construction, $g_{i,j} s_{i,j} \leq 0$, we finally have

$$g_{i,j} s_{i,j} \leq -\varsigma \frac{v_{i,j}^2}{\eta_{i,j}}. \tag{5.50}$$

In addition, it trivially follows from (5.45) and (5.46) that $s_{i,k}^2 \leq v_{i,k}^2/\eta_{i,k}^2$

Using AS.3 and the observation in (4.9)

$$\begin{aligned}
\mathbb{E}_j\left[f(x_{j+1})\right] &\leq f(x_j) + \mathbb{E}_j\left[g_j^T s_j\right] + \frac{L_{\ell_j}}{2}\mathbb{E}_j\left[||s_j||^2\right] \\
&= f(x_j) + \mathbb{E}_j\left[\sum_{i\in b_{\ell_j}} g_{i,j}s_{i,j}\right] + \frac{L_{\ell_j}}{2}\mathbb{E}_j\left[\sum_{i\in b_{\ell_j}} s_{i,j}^2\right] \\
&\leq f(x_j) - \varsigma\mathbb{E}_j\left[\sum_{i\in b_{\ell_j}} \frac{v_{i,j}^2}{\eta_{i,j}}\right] + \frac{L_{\ell_j}}{2}\mathbb{E}_j\left[\sum_{i\in b_{\ell_j}} \frac{v_{i,j}^2}{\eta_{i,j}^2}\right] \\
&= f(x_j) - \varsigma\mathbb{E}_j\left[\sum_{i=1}^N \frac{(v_{i,j}^{(\ell_j)})^2}{\eta_{i,j}}\right] + \frac{L_{\ell_j}}{2}\mathbb{E}_j\left[\sum_{i=1}^N \frac{(v_{i,j}^{(\ell_j)})^2}{\eta_{i,j}^2}\right],
\end{aligned}$$

proving the lemma. □

We now present our non-asymptotic convergence result for Algorithm 5.1, covering the GS, UR and C block selection rules. The proof is omitted since it follows directly from the new descent Lemma 5.1, by applying the same strategy used in Theorem 4.5. The only difference is that the bounds are derived in terms of the suitable optimality measure $\|v_j\|$ for the constrained setting, rather than the standard gradient norm. While the extension for the GS and UR rules is straightforward, we show in Appendix A (Lemma 7) how to derive an upper bound on the average norm of v_j in terms of the average norm of $v_j^{(\ell_j)}$, when the C rule is employed.

Theorem 5.2 *Suppose that AS.1–AS.3 hold and that the Algorithm 5.1 is applied to problem (5.44). Let $\varsigma > 0$ and d the number of blocks. Defining $\Gamma_0 \stackrel{\text{def}}{=} f(x_0) - f_{\text{low}}$,*

- ***GS:***

$$\operatorname*{average}_{j\in\{0,\dots,k\}} \|v_j\| \leq \sqrt{d}\frac{\Theta_c}{\sqrt{k+1}}. \tag{5.51}$$

- ***C:*** *if, in addition, AS.4 holds, then*

$$\operatorname*{average}_{t\in\{0,\dots,T\}} \|v_{td}\| \leq \sqrt{2\left(1 + d\frac{6+4L^2}{\varsigma^2}\right)}\frac{\Theta_c}{\sqrt{T+1}}. \tag{5.52}$$

- ***UR:***

$$\mathbb{E}\left[\operatorname*{average}_{j\in\{0,\dots,k\}} \|v_j\|\right] \leq d\frac{\Theta_c}{\sqrt{k+1}}, \tag{5.53}$$

with

$$\Theta_c \stackrel{\text{def}}{=} \frac{2(\Gamma_0 + \varsigma^{\frac{3}{2}})}{\varsigma} + \frac{2N\bar{L}}{\varsigma}\left[\log\left(\frac{2N\bar{L}}{\varsigma^{3/2}}\right) - 1\right], \tag{5.54}$$

where $\bar{L} = \max_{\ell=1,\dots,d} L_\ell$.

6 Numerical experiments

In this section, we present numerical results on synthetic and real-world optimization problems, both in convex and nonconvex settings. Specifically, we analyze bAdag applied to convex least-squares and to nonconvex logistic regression with log-sum penalization and its box constrained version applied to robust Nonnegative Matrix Factorization (NMF) with the Huber function (Huber-NMF).

In what follows, we also consider noisy objective functions and gradients. The random noise is added following the idea proposed in [35], that is,

$$f(x) = f(x)(1 + \delta\mathcal{N}(0,1)), \quad \text{and} \quad [g(x)]_i = [g(x)]_i(1 + \delta\mathcal{N}(0,1)), \tag{6.55}$$

where $\delta \in [0,1]$ controls the noise level and $\mathcal{N}(0,1)$ is the normal distribution. We aim at:

1. comparing the performances of the three block selection strategies of bAdag across different applications;
2. comparing the bAdag algorithm with BCG using line-search to choose the step size (BCG-LS) showing the advantages of using an OFFO algorithm, such as bAdag, in situations where the objective function and the gradient are affected by noise;
3. showing the effectiveness of bAdag with bound constraint (Algorithm 5.1) in solving Huber-NMF.

In the least squares and logistic regression experiments, we compare our method with BCG-LS, where at each iteration we choose a step size satisfying the following Armijo condition:

$$f(x_k + \alpha_k g_k^{(\ell_k)}) \leq f(x_k) - c_1\alpha_k \|g_k^{(\ell_k)}\|^2, \tag{6.56}$$

where c_1 is set to 10^{-4}. The condition is checked using a standard backtracking strategy with a maximum of 20 inner iterations and a reduction of the step by a factor of 2 any time the Armijo condition is not satisfied. The first step at every iteration in the backtracking procedure is tuned for each class of problems. When dealing with noisy functions and gradients, there is no guarantee that the backtracking procedure in the BCG-LS algorithm produces an admissible step size; however, we do not report it as a failure but we take the last value given by the procedure and continue.

Setup We set $\varsigma = 0.0001$ in the bAdag implementations. We stop each algorithm either when a fixed time limit or maximum number of iterations is reached, or when the norm of the gradient is below a fixed threshold depending on the experiment; the stopping criteria are specified in each experiment. Since we do not want to evaluate the full gradient at each iteration, we check the convergence criteria on the full gradient norm every 20 iterations . All the results displayed are averaged over 10 runs. The algorithms are implemented in MATLAB R2021b on a 64-bit Samsung/Galaxy with 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz and 8 GB of RAM, under Windows 11 version 23H2.

6.1 Least squares

Let $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$, we solve least squares problems of the form

$$f(x) = \frac{1}{2}\|Ax - b\|_2^2.$$

For this experiment, we subdivide the variable $x \in \mathbb{R}^n$ into 10 blocks of dimension $\lfloor n/10 \rfloor$ or $\lceil n/10 \rceil$. In our experiments, we consider 14 different matrices A divided into the following classes:

1. Random matrices: randomly generated matrices from a standard normal distribution of dimensions $m \times n$ (`X = randn(m,n)` in MATLAB) with $(m,n) \in \{(1000,500),(500,500),(500,1000)\}$.
2. Random sparse matrices: randomly generated matrices from a standard normal distribution of dimensions with $(m,n) \in \{(1000,500),(500,500),(500,1000)\}$ where 50% of the entries are randomly replaced with zeros.
3. Bernoulli matrices: random matrix of dimension 500×1000 where each entry is either 0 or 1 with the same probability.

4. Discrete cosine transform matrix: we generate the matrix with `dctmtx(n)` in MATLAB and we truncate it to the first 500 rows corresponding to lowest frequencies, obtaining a wide matrix of size 500×1000.

5. Sparco matrices: 6 matrices from the collection of sparse signal recovery problems in [87] and supplied by S2MPJ [36]. Given a sparse vector x^*, the observation is generated as $b = Ax^* + r$, where r is additive noise vector of appropriate dimension and A is a fixed dictionary consisting of various bases such as wavelet, discrete cosine, and Fourier.

According to the definition in (6.55), we apply random noise to our least square problems, considering four noise levels ($\delta = 0, 0.2, 0.3, 0.4$). We generate 10 different instances for each problem by randomly changing the noise and the starting point for the algorithms. Thus, we run a total of 140 experiments. We randomly initialize each algorithm from a norm one vector $x_0/||x_0||_2$, where $x_0 =$ `randn(n,1)` in MATLAB. We set a tolerance on the norm of the gradient at 10^{-3} and a maximum time limit of 20 seconds. If the tolerance is not reached within the allotted time, we report the run as a failure. We consider performance profiles for assessing the quality of the solution, according to the definition in [26]. We choose the weighted sum of gradient and function evaluations, denoted as N_f and N_g respectively, as performance measure, with the relation $N_g = (1 + \frac{d_{\max}}{N})N_f$, where $d_{\max}$ is the dimension of the largest block. This relation is due to the higher computational cost of computing the block gradient with respect to evaluating the objective function. Therefore, one iteration of bAdag with C or UR rule which requires one block gradient computation, has a cost per iteration of $(1 + \frac{d_{\max}}{N})N_f$; while the cost per iteration is $2N_f$ if the GS rule is used, since it requires a full gradient evaluation per iteration. For BCG-LS, the cost is $(1 + \frac{d_{\max}}{N})N_f + \mu N_f$, where μ is the number of inner cycle in the backtracking strategy. We then compute

$$c_{p,s} = \text{cost required by solver } s \text{ to solve problem } p,$$

and evaluate

$$\rho(\tau) = \frac{1}{n_p} \left| \left\{ p \ : \ \frac{c_{p,s}}{\min_s c_{p,s}} \leq \tau \right\} \right|, \tag{6.57}$$

where n_p is the total number of problems considered. Our performance profile shows the percentage $\rho(\tau)$ computed over all problems considered of a given solver to have a performance ratio (solver performance measure / best performance measure) within a factor τ of the best possible solver. The results are displayed in Figure 1. The initial step size in the backtracking procedure of the BCG-LS algorithm is set to 1.

The performance profiles show that, in the absence or for small values of noise, BCG-LS is the best algorithm for any block selection rule considered, being more than 90 times less expensive than bAdag in the noiseless case and more than 30 times less expensive with 20% of noise. We observe an opposite behavior when the level of noise increases, in fact, the percentage of successfully solved problem of BCG-LS with random or cyclic block selection rule, decreases from 80% to 40% when the noise level is above the 30%. BCG-LS with GS rule still maintains a success rate over 70% when the noise level is 30%, while the accuracy drops to 40% when the noise is increased. On the other hand, bAdag is more robust, maintaining a constant success percentage in the allotted time of approximately 80% for all noise levels. Among the different variants of bAdag, the cyclic rule outperforms the GS criteria for all noise levels considered.

6.2 Logistic regression with log-sum penalty (LSP)

We address a binary logistic regression problem with a smooth log-sum penalty (LSP) of the form

$$f(x) = \frac{1}{N_S} \sum_{j=1}^{N_S} \log(1 + e^{-z_j y_j^T x}) + \lambda \sum_{i=1}^{N} \log(1 + \alpha x_i^2), \tag{6.58}$$

where $\{y_j, z_j\}$ is a labeled dataset, and at each sample $y_j \in \mathbb{R}^N$ corresponds a binary label $z_j \in \{0, 1\}$ for $j = 1, \ldots, N_S$ that classifies y_j into two disjoint classes. The objective function

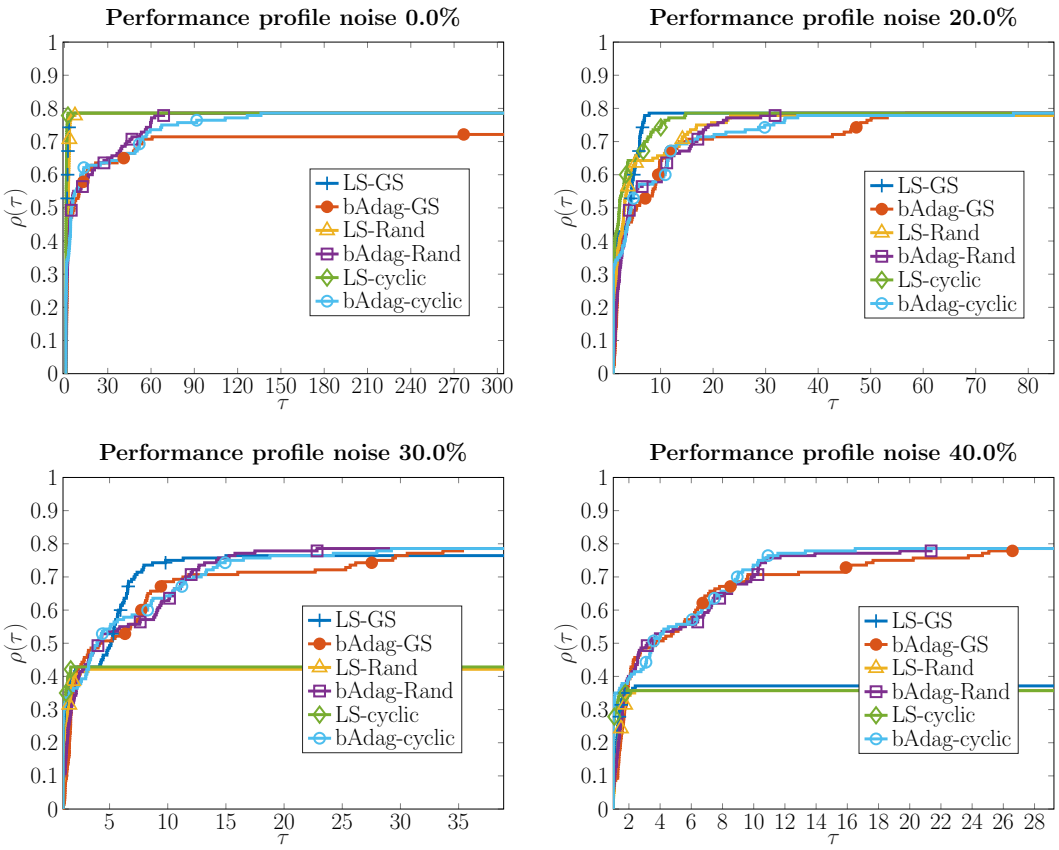

Figure 1: Performance profile according the definition given in [26]. The graphs show the percentage $\rho(\tau)$ in (6.57) having a performance ratio within a factor τ from the best solver. Results on 140 instances of least squares problems.

in (6.58) is smooth and nonconvex due to the LSP term. The log-sum function was first proposed by Candes et al. [18] as a relaxation of the sparsity-inducing ℓ_0-norm and it can be used as an alternative regularizer to the ℓ_1-norm in logistic regression that is less biased towards large coefficients [96].

In our experiments, we consider the following labeled data sets: MUSH [1], MNIST[(3)] [53], GISETTE [2], REGEDO [19], A9A [2], and MOLECULE [2]. For the MNIST and A9A, datasets, a random subset of 1,000 samples is selected for the experiments while all remaining datasets are used in their entirety. The data samples are randomly divided into training and testing set with a ratio of 70:30. We subdivide the variable $x \in \mathbb{R}^n$ into 10 blocks of dimension $\lfloor n/10 \rfloor$ or $\lceil n/10 \rceil$. We minimize the logistic loss in (6.58) on the training set, fixing the parameters to $\lambda = 0.1$ and $\alpha = 10$. We randomly initialize each algorithm from a norm one vector $x_0/||x_0||_2$, where $x_0 =$ `randn(n,1)` in MATLAB. We run each algorithm for 20 seconds and we stop it if a tolerance of 10^{-9} on the real gradient norm is reached. The initial step size in the backtracking procedure of the BCG-LS algorithm is chosen as $\|x_k\|/\|g_k^{(\ell_k)}\|$. We report in Table 3 the final objective function value averaged over 10 random initializations, for different values of the noise level δ. For completeness, we include in Appendix B the tables containing the final gradient norm (Table 6) and the percentage of correct classification on the testing set (Table 5).

Table 3 highlights the robustness of bAdag with respect to BCG-LS. In fact, we observe that in the noiseless scenario, BCG-LS provides the lowest value in the objective function for 4 out of

[(3)] Classification between even and odd numbers.

		bAdag			BCG-LS		
Data set	δ	GS	UR	Cyclic	GS	UR	Cyclic
Mush	0	0.1655	0.1655	0.1656	0.1653	0.1659	**0.1652**
	0.10	**0.1655**	**0.1655**	0.1656	0.1753	1.8548	0.5943
	0.20	**0.1656**	**0.1656**	0.1657	0.1874	13.49	13.97
	0.30	**0.1656**	**0.1656**	0.1658	1.3503	Inf	Inf
Gisette	0	0.0952	0.2555	0.1068	0.0929	0.0945	**0.0924**
	0.10	0.1578	0.2641	0.1085	**0.0989**	0.1510	0.1312
	0.20	**0.0971**	0.1658	0.1056	0.1228	0.3250	0.2861
	0.30	0.1388	0.6569	0.1335	**0.1265**	31.01	15.77
MNIST	0	**0.2952**	**0.2952**	**0.2952**	**0.2952**	**0.2952**	0.2954
	0.10	**0.2952**	**0.2952**	**0.2952**	0.3143	0.6423	0.5726
	0.20	**0.2952**	**0.2952**	**0.2952**	0.3221	Inf	Inf
	0.30	**0.2952**	**0.2952**	**0.2952**	0.4235	Inf	Inf
Regedo	0	0.1758	**0.1752**	0.2431	0.1797	0.1810	0.1834
	0.10	**0.1748**	0.1764	0.2360	0.1965	0.3001	0.2897
	0.20	0.1782	**0.1745**	0.2478	0.2575	8.872	1.378
	0.30	0.1790	**0.1754**	0.2137	0.2411	Inf	73.45
A9A	0	**0.3930**	**0.3930**	**0.3930**	**0.3930**	**0.3930**	**0.3930**
	0.10	**0.3930**	**0.3930**	**0.3930**	0.4143	20.41	20.27
	0.20	**0.3930**	**0.3930**	**0.3930**	0.4367	Inf	Inf
	0.30	**0.3930**	**0.3930**	**0.3930**	0.5822	Inf	Inf

Table 3: Final function value averaged over 10 random initializations for the logistic regression with LSP, for different levels of noise δ according to the definition in (6.55). Lowest function values per row are highlighted in bold and we denote as Inf if the simulation produced a non-finite function value at least once over 10 random initializations.

5 data seta but it suffers a significant drop in accuracy even for small values of noise. On the contrary, bAdag is less affected by the noise as the unchanged value of the function for MUSH, MNIST, and A9A testifies. Moreover, the GS rule achieves better accuracy on average both for bAdag and BCG-LS.

6.3 Nonnegative Matrix Factorization (NMF) with the Huber loss

We consider a robust NMF problem using the Huber function as error measure. NMF aims at approximating a given nonnegative matrix $X \in \mathbb{R}_+^{m\times n}$ by the product of two nonnegative factors, $W \in \mathbb{R}^{m\times r}$ and $H \in \mathbb{R}^{r\times n}$, where $r \ll \min(m,n)$ is the rank of the decomposition. NMF model finds application in several fields, such as hyperspectral unmixing [14, 63], topic modeling and document classifications [78, 25], and feature extraction [55, 41], see [31] for a book on the topic.

When data are affected by sparse noise or outliers, the common NMF models using the Frobenius norm or the KL divergence as an error measure might suffer from poor performance. For this reason, robust NMF models arise as a possible alternative by considering robust error measures such as ℓ_1-norm [49, 40, 77], $\ell_{2,1}$-norm [51], or the Huber function [27, 42, 9]. In this work, we focus on a regularized version of Huber NMF, that can be formulated as follows:

$$\min_{W\geq 0, H\geq 0} \Psi_\rho\left(X - WH\right) + \|W\|_F^2 + \|H\|_F^2 \quad \text{with} \quad \Psi_\rho(a) = \begin{cases} \frac{1}{2}a^2 & \text{if } |a| \leq \rho \\ \rho\left(|a| - \frac{1}{2}\rho\right) & \text{if } |a| > \rho \end{cases}, \tag{6.59}$$

where the Huber function is applied componentwise. The objective function in (6.59) is smooth, nonconvex, and its gradient is block-Lipschitz. However, due to the product WH, the function does not have a globally Lipschitz gradient. BCD algorithms are state-of-the-art techniques for solving matrix factorization problems, as they decompose the optimization into subproblems

corresponding to each factor. In this setting, due to the presence of the Huber loss, exact BCD approaches do not admit closed-form updates for W and H. Moreover, deriving explicit block-wise Lipschitz constants is not straightforward, which makes the choice of step size in BCG nontrivial. Even though the lack of the global Lipschitz property of the gradient prevents cyclic bAdag from having convergence guarantee, cyclic block selection rule is the most common approach for BCG in the NMF context. Therefore, we choose the cyclic rule for the experiments in this section.

Despite NMF being a nonconvex problem, it is well-known that extrapolation helps accelerating the convergence of BCG in the context of matrix factorization [46, 90, 5]. Thus, we implemented an extrapolated version of bAdag, dubbed ebAdag, which used two extrapolation steps after the update of each factor, that are

$$W^{k+1} = \max(0, W^{k+1} + \beta(W^{k+1} - W^k)), \quad H^{k+1} = \max(0, H^{k+1} + \beta(H^{k+1} - H^k)),$$

with β fixed to 0.9. We compare our algorithms bAdag and ebAdag with the Multiplicative Update (MU) algorithm proposed in [27], which is a specialized algorithm for NMF.

For this experiment, we consider the CBCL dataset, containing 2429 vectorized greyscale facial images of dimension 19×19, resulting in a data set $X_t \in \mathbb{R}^{2429\times 361}$. We add sparse noise to the images by randomly setting 7% of the pixels in white, obtaining a noisy version of the data $\hat{X}$. The rank of the factorization is $r = 49$. The ρ parameter in the Huber function is set to the mean value of the noisy dataset $\hat{X}$ and we fix both λ_W and λ_H to 10^{-4}. All the algorithms are randomly initialized: the entries of W and H are sampled from a uniform distribution in $[0, 1]$. We then scale them so that the initial point matches the magnitude of the original matrix, that is, we multiply W by $\sqrt{\|\hat{X}\|_F}/\|W\|_F$ and H by $\sqrt{\|\hat{X}\|_F}/\|W\|_F$. For this experiments, we run each algorithm for 500 iterations and no other stopping criteria is considered. We show the decrease of the objective function in (6.59), normalized dividing by the norm of X in Figure 2. Moreover, we display in Table 4 the final relative error in Frobenius norm with the original data ($\|X_t - WH\|_F/\|X_t\|_F$) and the CPU time for each algorithm.

We observe from Figure 2 that all algorithms have comparable results with the ebAdag having a faster decrease in the first stages and MU being slightly faster than bAdag. The final accuracy and running time in Table 4 confirms the similar behavior of ebAdag and MU that reach a better accuracy than bAdag.

	bAdag	ebAdag	MU
Relative error	0.1684	0.1681	**0.1675**
CPU time	13.39	13.54	**12.72**

Table 4: Final relative error in Frobenius norm ($\|X_t - WH\|_F/\|X_t\|_F$) and CPU time for each algorithm to compute Huber-NMF on the CBCL data. Results are averaged over 10 random initializations. Lowest error and running time are highlighted in bold.

We show an example of the images reconstructed by the ebAdag algorithm in Figure 3.

7 Conclusion

In this paper, we proposed the bAdag algorithm: a novel adaptive block coordinate gradient method which extends the AdaGrad algorithm to the block coordinate context. We proved sub-linear convergence rates for the three main block selection rules, namely Gauss-Southwell, uniform random selection, and cyclic rule. To our knowledge, this is the first adaptive algorithm covering the three block selection rules in the smooth, nonconvex setting. We also extended our framework and its convergence properties to box-constrained problems. We validated our algorithms on least squares and logistic regression, as well as on robust NMF.

Availability of data and materials

The data that support the findings of this study are available from the author upon request.

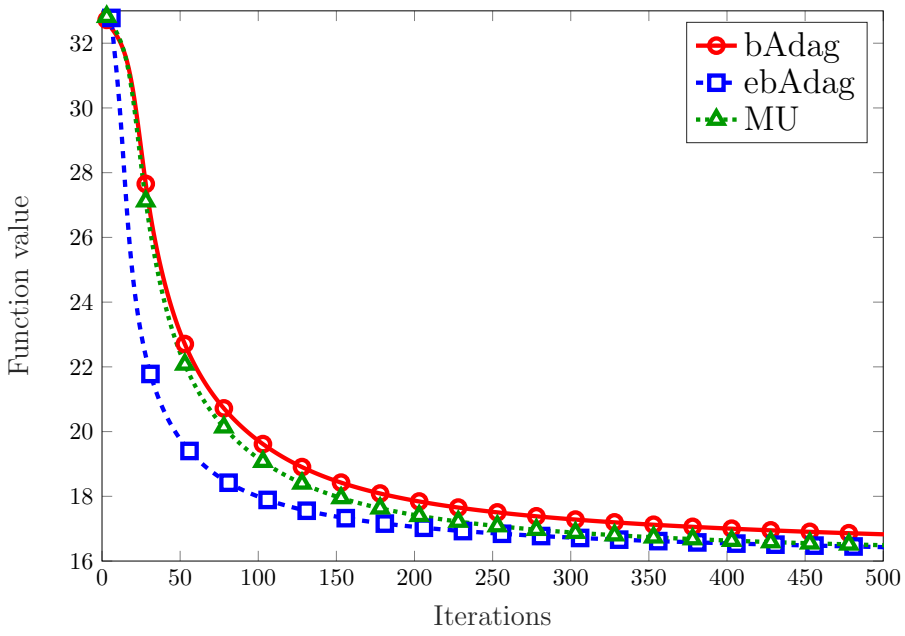

Figure 2: Huber-NMF objective function values along the iterations. All the algorithms run for 500 iterations and results are averaged over 10 random initializations.

Funding
The author acknowledges the support by the European Union (ERC consolidator, eLinoR, no 101085607). The work of the author was partially supported by INdAM-GNCS through Progetti di Ricerca.
Acknowledgment
I thank Margherita Porcelli, Stefania Bellavia, and all the NODA group (University of Florence) and Nicolas Gillis (University of Mons) for useful discussion and feedback on this work.

Appendix A

This lemma completes the convergence analysis in Section 5

Lemma 7.1

Suppose AS.1-AS.4 hold and that Algorithm 5.1 is applied to problem (5.44). Then it holds

$$\operatorname*{average}_{j\in\{0,\dots,k\}} \|v_j\| \leq \sqrt{2\left(1+d\frac{6+4L^2}{\varsigma^2}\right)\operatorname*{average}_{j\in\{0,\dots,k\}} \|v_j^{(\ell_j)}\|^2}.$$

Proof.

Let us use the same reparametrization introduced in case 2 of the proof of Theorem 4.5. Thus, we rewrite the iteration count as $k = Td + \ell$, where $\ell = 0, \dots, d-1$. By the structure of Algorithm 5.1,

$$x_{td+\ell} = x_{td} - \sum_{s=1}^{\ell-1} \frac{v_{td+s}^{(s)}}{\max(\eta_{td+s}, 1)}. \tag{7.60}$$

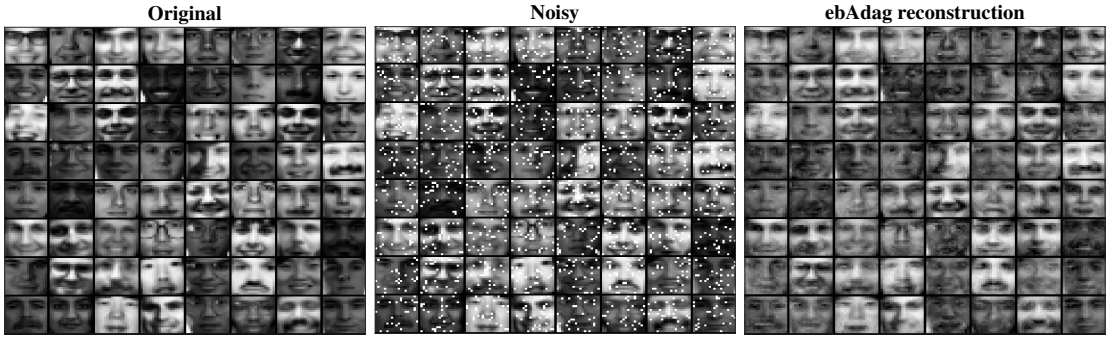

Figure 3: Visual reconstruction of 64 randomly selected images from the CBCL data. Original images on the left, noisy images in the center with 7% of randomly added white pixels, rank-49 Huber-NMF reconstruction after 500 iterations of the ebAdag algorithm.

Using (7.60) and the Lipschitz continuity of f in AS.4, and $\varsigma \leq \max(\eta_{td+s}, 1)$ for every t and ℓ,

$$\begin{aligned}
\|v_{td}^{(\ell)}\|^2 &\leq \left(\|v_{td}^{(\ell)} - v_{td+\ell}^{(\ell)}\| + \|v_{td+\ell}^{(\ell)}\|\right)^2 \leq 2\|v_{td}^{(\ell)} - v_{td+\ell}^{(\ell)}\|^2 + 2\|v_{td+\ell}^{(\ell)}\|^2 \\
&\leq 2\|v_{td} - v_{td+\ell}\|^2 + 2\|v_{td+\ell}^{(\ell)}\|^2 \\
&= 2\|-x_{td} + x_{td+\ell} + P_C(x_{td} - g_{td}) - P_C(x_{td+\ell} - g_{td+\ell})\|^2 + 2\|v_{td+\ell}^{(\ell)}\|^2 + 2\|v_{td+\ell}^{(\ell)}\|^2 \\
&\leq 4\|-x_{td} + x_{td+\ell}\|^2 + 4\|x_{td+\ell} - x_{td} + g_{td} - g_{td+\ell}\|^2 + 2\|v_{td+\ell}^{(\ell)}\|^2 \\
&\leq 12\|-x_{td} + x_{td+\ell}\|^2 + 8L^2\|-x_{td} + x_{td+\ell}\|^2 + 2\|v_{td+\ell}^{(\ell)}\|^2 \\
&\leq (12 + 8L^2)\left\|\sum_{s=1}^{\ell-1} \frac{v_{td+s}^{(s)}}{\max(\eta_{i,j}, 1)}\right\|^2 + 2\|v_{td+\ell}^{(\ell)}\|^2 \\
&\leq (12 + 8L^2)\sum_{s=1}^{\ell-1}\left\|\frac{v_{td+s}^{(s)}}{\max(\eta_{i,j}, 1)}\right\|^2 + 2\|v_{td+\ell}^{(\ell)}\|^2 \\
&\leq \frac{(12 + 8L^2)}{\varsigma^2}\sum_{s=1}^{\ell-1}\|v_{td+s}^{(s)}\|^2 + 2\|v_{td+\ell}^{(\ell)}\|^2
\end{aligned} \tag{7.61}$$

By applying the same reasoning presented in (4.32), and (4.33) we get

$$\frac{1}{T+1}\sum_{t=0}^{T}\|v_{td}\| \leq \frac{1}{\sqrt{T+1}}\sqrt{\sum_{t=0}^{T}\|v_{td}\|^2} \leq \frac{1}{\sqrt{T+1}}\sqrt{2\left(1 + d\frac{6 + 4L^2}{\varsigma^2}\right)\sum_{j=0}^{k}\|v_j^{(\ell_j)}\|^2}. \tag{7.62}$$

□

Appendix B

This section copletes the numerical experiments shown in Subsection 6.2. We report in Table 5 and Table 6 the final gradient norm and the accuracy on the test set, respectively, on the experiment following the same settings described in Subsection 6.2.

		bAdag			BCG-LS		
Data set	δ	GS	UR	Cyclic	GS	UR	Cyclic
Mush	0	$7.3 \cdot 10^{-4}$	$\mathbf{4.5 \cdot 10^{-4}}$	$4.8 \cdot 10^{-4}$	0.0011	0.0012	0.0011
	0.10	$7.2 \cdot 10^{-4}$	$\mathbf{4.8 \cdot 10^{-4}}$	$5.0 \cdot 10^{-4}$	0.0513	0.0740	0.1297
	0.20	$7.3 \cdot 10^{-4}$	$\mathbf{4.9 \cdot 10^{-4}}$	$5.1 \cdot 10^{-4}$	0.1657	0.0125	0.0122
	0.30	$6.6 \cdot 10^{-4}$	$\mathbf{4.8 \cdot 10^{-4}}$	$4.9 \cdot 10^{-4}$	0.1112	0.2223	0.4120
Gisette	0	0.0144	0.0508	0.0207	0.0129	0.0151	**0.0088**
	0.10	**0.0063**	0.0907	0.0279	0.0819	0.2145	0.1613
	0.20	**0.0074**	0.0712	0.0359	0.2790	0.2638	0.2408
	0.30	**0.0139**	0.0762	0.0313	0.2344	0.8555	0.9220
MNIST	0	0.0162	$\mathbf{4.9 \cdot 10^{-5}}$	0.0035	$1.4 \cdot 10^{-4}$	$2.3 \cdot 10^{-4}$	0.0045
	0.10	$\mathbf{3.2 \cdot 10^{-5}}$	0.0056	0.0031	0.1285	0.2360	0.2985
	0.20	0.0117	$\mathbf{5.6 \cdot 10^{-4}}$	0.0024	0.2282	0.2769	0.2498
	0.30	$\mathbf{2.1 \cdot 10^{-6}}$	$8.2 \cdot 10^{-4}$	0.0028	0.5221	1.5072	1.8589
Regedo	0	0.0639	$\mathbf{6.3 \cdot 10^{-4}}$	0.0590	0.0113	0.0084	0.0111
	0.10	0.0041	**0.0015**	0.0473	0.3390	0.4347	0.6469
	0.20	0.0089	**0.0020**	0.0521	0.3038	1.0931	0.6725
	0.30	0.0262	**0.0011**	0.0468	1.0205	2.5427	2.3687
A9A	0	$\mathbf{9.9 \cdot 10^{-10}}$	$\mathbf{9.9 \cdot 10^{-10}}$	$\mathbf{9.9 \cdot 10^{-10}}$	$1.3 \cdot 10^{-6}$	$8.1 \cdot 10^{-7}$	$5.4 \cdot 10^{-7}$
	0.10	$\mathbf{9.9 \cdot 10^{-10}}$	$\mathbf{9.9 \cdot 10^{-10}}$	$\mathbf{9.9 \cdot 10^{-10}}$	0.0916	0.1656	0.1554
	0.20	$\mathbf{9.9 \cdot 10^{-10}}$	$\mathbf{9.9 \cdot 10^{-10}}$	$\mathbf{9.9 \cdot 10^{-10}}$	0.1322	0.1787	0.1602
	0.30	$\mathbf{9.9 \cdot 10^{-10}}$	$\mathbf{9.9 \cdot 10^{-10}}$	$\mathbf{9.9 \cdot 10^{-10}}$	0.2295	0.4788	0.6092

Table 5: Final gradient norm for logistic regression with LSP, for different levels of noise τ according to the definition in (6.55). Lowest gradient norm per row are highlighted in bold.

References

[1] Mushroom. UCI Machine Learning Repository, 1981. DOI: https://doi.org/10.24432/C5959T.

[2] ___. UCI machine learning repository, 2013.

[3] Aviad Aberdam and Amir Beck. An accelerated coordinate gradient descent algorithm for non-separable composite optimization. *Journal of Optimization Theory and Applications*, 193(1):219–246, 2022.

[4] Zeyuan Allen-Zhu, Zheng Qu, Peter Richtárik, and Yang Yuan. Even faster accelerated coordinate descent using non-uniform sampling. In *International Conference on Machine Learning*, pages 1110–1119. PMLR, 2016.

[5] Andersen Man Shun Ang and Nicolas Gillis. Accelerating nonnegative matrix factorization algorithms using extrapolation. *Neural Computation*, 31(2):417–439, 2019.

[6] Hédy Attouch, Jérôme Bolte, Patrick Redont, and Antoine Soubeyran. Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the Kurdyka-łojasiewicz inequality. *Mathematics of Operations Research*, 35(2):438–457, 2010.

[7] Hedy Attouch, Jérôme Bolte, and Benar Fux Svaiter. Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward-backward splitting, and regularized Gauss-Seidel methods. *Mathematical programming*, 137(1):91–129, 2013.

[8] Alfred Auslender. Optimisation. *Méthodes numériques*, 1976.

[9] Wafa Barkhoda, Amjad Seyedi, Nicolas Gillis, and Fardin Akhlaghian Tab. Instance-wise distributionally robust nonnegative matrix factorization. *Pattern Recognition*, 169:111732, 2026.

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

[11] Amir Beck and Luba Tetruashvili. On the convergence of block coordinate descent type methods. *SIAM journal on Optimization*, 23(4):2037–2060, 2013.

[12] Stefania Bellavia, Serge Gratton, Benedetta Morini, and Ph L Toint. Fast stochastic second-order adagrad for nonconvex bound-constrained optimization. *arXiv preprint arXiv:2505.06374*, 2025.

[13] Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

		bAdag			BCG-LS		
Data set	δ	GS	UR	Cyclic	GS	UR	Cyclic
Mush	0	96.96	97.02	97.13	96.90	**97.58**	96.87
	0.10	96.96	97.03	97.01	**98.27**	91.03	87.82
	0.20	**98.99**	97.27	97.26	97.77	90.34	85.18
	0.30	97.00	97.22	**97.51**	94.39	80.26	79.11
Gisette	0	94.93	93.63	94.47	**95.10**	94.93	94.97
	0.10	92.73	93.07	**94.76**	94.50	93.93	94.93
	0.20	**94.80**	94.10	94.60	93.77	92.16	92.43
	0.30	**94.37**	91.80	94.30	94.10	85.90	87.47
MNIST	0	85.67	85.67	85.67	85.67	85.67	**86.00**
	0.10	**85.67**	**85.67**	**85.67**	84.27	80.03	80.83
	0.20	**85.67**	**85.67**	**85.67**	84.83	76.13	77.33
	0.30	**85.67**	**85.67**	**85.67**	80.43	62.57	70.70
Regedo	0	95.13	**95.40**	**95.40**	94.07	94.67	94.13
	0.10	95.47	95.13	**95.53**	94.00	94.67	93.80
	0.20	95.40	95.33	**95.73**	93.27	84.13	91.13
	0.30	95.07	95.07	**95.47**	93.80	72.93	88.13
A9A	0	**84.67**	**84.67**	**84.67**	**84.67**	**84.67**	**84.67**
	0.10	**84.67**	**84.67**	**84.67**	84.56	74.10	76.23
	0.20	**84.67**	**84.67**	**84.67**	80.63	75.90	77.53
	0.30	**84.67**	**84.67**	**84.67**	78.50	61.07	63.70

Table 6: Percentage of correct classification on the testing set, for different levels of noise τ according to the definition in (6.55). Highest percentages per row are highlighted in bold.

[14] José M Bioucas-Dias, Antonio Plaza, Nicolas Dobigeon, Mario Parente, Qian Du, Paul Gader, and Jocelyn Chanussot. Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 5(2):354–379, 2012.

[15] Jérôme Bolte, Shoham Sabach, and Marc Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, 146(1):459–494, 2014.

[16] Silvia Bonettini, Marco Prato, and Simone Rebegoldi. A cyclic block coordinate descent method with generalized gradient projections. *Applied Mathematics and Computation*, 286:288–300, 2016.

[17] Xufeng Cai, Chaobing Song, Stephen Wright, and Jelena Diakonikolas. Cyclic block coordinate descent with variance reduction for composite nonconvex optimization. In *International Conference on Machine Learning*, pages 3469–3494. PMLR, 2023.

[18] Emmanuel J Candes, Michael B Wakin, and Stephen P Boyd. Enhancing sparsity by reweighted $\ell 1$ minimization. *Journal of Fourier Analysis and Applications*, 14(5):877–905, 2008.

[19] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):1–27, 2011.

[20] Andrew R Conn, Nicholas IM Gould, and Philippe L Toint. *Trust region methods*. SIAM, 2000.

[21] Dominik Csiba and Peter Richtárik. Global convergence of arbitrary-block gradient methods for generalized Polyak- Lojasiewicz functions. *arXiv preprint arXiv:1709.03014*, 2017.

[22] Alexandre Défossez, Leon Bottou, Francis Bach, and Nicolas Usunier. A simple convergence proof of Adam and Adagrad. *Transactions on Machine Learning Research*, 2022.

[23] Inderjit Dhillon, Pradeep Ravikumar, and Ambuj Tewari. Nearest neighbor based greedy coordinate descent. *Advances in Neural Information Processing Systems*, 24, 2011.

[24] Jelena Diakonikolas and Lorenzo Orecchia. Alternating randomized block coordinate descent. In *International Conference on Machine Learning*, pages 1224–1232. PMLR, 2018.

[25] Chris Ding, Tao Li, and Wei Peng. On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing. *Computational Statistics & Data Analysis*, 52(8):3913–3927, 2008.

[26] Elizabeth D Dolan and Jorge J Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002.

[27] Liang Du, Xuan Li, and Yi-Dong Shen. Robust nonnegative matrix factorization via half-quadratic minimization. In *2012 IEEE 12th International Conference on Data Mining*, pages 201–210. IEEE, 2012.

[28] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(7), 2011.

[29] Tianxiang Gao, Songtao Lu, Jia Liu, and Chris Chu. Leveraging two reference functions in block Bregman proximal gradient descent for non-convex and non-lipschitz problems. *arXiv preprint arXiv:1912.07527*, 2019.

[30] Tianxiang Gao, Songtao Lu, Jia Liu, and Chris Chu. On the convergence of Randomized Bregman Coordinate Descent for Non-Lipschitz Composite Problems. In *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5549–5553, 2021.

[31] Nicolas Gillis. *Nonnegative Matrix Factorization*. SIAM, Philadelphia, 2020.

[32] Serge Gratton, Sadok Jerad, and Ph L Toint. Complexity of a class of first-order objective-function-free optimization algorithms. *Optimization Methods and Software*, pages 1–31, 2024.

[33] Serge Gratton, Sadok Jerad, and Ph L Toint. Complexity and performance for two classes of noise-tolerant first-order algorithms. *Optimization Methods and Software*, pages 1–27, 2025.

[34] Serge Gratton, Sadok Jerad, and Philippe L Toint. Parametric complexity analysis for a class of first-order adagrad-like algorithms. *arXiv preprint arXiv:2203.01647*, 2022.

[35] Serge Gratton, Sadok Jerad, and Philippe L Toint. Complexity of adagrad and other first-order methods for nonconvex optimization problems with bounds constraints. *arXiv preprint arXiv:2406.15793*, 2024.

[36] Serge Gratton and Ph L Toint. S2MPJ and CUTEst optimization problems for Matlab, Python and Julia. *Optimization Methods and Software*, 40(4):871–903, 2025.

[37] Serge Gratton and Ph L Toint. A unified convergence theory for adaptive first-order methods in the nonconvex case, including AdaNorm, full and diagonal AdaGrad, Shampoo and muon. *arXiv preprint arXiv:2604.17423*, 2026.

[38] Serge Gratton and Philippe L Toint. Stochastic convergence of parallel asynchronous adaptive first-order methods. *arXiv preprint arXiv:2606.01787*, 2026.

[39] Luigi Grippo and Marco Sciandrone. On the convergence of the block nonlinear Gauss-Seidel method under convex constraints. *Operations Research Letters*, 26(3):127–136, 2000.

[40] Naiyang Guan, Dacheng Tao, Zhigang Luo, and John Shawe-Taylor. MahNMF: Manhattan non-negative matrix factorization. *arXiv preprint arXiv:1207.3438*, 2012.

[41] David Guillamet and Jordi Vitria. Non-negative matrix factorization for face recognition. In *Catalonian Conference on Artificial Intelligence*, pages 336–344. Springer, 2002.

[42] Ziyang Guo, Anyou Min, Bing Yang, Junhong Chen, and Hong Li. A modified Huber nonnegative matrix factorization algorithm for hyperspectral unmixing. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 14:5559–5571, 2021.

[43] Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pages 1842–1850. PMLR, 2018.

[44] Filip Hanzely and Peter Richtárik. Fastest rates for stochastic mirror descent methods. *Computational Optimization and Applications*, 79(3):717–766, 2021.

[45] Filip Hanzely, Peter Richtarik, and Lin Xiao. Accelerated Bregman proximal gradient methods for relatively smooth convex optimization. *Computational Optimization and Applications*, 79(2):405–440, 2021.

[46] Le Thi Khanh Hien, Valentin Leplat, and Nicolas Gillis. Block majorization minimization with extrapolation and application to NMF. *SIAM Journal on Mathematics of Data Science*, 7(3):1292–1314, 2025.

[47] Mingyi Hong, Xiangfeng Wang, Meisam Razaviyayn, and Zhi-Quan Luo. Iteration complexity analysis of block coordinate descent methods. *Mathematical Programming*, 163(1):85–114, 2017.

[48] Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024.

[49] Qifa Ke and Takeo Kanade. Robust ℓ_1-norm factorization in the presence of outliers and missing data by alternative convex programming. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 739–746, 2005.

[50] Diederik P Kingma. Adam: a method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[51] Deguang Kong, Chris Ding, and Heng Huang. Robust nonnegative matrix factorization using l21-norm. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pages 673–682, 2011.

[52] Puya Latafat, Andreas Themelis, and Panagiotis Patrinos. Block-coordinate and incremental aggregated proximal gradient methods for nonsmooth nonconvex problems. *Mathematical Programming*, 193(1):195–224, 2022.

[53] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[54] Ching-Pei Lee and Stephen J Wright. Random permutations fix a worst case for cyclic coordinate descent. *IMA Journal of Numerical Analysis*, 39(3):1246–1275, 2019.

[55] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *nature*, 401(6755):788–791, 1999.

[56] Yin Tat Lee and Aaron Sidford. Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 147–156. IEEE, 2013.

[57] Qi Lei, Kai Zhong, and Inderjit S Dhillon. Coordinate-wise power method. *Advances in neural information processing systems*, 29, 2016.

[58] Xingguo Li, Tuo Zhao, Raman Arora, Han Liu, and Mingyi Hong. On faster convergence of cyclic block coordinate descent-type methods for strongly convex minimization. *Journal of Machine Learning Research*, 18(184):1–24, 2018.

[59] Miaomiao Liu, Dan Yao, Zhigang Liu, Jingfeng Guo, and Jing Chen. An improved Adam optimization algorithm combining adaptive coefficients and composite gradients based on randomized block coordinate descent. *Computational Intelligence and Neuroscience*, 2023(1):4765891, 2023.

[60] Zhaosong Lu and Lin Xiao. Randomized block coordinate non-monotone gradient method for a class of nonlinear programming. *arXiv preprint arXiv:1306.5918*, 1273, 2013.

[61] Zhaosong Lu and Lin Xiao. On the complexity analysis of randomized block-coordinate descent methods. *Mathematical Programming*, 152(1):615–642, 2015.

[62] Zhi-Quan Luo and Paul Tseng. On the convergence of the coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, 72(1):7–35, 1992.

[63] Wing-Kin Ma, José M Bioucas-Dias, Tsung-Han Chan, Nicolas Gillis, Paul Gader, Antonio J Plaza, Arul-Murugan Ambikapathi, and Chong-Yung Chi. A signal processing perspective on hyperspectral unmixing: Insights from remote sensing. *IEEE Signal Processing Magazine*, 31(1):67–81, 2013.

[64] H Brendan McMahan and Matthew Streeter. Adaptive bound optimization for online convex optimization. *arXiv preprint arXiv:1002.4908*, 2010.

[65] Ion Necoara and Flavia Chorobura. Efficiency of stochastic coordinate proximal gradient methods on nonseparable composite optimization. *Mathematics of Operations Research*, 50(2):993–1018, 2025.

[66] Ion Necoara and Dragos Clipici. Parallel random coordinate descent method for composite minimization: Convergence analysis and error bounds. *SIAM Journal on Optimization*, 26(1):197–226, 2016.

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

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

[69] Julie Nutini, Issam Laradji, and Mark Schmidt. Let's make block coordinate descent converge faster: Faster Greedy Rules, Message-Passing, Active-Set Complexity, and Superlinear Convergence. *Journal of Machine Learning Research*, 23(131):1–74, 2022.

[70] Julie Nutini, Mark Schmidt, Issam Laradji, Michael Friedlander, and Hoyt Koepke. Coordinate descent converges faster with the Gauss-Southwell rule than random selection. In *International Conference on Machine Learning*, pages 1632–1641. PMLR, 2015.

[71] Andrei Patrascu and Ion Necoara. Efficient random coordinate descent algorithms for large-scale structured nonconvex optimization. *Journal of Global Optimization*, 61(1):19–46, 2015.

[72] Zhimin Peng, Tianyu Wu, Yangyang Xu, Ming Yan, and Wotao Yin. Coordinate friendly structures, algorithms and applications. *arXiv preprint arXiv:1601.00863*, 2016.

[73] Margherita Porcelli, Giovanni Seraghiti, and Philippe L Toint. prunadag: an adaptive pruning-aware gradient method. *Computational Optimization and Applications*, 2026.

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

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

[76] Ankan Saha and Ambuj Tewari. On the finite time convergence of cyclic coordinate descent methods. *arXiv preprint arXiv:1005.2146*, 2010.

[77] Giovanni Seraghiti, Kévin Dubrulle, Arnaud Vandaele, and Nicolas Gillis. Nonnegative matrix factorization in the component-wise l1 norm for sparse data. *arXiv preprint arXiv:2603.29715*, 2026.

[78] Farial Shahnaz, Michael W Berry, V Paul Pauca, and Robert J Plemmons. Document clustering using nonnegative matrix factorization. *Information Processing & Management*, 42(2):373–386, 2006.

[79] Shai Shalev-Shwartz and Ambuj Tewari. Stochastic methods for ℓ_1 regularized loss minimization. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 929–936, 2009.

[80] Hao-Jun Michael Shi, Shenyinying Tu, Yangyang Xu, and Wotao Yin. A primer on coordinate descent algorithms. *arXiv preprint arXiv:1610.00040*, 2016.

[81] Chaobing Song and Jelena Diakonikolas. Cyclic coordinate dual averaging with extrapolation. *SIAM Journal on Optimization*, 33(4):2935–2961, 2023.

[82] Sebastian U Stich, Anant Raj, and Martin Jaggi. Approximate steepest coordinate descent. In *International Conference on Machine Learning*, pages 3251–3259. PMLR, 2017.

[83] Ruoyu Sun and Mingyi Hong. Improved iteration complexity bounds of cyclic block coordinate descent for convex problems. *Advances in Neural Information Processing Systems*, 28, 2015.

[84] Ruoyu Sun and Yinyu Ye. Worst-case complexity of cyclic coordinate descent: $o(n^2)$ gap with randomized version. *Mathematical Programming*, 185(1):487–520, 2021.

[85] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-RMSPROP, COURSERA: Neural networks for machine learning. *University of Toronto, Technical Report*, 6, 2012.

[86] Paul Tseng and Sangwoon Yun. Block-coordinate gradient descent method for linearly constrained nonsmooth separable optimization. *Journal of optimization theory and applications*, 140(3):513–535, 2009.

[87] Ewout van den Berg, MP Friedlander, G Hennenfent, F Herrmann, R Saab, and O Yılmaz. Sparco: A testing framework for sparse reconstruction. *Dept. Comput. Sci., Univ. British Columbia, Vancouver, Tech. Rep. TR-2007-20,[Online]. Available: http://www. cs. ubc. ca/labs/scl/sparco*, 2007.

[88] Rachel Ward, Xiaoxia Wu, and Leon Bottou. AdaGrad stepsizes: Sharp convergence over nonconvex landscapes. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 6677–6686. PMLR, 2019.

[89] Yi Wei, Xufeng Cai, and Jelena Diakonikolas. Adaptive delayed-update cyclic algorithm for variational inequalities. *arXiv preprint arXiv:2603.29128*, 2026.

[90] Zaiwen Wen, Wotao Yin, and Yin Zhang. Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm. *Mathematical Programming Computation*, 4(4):333–361, 2012.

[91] Stephen Wright and Ching-pei Lee. Analyzing random permutations for cyclic coordinate descent. *Mathematics of Computation*, 89(325):2217–2248, 2020.

[92] Stephen J Wright. Coordinate descent algorithms. *Mathematical programming*, 151(1):3–34, 2015.

[93] Xiaoxia Wu, Rachel Ward, and Léon Bottou. WNGrad: Learn the learning rate in gradient descent. *arXiv preprint arXiv:1803.02865*, 2018.

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

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

[96] Min Yuan and Yitian Xu. Feature screening strategy for non-convex sparse logistic regression with log sum penalty. *Information Sciences*, 624:732–747, 2023.

[97] Matthew D Zeiler. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.