Block Delayed Majorize-Minimize Subspace Algorithm for Large Scale Image Restoration

Mathieu Chalvidal, Emilie Chouzenoux, Jean-Baptiste Fest, Claire Lefort

▶ **To cite this version:**

Mathieu Chalvidal, Emilie Chouzenoux, Jean-Baptiste Fest, Claire Lefort. Block Delayed Majorize-Minimize Subspace Algorithm for Large Scale Image Restoration. 2022. hal-04281658v2

HAL Id: hal-04281658

https://hal.science/hal-04281658v2

Preprint submitted on 7 Jan 2023 (v2), last revised 13 Nov 2023 (v4)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Block Delayed Majorize-Minimize Subspace Algorithm for Large Scale Image Restoration‡

Mathieu Chalvidal[(a)], **Emilie Chouzenoux**[(b)], **Jean-Baptiste Fest**[(b)]§, **and Claire Lefort**[(c)]

(a) ANITI - CerCo UMR CNRS 5549, Toulouse France - Brown University, USA
(b) CVN, CentraleSupélec, Inria, Université Paris-Saclay, France.
(c) XLIM Research Institute, UMR CNRS 7252, Université de Limoges, France.

Abstract. In this work, we propose an asynchronous Majorization-Minimization (MM) algorithm for solving large scale differentiable non-convex optimization problems. The proposed algorithm runs efficient MM memory gradient updates on block of coordinates, in a parallel and possibly asynchronous manner. We establish the convergence of the resulting sequence of iterates under mild assumptions. The performance of the algorithm is illustrated on the restoration of 3D images degraded by depth-variant 3D blur, arising in multiphoton microscopy. Significant computational time reduction, scalability and robustness are observed on synthetic data, when compared to state-of-the-art methods. Experiments on the restoration of real acquisitions of a muscle structure illustrate the qualitative performance of our approach and its practical applicability.

Keywords Distributed optimization, asynchronous implementation, block alternating method, majorization-minimization, non-convex optimization, image restoration, depth-variant blur, multiphoton microscopy.

Submitted to: *XXX*

‡ This research work received funding support from the European Research Council Starting Grant MAJORIS ERC-2019-STG-850925.
§ Corresponding author. Contact: jean-baptiste.fest@centralesupelec.fr, 9 rue Joliot Curie, 91190 Gif-sur-Yvette, France.

1. Introduction

Large-scale optimization algorithms, benefiting from fast convergence, capable of utilizing modern computing infrastructures, and dealing with distributed datasets are becoming compulsory for solving inverse problems in modern imaging [23]. The ever-growing need for fast processing solutions that can operate on high-dimensional problems (i.e implying a huge number of variables) calls for the development of parallel methods harnessing the power of distributed computational architectures. In addition, the expansion of IoT systems and remote highly parallel computing induce new network issues with specific constraints. For instance, instabilities may occur whenever the volume of data dwarfs the memory capacity of a single agent or when the processing power is shared (potentially unevenly) between devices [46]. Several classes of so-called distributed optimization methods, have been investigated under various assumptions on the computing scenario and on the optimization problem itself, that we review hereafter (see also [77, 73]).

Distributed optimization approaches inherit from block alternating methods. In the latter, at each iteration, only a subset of the variables are updated, by minimizing the objective function with respect to only those variables, the others being fixed. The blocks are selected sequentially following a cyclic (or quasi-cyclic) order or a random rule. Exact minimization with respect to a given block of variables is rarely possible in a closed form. It is not even desirable as it may lead to convergence issues [68]. More efficient and stable block alternating schemes rely on a so-called Majorization-Minimization (MM) strategy [43]. It consists in building, at each iteration, a majorizing approximation for the objective function within the active block of variables, whose minimizer has a more tractable form. Many powerful algorithms fall within this framework, such as BSUM [40], PALM [8], multiplicative methods for NMF [47], to name a few. By exploiting the structure of the objective function, block alternating MM methods can reach fast convergence rates [28, 31, 54, 57] while offering theoretical guarantees in non-convex cases [8, 21, 9].

When the problem size increases, as in 3D microscopy imaging [48] and astronomy [58, 61], running block alternating methods gets inefficient. Parallel implementations have been devised, where the block updates are performed simultaneously, allowing to distribute computations on different nodes (or machines) [10, 65, 63]. Implementation on parallel architecture requires to pay attention to communication cost. The latter can be reduced by resorting to an asynchronous parallel implementation, yielding the so-called distributed optimization approach. Each computation node has now its own iteration loop, so local variables are updated without the need to wait for distant variables update. This however raises challenging questions, in terms of convergence analysis, as the communication delays may introduce instabilities. A plethora of recent works have focused on proposing distributed optimization algorithms with assessed convergence, based on stochastic proximal primal [37, 50, 52] or primal-dual [60, 39, 75, 14, 58, 1] techniques. Recent contributions in the field of federated learning are also highly related

[41, 51, 70]. However, as the aforementioned works rely on specific fixed-point analysis tools involving Fenchel-Rockafellar duality [4], the proposed algorithms are limited to convex (sometimes even strongly convex) optimization and often require specific probabilistic assumptions on the block update rule difficult to meet in practice. In the context of MM algorithms, although the need for distributed implementation strategies is crucial (see the discussion in [40] and the specific examples in [69, 30]), theoretical results regarding convergence guarantees of MM technique in a distributed context are rather scarce. Let us first mention the work of [24, 25], that proposes an asynchronous version of PALM, with proven convergence of the iterates in non-convex case, and good practical behaviour [67]. The convergence of distributed MM methods was also explored in the recent works [11, 49]. However, the analysis of [49] is limited to the convex case. In [11], the analysis covers non-convex terms in the objective function, but it only shows the convergence of the sequence of objective function values, and not the convergence of the iterates themselves (thus, the results is weaker than the one of [24]).

In this work, we aim at solving a smooth optimization problem of the form

$$\underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{minimize}} \quad f(\boldsymbol{x}), \tag{1}$$

where $f : \mathbb{R}^N \mapsto \mathbb{R}$ is (Fréchet) differentiable but non necessarily convex. In the context of inverse problems in imaging, f typically reads as the sum of a data fidelity term (e.g., a least-squares term) measuring the discrepancy between an acquired, degraded (e.g., blurry, noisy) image, and its estimate (usually, through a linear observation operator), and a regularization term incorporating prior information on the sought solution [23, 5] (see also our Section 5). We introduce the block delayed MM memory gradient (BD3MG) algorithm for the resolution of Problem (1). BD3MG is a distributed MM algorithm designed for an efficient implementation on a multi-CPU computing architecture, such as a high performance calculation unit. Our contributions are†:

- Introduction of the BD3MG algorithm, that implements an advanced distributed asynchronous update rule within the block alternating MM method we recently proposed in [32].
- Proof for the convergence of BD3MG iterates to a stationary point of f under mild assumptions (in particular, no convexity is assumed), using recent tools of Lyapunov analysis [71].
- Illustration of the performance of BD3MG by means of various experiments on a real inverse problem of 3D image restoration arising in the context of multiphotonic miscroscopy.

The paper is organized as follows. Section 2 introduces our notations, recalls the principle of MM schemes and finally presents our proposed algorithm. Section 3 states

† A preliminary version of this work has been presented in the conference proceedings [13]. The convergence result was weaker, and stated without proof. The experimental validation was limited to a single, simpler, numerical scenario.

our mathematical assumptions for the convergence analysis and presents preliminary technical propositions and lemmas. Section 4 presents our main theoretical contribution, dedicated to the convergence analysis of the proposed BD3MG scheme. Section 5 illustrates the qualitative and computational performance of BD3MG in the applicative context of 3D image deblurring in the presence of a depth-variant 3D blur. Section 6 concludes the paper.

2. Proposed algorithm

2.1. Notations

Throughout the paper, we consider the euclidean space $\mathbb{R}^N$ endowed with the usual scalar product $\langle \cdot | \cdot \rangle$ (or, equivalently, $\cdot^\top \cdot$) and the norm $\| \cdot \|$. $\mathbf{0}_N$ is the vector with null entries of $\mathbb{R}^N$. $\boldsymbol{I}_N$ is the identity matrix of $\mathbb{R}^N$. We use the short notation $[\![1, N]\!]$, to denote $\{1, 2, \ldots, N\}$, i.e. the set of integers from 1 to N. $\mathbb{S}^N$ denotes the set of symmetric matrices of $\mathbb{R}^{N\times N}$, and $\mathbb{S}^N_+$ (resp. $\mathbb{S}^N_{++}$) the set of positive (resp definite positive) symmetric matrices. Given some $\boldsymbol{M} \in \mathbb{S}^N_{++}$, we denote by $\| \cdot \|_{\boldsymbol{M}}$ the induced weighted euclidean norm, such that, for all $\boldsymbol{v} \in \mathbb{R}^N$, $\|\boldsymbol{v}\|^2_{\boldsymbol{M}} = \boldsymbol{v}^\top \boldsymbol{M}\boldsymbol{v}$. We use the Loewner orders symbols $\prec$ and $\preceq$, to compare real symmetric matrices $(\boldsymbol{A}, \boldsymbol{B}) \in (\mathbb{S}^N)^2$ i.e., $\boldsymbol{A} \preceq \boldsymbol{B}$ (resp. $\boldsymbol{A} \prec \boldsymbol{B}$) is verified when difference $\boldsymbol{B} - \boldsymbol{A}$ belongs to $\mathbb{S}^N_+$ (resp. $\mathbb{S}^N_{++}$).

Let us introduce extra notations, that will be useful to present block coordinate optimization strategy. Most notations hereafter are reminiscent from [32]. Let $\mathcal{S} \subset [\![1, N]\!]$.

- ▷ We denote by $\overline{\mathcal{S}}$ its complementary set $[\![1, N]\!] \backslash \mathcal{S}$, $|\mathcal{S}|$ its cardinal and $\left(\mathbb{R}^{|\mathcal{S}|}, \langle \cdot, \cdot \rangle\right)$ the resulting euclidean space (with a slight abuse of notation). Moreover, we also denote by $\mathbb{S}^{|\mathcal{S}|}, \mathbb{S}^{|\mathcal{S}|}_+, \mathbb{S}^{|\mathcal{S}|}_{++}$ respectively the set of symmetric, symmetric positive, and symmetric definite positive matrices of $\mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$.
- ▷ Let $\boldsymbol{x} = (x_n)_{n\in[\![1,N]\!]} \in \mathbb{R}^N$. We denote $\boldsymbol{x}_{(\mathcal{S})} = (x_i)_{i\in\mathcal{S}} \in \mathbb{R}^{|\mathcal{S}|}$ the vector gathering the entries of $\boldsymbol{x}$ with indexes within the set $\mathcal{S}$ of coordinates.
- ▷ Let $\boldsymbol{x} \in \mathbb{R}^N$. $\nabla f(\boldsymbol{x})$ is the gradient of f evaluated at $\boldsymbol{x}$. Moreover, $\nabla_{(\mathcal{S})} f(\boldsymbol{x}) = \left(\left[\nabla f(\boldsymbol{x})\right]_i\right)_{i\in\mathcal{S}} \in \mathbb{R}^{|\mathcal{S}|}$ denotes the partial gradient of f with respect to the coordinates with indexes in $\mathcal{S}$, evaluated at $\boldsymbol{x}$.
- ▷ Let $\boldsymbol{M} \in \mathbb{S}^N$. We denote the (symmetric) sub-matrix $\boldsymbol{M}_{(\mathcal{S})} = \left(M_{i,j}\right)_{(i,j)\in\mathcal{S}^2} \in \mathbb{S}^{|\mathcal{S}|}$. If $\boldsymbol{M}_{(\mathcal{S})} \in \mathbb{S}^{|\mathcal{S}|}_+$, we define the induced weighted Euclidean norm as $\| \cdot \|_{\boldsymbol{M}_{(\mathcal{S})}}$.
- ▷ For any $\boldsymbol{x} \in \mathbb{R}^N$, we introduce the restriction of f to the block $\mathcal{S}$ and vector $\boldsymbol{x}$ as the function $f_{(\mathcal{S})}(.\,, \boldsymbol{x}) : \boldsymbol{v} \in \mathbb{R}^{|\mathcal{S}|} \mapsto f(\boldsymbol{u})$ where $\boldsymbol{u}$ is related to $(\boldsymbol{v}, \boldsymbol{x})$ through the relations $\boldsymbol{u}_{(\mathcal{S})} = \boldsymbol{v}$ and $\boldsymbol{u}_{(\overline{\mathcal{S}})} = \boldsymbol{x}_{(\overline{\mathcal{S}})}$.

2.2. Block MM principle

MM approach to the resolution of Problem (1) is a generic iterative procedure where each iteration amounts to minimizing (exactly or not) a surrogate for f satisfying a majorizing property [66, 76, 43, 42]. The theoretical and practical properties of an MM algorithm greatly depend on (i) the family of considered surrogates, (ii) the procedure to minimize it. In this work, we focus on quadratic MM techniques, where f is such that it can be upper bounded by quadratic functions (typically, f is Lipschitz differentiable). In such context, the inner step of an MM algorithm amounts to minimizing a quadratic function on $\mathbb{R}^N$ or, otherwise stating, to invert an $N \times N$ system. In the large scale context, this is not desirable and various approaches have been proposed to cope with the curse of dimensionality in MM quadratic methods [16, 17, 20, 40, 10, 65, 32]. In particular, to limit the dependence of the MM algorithm on the dimension of the problem, block alternating approaches have been developed. In these schemes, at each iteration only a subset of the variables are updated [40], giving rise to so-called block MM algorithms, that we describe hereafter.

Define a partition $\mathbb{T}$ of $[\![1, N]\!]$. The block MM approach requires to build a majorizing surrogate for the restriction $f_{(\mathcal{S})}(\cdot, \boldsymbol{x})$ for any $\boldsymbol{x} \in \mathbb{R}^N$ and block index $\mathcal{S} \in \mathbb{T}$. Let us assume the existence of a mapping $\boldsymbol{A} : \boldsymbol{x} \in \mathbb{R}^N \mapsto \boldsymbol{A}(\boldsymbol{x}) \in \mathbb{S}^N_{++}$ such that for all $\mathcal{S} \in \mathbb{T}$, $\boldsymbol{v} \in \mathbb{R}^{|\mathcal{S}|}$ and $\boldsymbol{x} \in \mathbb{R}^N$

$$Q_{\mathcal{S}}(\boldsymbol{v}, \boldsymbol{x}) = f(\boldsymbol{x}) + \langle \nabla_{(\mathcal{S})} f(\boldsymbol{x}), \boldsymbol{v} - \boldsymbol{x}_{(\mathcal{S})} \rangle + \frac{1}{2} \left\| \boldsymbol{v} - \boldsymbol{x}_{(\mathcal{S})} \right\|^2_{\boldsymbol{A}_{(\mathcal{S})}(\boldsymbol{x})}, \tag{2}$$

fulfills the majorizing condition

$$Q_{\mathcal{S}}(\boldsymbol{v}, \boldsymbol{x}) \geqslant f_{(\mathcal{S})}(\boldsymbol{v}, \boldsymbol{x}). \tag{3}$$

Note that, by (2),

$$Q_{\mathcal{S}}(\boldsymbol{x}_{(\mathcal{S})}, \boldsymbol{x}) = f_{(\mathcal{S})}(\boldsymbol{x}_{(\mathcal{S})}, \boldsymbol{x}). \tag{4}$$

The existence of such mapping $\boldsymbol{A}$ can be ensured under mild assumptions. For instance, it is satisfied as soon as f is Lipschitz differentiable. Morever, [21, Remark 2.4] shows that, as soon as the above mapping holds for $\mathcal{S} = \mathbb{R}^N$, it stays valid for any block subset $\mathcal{S} \subset [\![1, N]\!]$. Examples of constructions of majorant mappings have been extensively discussed in [66, 17, 65] for optimization problems arising in the fields of inverse problems, image processing and telecommunication.

Once the block majorant approximations (2) satisfying (3) are built, the block MM (B2M) algorithm reads [43] (also called BSUM in [40]):

$$(\forall k \in \mathbb{N}) \quad \begin{cases} \text{Choose } \mathcal{S}^k \in \mathbb{T}, \\ \boldsymbol{x}^{k+1}_{(\mathcal{S}^k)} \in \underset{\boldsymbol{v} \in \mathbb{R}^{|\mathcal{S}^k|}}{\arg\min}\, Q_{\mathcal{S}^k}(\boldsymbol{v}, \boldsymbol{x}^k), \\ \boldsymbol{x}^{k+1}_{(\overline{\mathcal{S}^k})} = \boldsymbol{x}^k_{(\overline{\mathcal{S}^k})}. \end{cases} \tag{B2M}$$

Hereabove, $\left(\mathcal{S}^k\right)_{k\in\mathbb{N}}$ is a sequence of subsets (i.e., blocks) chosen in the predefined partition $\mathbb{T}$. The most current strategy is to adopt a cyclic rule, where each element of $\mathbb{T}$ is selected sequentially until the end of the partition list, and then the loop is repeated until convergence of the algorithm. A more flexible option is to adopt a so-called quasi-cyclic (or acyclic) rule where each $\mathcal{S} \in \mathbb{T}$ must be updated at least once per K iterations period.

The interest of scheme (B2M) and more generally block coordinate methods notably lies in the large scale context involving a very huge N, for which dealing with all the coordinates of the current iterate may be too high time consuming and even infeasible due to memory limitations. However, block MM methods require a sequential update of the blocks and thus, by construction, might require many iterations to reach convergence. To limit this issue, (block) diagonal mappings have been considered for instance in [65, 10]. The underlying idea is to choose the mapping so that the inner minimization problem in (B2M) is separable, and thus can be performed in parallel over the entries of the selected block. This yields the so-called block parallel MM schemes that take advantage of recent technological advances in parallel computing on multicore architectures. In particular, these methods can tailor the number of available processors to the computational load. However, such block diagonal structure may be detrimental to the approximation quality of the surrogates, and thus reduce again the practical convergence rate. In the present work, we opt for not making any extra structural assumption on the majorant mapping, thanks to the introduction of two catalizers into (B2M), namely (i) a subspace acceleration approach, (ii) a distributed asynchronous update strategy, that we describe hereafter.

2.3. Subspace acceleration

Our first catalyst is to introduce a subspace acceleration [66], in (B2M). This strategy has been initially introduced for full-batch MM algorithms (i.e., without any block coordinate strategy) in [16]. Convergence analysis can be found in [17, 15, 20, 19, 18] under various situations. We recently extended this strategy to cope with block coordinate updates with the form of (B2M) [32], leading to the B2MS (**B**lock **MM** **S**ubspace) scheme that we present hereafter.

Starting with the (B2M) iteration, the subspace acceleration subspace acceleration subspace acceleration consists in performing the minimization of the majorant function within the current block $\mathcal{S}^k$ in a constrained vectorial subspace spanned by a small

number $M_k \geqslant 1$ of search directions. This reads:

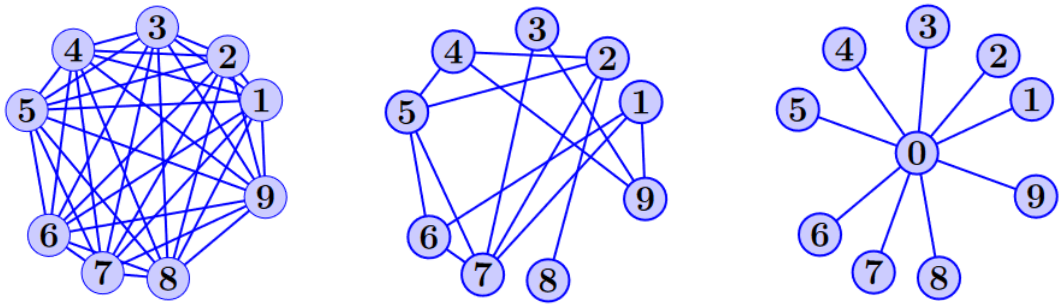
$$(\forall k \in \mathbb{N}) \quad \begin{cases} \text{Choose } \mathcal{S}^k \in \mathbb{T}, \\ \text{Choose } \boldsymbol{D}^k \in \mathbb{R}^{M_k \times |\mathcal{S}^k|}, \\ \boldsymbol{v}^k \in \underset{\boldsymbol{v} \in \mathbb{R}^{M_k}}{\arg\min}\, Q_{\mathcal{S}^k}(\boldsymbol{x}^k_{(\mathcal{S}^k)} + \boldsymbol{D}^k \boldsymbol{v}, \boldsymbol{x}^k), \\ \boldsymbol{x}^{k+1}_{(\mathcal{S}^k)} = \boldsymbol{x}^k_{(\mathcal{S}^k)} + \boldsymbol{D}^k \boldsymbol{v}^k, \\ \boldsymbol{x}^{k+1}_{(\overline{\mathcal{S}^k})} = \boldsymbol{x}^k_{(\overline{\mathcal{S}^k})}, \end{cases} \quad . \tag{B2MS}$$

Hereabove, for every $k \in \mathbb{N}$, $\boldsymbol{D}^k \in \mathbb{R}^{M_k \times |\mathcal{S}^k|}$ is the so-called subspace matrix. It stacks, row-wise, $M_k \geqslant 1$ vectors of dimension $|\mathcal{S}^k|$, spanning a vectorial subspace within which we seek for a minimizer of the majorant function $Q_{\mathcal{S}^k}(.\,, \boldsymbol{x}^k)$ (i.e., our next iterate). The advantage is to reduce again the dimensionality of the inner MM problems, without jeopardizing the convergence rate [19]. Several choices for the subspace matrix are discussed in [16, 20, 18]. Intensive comparisons in the fields of inverse problems, image processing and machine learning (e.g., [33, 15]), have shown the superiority of the so-called memory gradient subspace which seems to reach the best compromise between simplicity and efficiency. In the context of (B2MS), this amounts to defining, for every $k \in \mathbb{N}$, the memory gradient matrix $\boldsymbol{D}^k = \left[\nabla_{(\mathcal{S}^k)} f(\boldsymbol{x}^k), \boldsymbol{x}^k_{(\mathcal{S}^k)} - \boldsymbol{x}^{k-1}_{(\mathcal{S}^k)}\right]$ (with the convention $\boldsymbol{x}_{-1} = \boldsymbol{0}_N$), so that $M_k = 2$. When combined with a block diagonal majorant mapping, (B2MS) becomes equivalent to the BP3MG method considered in [13] for 3D image deblurring. The convergence properties of (B2MS) have recently been studied in [32].

2.4. Block Delayed Majorize-Minimize Memory Gradient (BD3MG)

The second catalyst we introduce is the main contribution of this paper, namely the introduction of a distributed asynchronous update rule within (B2MS). Our motivation is to make the algorithm well suited to an implementation on a multi-core / multi-processor architecture, while not being endangered by potential communication delays within the computing units. Let us consider a computing architecture with C units (or cores), each of them being able to communicate (i.e., send or receive) information to a master node. The architecture thus considered is forming a *star* graph as presented in Figure (1c). The two other graph topologies are discarded from this present study (see, for example, [1] for an efficient distributed method running on a generic hypergraph topology).

The proposed method BD3MG is presented in Algorithms 1-2, describing the iterations of the master (i.e., node 0) and a given worker/node $c \in [\![1, C]\!]$, respectively. Let us describe these two algorithms. Each computation node $c \in [\![1, C]\!]$ updates (independently from the other nodes) a subset of coordinates $\mathcal{S}_c \in \mathbb{T}$ (which can change over the process) by applying an MM iteration including a memory gradient acceleration and thus "books" its running block $\mathcal{S}_c$ so that no other worker overwrites the associated



(a) Complete graph (b) Connected graph (c) Star graph

Figure 1: Examples of graph topologies. The graph in (c) is encompassed by our framework.

coordinates. Conversely, any other $\mathcal{S} \in \mathbb{T}\backslash\{\mathcal{S}_c\}$ remains free to be updated by other workers. Communication steps are performed in order to maintain convergence to a minimizer of the globally shared objective function f and to control the propagation of errors. Basically, even if the other workers are still busy on their tasks, every time a worker $c \in [\![1, C]\!]$ ends one MM iteration on its running block $\mathcal{S}_c$, it sends a feedback to the master. As a response, the latter updates it with most recent available information, and assigns it a new task.

We denote $(\boldsymbol{x}^k)_{k\in\mathbb{N}}$ the sequence of iterates gathered by the master loop. For any given node index $c \in [\![1, C]\!]$ and $k \in \mathbb{N}$, $\mathcal{S}_c^k$ denotes the block of coordinates processed by worker c during step k. We impose, by construction, that two nodes do not update the same block of coordinates at the same time, so that we ensure the no-overlap condition

$$(\forall k \in \mathbb{N}) \quad (\forall (c, c') \in [\![1, C]\!]^2, c \neq c') \quad \mathcal{S}_c^k \cap \mathcal{S}_{c'}^k = \varnothing. \tag{5}$$

At iteration $k \in \mathbb{N}$, worker $c^k \in [\![1, C]\!]$, updates the block of coordinates $\mathcal{S}_{c^k}^k$ and sends to the master a vector $\boldsymbol{d}_k$ of size $\left|\mathcal{S}_{c^k}^k\right|$. The corresponding indexes of variable $\boldsymbol{x}^k$ within block $\mathcal{S}_{c^k}^k$ are then incremented with $\boldsymbol{d}_k$ while the others remain unchanged, thus defining $\boldsymbol{x}^{k+1}$. The master then defines a new set of coordinates $\mathcal{S}_{c^k}^{k+1}$ to be treated by worker c^k, so as to satisfy the non-overlap rule. The master informs worker c^k of this new running set of coordinates, and sends him the most recent information $\boldsymbol{x}^{k+1}$ and the difference $(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k)_{(\mathcal{S}_{c^k}^{k+1})}$. Meanwhile, the other workers keep processing their allocated indexes. The master then waits until a new worker (possibly the same one) sends a new increment.

Let us now make a focus on the worker loop described in Algorithm 2. Remark that, even if worker c has access to some properties of function f (i.e., the expression for its gradient and for its majorizing approximation $(Q_{\mathcal{S}})_{\mathcal{S}\in\mathbb{S}}$), it is not informed about the work done by the master or those of the other workers . It can only rely on the data the master sends to it to perform its local task. From the viewpoint of the worker, a triplet set $(\boldsymbol{x}, \mathcal{S}, \boldsymbol{d}) \in \mathbb{R}^N \times \mathbb{T} \times \mathbb{R}^{|\mathcal{S}|}$ is received from the master and must be used to perform its MM update with memory gradient acceleration. The worker is in charge of

Algorithm 1. *BD3MG algorithm - Master loop*

Initialization.

(a) Set $k = 0$ *and* $\boldsymbol{x}^0 \in \mathbb{R}^N$.

(b) Set $\mathcal{S}_1^0, ..., \mathcal{S}_C^0 \in \mathbb{T}$ *such that* $\forall (c, c^{'}) \in [\![1, C]\!]^2,\ \mathcal{S}_c^0 \cap \mathcal{S}_{c^{'}}^0 = \varnothing$.

(c) 0*-th transmission: For every* $c \in [\![1, C]\!]$, ***send*** $\left(\boldsymbol{x}^0, \mathcal{S}_c^0, 0_{|\mathcal{S}_c^0|}\right)$ *to worker* c

While *a stopping criterion is not met:*

(Wait for a feedback from a worker)

(a) $(k+1)$*-th reception:* ***Receive*** $\boldsymbol{d}_k$ *from a worker* c^k.

(b) Update $\boldsymbol{x}^{k+1}_{\left(\mathcal{S}^k_{c^k}\right)} = \boldsymbol{x}^k_{\left(\mathcal{S}^k_{c^k}\right)} + \boldsymbol{d}_k$ *and* $\boldsymbol{x}^{k+1}_{\left(\overline{\mathcal{S}^k_{c^k}}\right)} = \boldsymbol{x}^k_{\left(\overline{\mathcal{S}^k_{c^k}}\right)}$.

(c) Set $\mathcal{S}_1^{k+1}, \ldots, \mathcal{S}_C^{k+1} \in \mathbb{T}$ *such that* $\mathcal{S}_{c^k}^{k+1} \in \mathbb{T} \backslash \{\mathcal{S}_c^k\}_{c \neq c^k}$ *and,* $(\forall c \in [\![1, C]\!] \backslash c^k)$, $\mathcal{S}_c^{k+1} = \mathcal{S}_c^k$.

(d) $(k+1)$*-th transmission:* ***Send*** $\left(\boldsymbol{x}^{k+1}, \mathcal{S}_{c^k}^{k+1}, (\boldsymbol{x}^{k+1} - \boldsymbol{x}^k)_{\left(\mathcal{S}^{k+1}_{c^k}\right)}\right)$ *to worker* c^k.

(e) $k = k + 1$

End While

Output. *Vector* $\boldsymbol{x}^k$.

first building the new memory gradient matrix

$$\boldsymbol{D}(\boldsymbol{x}, \mathcal{S}, \boldsymbol{d}) = [-\nabla_{(\mathcal{S})} f(\boldsymbol{x}) \mid \boldsymbol{d}] \in \mathbb{R}^{|\mathcal{S}| \times 2}. \tag{6}$$

and then compute the MM increment $\boldsymbol{d}^{'} \in \mathbb{R}^{|\mathcal{S}|}$ defined as

$$\boldsymbol{d}^{'} = \boldsymbol{D}(\boldsymbol{x}, \mathcal{S}, \boldsymbol{d})\boldsymbol{u} \tag{7}$$

$$\text{with } \boldsymbol{u} \in \underset{\boldsymbol{v} \in \mathbb{R}^2}{\arg\min}\, Q_{\mathcal{S}}(\boldsymbol{x} + \boldsymbol{D}(\boldsymbol{x}, \mathcal{S}, \boldsymbol{d})\boldsymbol{v}, \boldsymbol{x}). \tag{8}$$

Note that the uniqueness of the solution for problem (7)-(8) is not guaranteed in general. To overcome such an obstacle, we follow the strategy in [16], and retain the pseudo-inverse solution given by

$$\boldsymbol{u} = -\left(\boldsymbol{D}(\boldsymbol{x}, \mathcal{S}, \boldsymbol{d})^\top \boldsymbol{A}_{(\mathcal{S})}(\boldsymbol{x})\ \boldsymbol{D}(\boldsymbol{x}, \mathcal{S}, \boldsymbol{d})\right)^\dagger \boldsymbol{D}(\boldsymbol{x}, \mathcal{S}, \boldsymbol{d})\ \nabla_{(\mathcal{S})} f(\boldsymbol{x}), \tag{9}$$

where † referes to the Moore-Penrose pseudo-inverse. Such solution notably verifies the normal equation

$$\left\langle \nabla_{(\mathcal{S})} f(\boldsymbol{x}), \boldsymbol{D}(\boldsymbol{x}, \mathcal{S}, \boldsymbol{d})\boldsymbol{u} \right\rangle = -\|\boldsymbol{D}(\boldsymbol{x}, \mathcal{S}, \boldsymbol{d})\boldsymbol{u}\|^2_{\boldsymbol{A}_{(\mathcal{S})}(\boldsymbol{x})}. \tag{10}$$

Algorithm 2. *BD3MG algorithm - Worker loop*

(Wait for a feedback from the master)

(a) `Receive` $(\boldsymbol{x}, \mathcal{S}, \boldsymbol{d})$ *from Master.*

(b) $\boldsymbol{D}(\boldsymbol{x}, \mathcal{S}, \boldsymbol{d}) = [-\nabla_{(\mathcal{S})} f(\boldsymbol{x}) \mid \boldsymbol{d}]$.

(c) Compute $\nabla_{(\mathcal{S})} f(\boldsymbol{x})$ *and* $\boldsymbol{A}_{(\mathcal{S})}(\boldsymbol{x})$.

(d) $\boldsymbol{u} = -\left(\boldsymbol{D}(\boldsymbol{x}, \mathcal{S}, \boldsymbol{d})^\top \boldsymbol{A}_{(\mathcal{S})}(\boldsymbol{x})\ \boldsymbol{D}(\boldsymbol{x}, \mathcal{S}, \boldsymbol{d})\right)^\dagger \boldsymbol{D}(\boldsymbol{x}, \mathcal{S}, \boldsymbol{d})\ \nabla_{(\mathcal{S})} f(\boldsymbol{x})$.

(e) `Send` $\boldsymbol{d}' = \boldsymbol{D}(\boldsymbol{x}, \mathcal{S}, \boldsymbol{d})\boldsymbol{u}$ *to the Master.*

2.5. Distributed structure of BD3MG

We first have to make an important remark, regarding the communication load in terms of memory, in between the master and the workers. Consider a worker associated to the block index $\mathcal{S}$. According to Algorithm 2, the worker receives three quantities, namely $\boldsymbol{x}$ of N real values, the set of integer indexes $\mathcal{S}$ with cardinality $|\mathcal{S}|$ and the vector $\boldsymbol{d}$ of $|\mathcal{S}|$ real values. The sent vector $\boldsymbol{d}'$ is again made of $|\mathcal{S}|$ real values. Clearly, the main memory load is related to the reception of vector $\boldsymbol{x}$. One should however notice that the worker only uses $\boldsymbol{x}$ to compute $\nabla_{(\mathcal{S})} f(\boldsymbol{x})$ and $\boldsymbol{A}_{(\mathcal{S})}(\boldsymbol{x})$. In most situations encountered in inverse problems of imaging, f shows some inherent separable structure, so that both of these quantities only depend on a subset of entries of vector $\boldsymbol{x}$ that can be of small cardinality compared to N. The practical implementation of Algorithm 2 should account for this specific situation, in order to avoid memory saturation and important communication delays. We give a detailed analysis for this aspect, in the case of our experimental example, in the Section 5.1.4.

The proposed distributed structure of BDM3G follows a star graph. Practically, it means that one of the computing unit has a higher load, in terms of memory, since it must process the full vector $\boldsymbol{x}$ of size N, while the memory load of the workers is limited, as we discussed hereabove. This can be viewed as a limitation for the proposed method. The extension of our analysis to the case of a hypergraph distributed framework would require to be more specific about the structure of function f (in the line of the study of [1]), which might reduce the versatility of the algorithm. Up to our knowledge, this analysis is not straightforward and is thus left as future work.

2.6. Equivalent form for BD3MG

The way we introduced our scheme BD3MG in the previous subsection was "implementation-oriented". In order to study its convergence behaviour, we must exhibit an equivalent form of it, mimicking the one of its non distributed counterpart, (B2MS). To do so, it is necessary to formalize the information gap between the master and the workers during the iterative process.

As we have already mentioned, all the information available to a worker (except those on f and $(Q_{\mathcal{S}})_{\mathcal{S}\in\mathbb{T}}$) is sent to it by the master only after it produces a feedback. For a given $k \in \mathbb{N}$, worker c^k does not receive any information between the $(k+1)$-reception and the previous one it made. During this time, its counterparts $c \in [\![1, C]\!] \backslash \{c^k\}$ may have performed additional updates to the master without c^k being informed. This results in an information mismatch, that we propose to formalize through a vector $\boldsymbol{x}^{\iota_k}$ where

$$(\forall k \in \mathbb{N}) \quad \iota_k = \begin{cases} 0 \text{ if } k = 0, \\ \max\left(\left\{\ell \in [\![1, k]\!] \mid c^{\ell-1} = c^k\right\} \cup \{0\}\right), \text{ otherwise.} \end{cases} \tag{11}$$

This vector corresponds to the iteration index of the working variable of worker c^k, which does not necessarily matches with the vector $\boldsymbol{x}^k$ manipulated by the master.
Let us list herebelow some situations of interest given the value of ι_k at some iteration $k \in \mathbb{N}$:

- If $\iota_k = 0$, and $k > 0$, it means that $\left\{\ell \in [\![1, k]\!] \mid c^{\ell-1} = c^k\right\}$ is an empty set. Hence, the worker c^k never returned any feedback to the master before the iteration k. Note that $\iota_0 = 0$ by construction.
- If $\iota_k = k$, we thus have $c^{k-1} = c^k$. Hence, worker c^k was in charge of the two most recent updates, namely the $(k+1)$-th and the k-th ones. As a consequence, to prepare the $(k+1)$-th update, worker c^k received vector $\boldsymbol{x}^k$ from the master.
- More generally, if $\iota_k > 0$, it follows that worker c^k at least returned one feedback to the master before iteration k. And we have the relation $c^{\iota_k - 1} = c^k$.

Moreover, the non-overlap rule translates into

$$(\forall k \in \mathbb{N}) \quad \boldsymbol{x}^{\iota_k}_{(\mathcal{S}^k_{c^k})} = \boldsymbol{x}^k_{(\mathcal{S}^k_{c^k})}. \tag{12}$$

For instance, if $\iota_k = k-1$ for some $k > 1$, this indicates that $c^{k-2} = c^k$ and $c^{k-1} \neq c^k$. The worker c^k thus proceeded to the $(k-1)$-th and $(k+1)$-th reception of the master while the k-th was made by another $\tilde{c}^k$ who received the vector $\boldsymbol{x}^k$ (from the master). However, since worker c^k was still processing block $\mathcal{S}^{k-1}_{c^k}$, the master was not able to update the associated coordinate for computing $\boldsymbol{x}^k$ from $\boldsymbol{x}^{k-1}$ for worker c^k, i.e $\boldsymbol{x}^{k-1}_{(\mathcal{S}^k_{c^k})} = \boldsymbol{x}^k_{(\mathcal{S}^k_{c^k})}$, which is typical from an asynchronous scheme.

More generally, when it comes to dealing with asynchronous algorithms, the use of a specific indexes with similar roles than our ι_k ($k \in \mathbb{N}$) is often necessary to build a theoretical delay model and thus to formulate an equivalent scheme being more compact and easier to analyse [24].

With this aim in mind, let us introduce the shorter notations

$$(\forall k \in \mathbb{N}) \quad \begin{cases} \mathcal{B}^k = \mathcal{S}^k_{c^k}, \\ \boldsymbol{D}^k = \boldsymbol{D}\left(\boldsymbol{x}^{\iota_k}, \mathcal{B}^k, \left(\boldsymbol{x}^{\iota_k} - \boldsymbol{x}^{\iota_k - 1}\right)_{(\mathcal{B}^k)}\right), \end{cases} \tag{13}$$

and $\boldsymbol{D}^k = \boldsymbol{D}\left(\boldsymbol{x}^{\iota_k}, \mathcal{B}^k, \left(\boldsymbol{x}^{\iota_k} - \boldsymbol{x}^{\iota_k - 1}\right)_{(\mathcal{B}^k)}\right)$ with convention $\boldsymbol{x}^{-1} = \boldsymbol{0}_N$. Then, the master/worker BD3MG loops from Algorithms 1-2 can be rewritten equivalently in a single compact scheme as:

$$(\forall k \in \mathbb{N}) \quad \begin{cases} \text{Let } c^k \in [\![1, C]\!], \\ \boldsymbol{u}^k = -\left((\boldsymbol{D}^k)^\top \boldsymbol{A}_{(\mathcal{B}^k)}\left(\boldsymbol{x}^{\iota_k}\right) \boldsymbol{D}^k\right)^\dagger (\boldsymbol{D}^k)^\top \nabla_{(\mathcal{B}^k)} f(\boldsymbol{x}^{\iota_k}), \\ \boldsymbol{x}^{k+1}_{(\mathcal{B}^k)} = \boldsymbol{x}^k_{(\mathcal{B}^k)} + \boldsymbol{D}^k \boldsymbol{u}^k, \\ \boldsymbol{x}^{k+1}_{(\overline{\mathcal{B}^k})} = \boldsymbol{x}^k_{(\overline{\mathcal{B}^k})}, \end{cases} \tag{14}$$

where we noticed that (12) now reads (using (13))

$$(\forall k \in \mathbb{N}) \quad \boldsymbol{x}^{\iota_k}_{(\mathcal{B}^k)} = \boldsymbol{x}^k_{(\mathcal{B}^k)}. \tag{15}$$

For every $k \in \mathbb{N}$, according to (14), $\boldsymbol{u}_k$ still reads (9) and thus verifies (10) with $\boldsymbol{D}^k = \boldsymbol{x}^{k+1}_{(\mathcal{B}^k)} - \boldsymbol{x}^k_{(\mathcal{B}^k)}$. The optimality equation can be rewritten as:

$$\left(\nabla_{(\mathcal{B}^k)} f(\boldsymbol{x}^{\iota_k})\right)^\top \left(\boldsymbol{x}^{k+1}_{(\mathcal{B}^k)} - \boldsymbol{x}^k_{(\mathcal{B}^k)}\right) = -\left\|\boldsymbol{x}^{k+1}_{(\mathcal{B}^k)} - \boldsymbol{x}^k_{(\mathcal{B}^k)}\right\|^2_{\boldsymbol{A}_{(\mathcal{B}^k)}(\boldsymbol{x}^{\iota_k})}. \tag{16}$$

The two next Sections are dedicated to establish the convergence of the iterates produced by (14).

2.7. Link with existing works

Let us discuss the links between our proposed scheme BD3MG and existing methods from the literature. When $\iota_k = k$ for any $k \in \mathbb{N}$ in BD3MG, the algorithm identifies with our block alternating scheme B2MS [32] where the blocks of variables were updated sequentially in a non parallel (thus, not asynchronous) manner. This present paper can thus be viewed as an extension of the framework and of the convergence analysis of [32] to the distributed setting. Other related methods are [24, 11, 49], and our convergence analysis relies on similar tools than the one from [24]. Assuming zero-valued non-smooth terms in [24, 11, 49] (i.e., the objective function is differentiable), these methods identify with particular instances of BD3MG that (i) would not implement any subspace acceleration (i.e., $\boldsymbol{D}_k = \boldsymbol{I}_N$ in (14)), (ii) would rely on the simple Lipschitz-based majorant metric (i.e., $\boldsymbol{A}_{(\mathcal{B}^k)}(\boldsymbol{x}^{\iota_k}) = \mathcal{L}\boldsymbol{I}_{|\mathcal{B}^k|}$ in (14)) in the case of [24]. As a consequence, assuming differentiability of all terms, our convergence analysis presented in the next section thus also covers the schemes of [24, 11, 49]. Up to our knowledge, our work is the first to show convergence of the iterates of a distributed MM algorithm involving generic quadratic surrogates and subspace acceleration, in the non-convex setting. Finally, we would like to point out that the 3MG update performed in Alg. 2 identifies with a non-linear conjugate gradient (NLCG) update, for a specific (and closed form) pair of stepsize and conjugacy parameters (see discussion in [18, Sec. 1]). Therefore, our work can also be understood as the first convergence analysis of a distributed NLCG method in the

non-convex setting. A comparative study will be conducted in our experimental section to illustrate the superiority of BD3MG with respect to the aforementioned existing methods in terms of convergence speed.

3. Assumptions and preliminary results

3.1. Assumptions

In order to analyse the asymptotic behaviour of the sequence $(\boldsymbol{x}^k)_{k\in\mathbb{N}}$ generated by scheme (14), we introduce technical assumptions both on function f and on the parameters of BD3MG method.

Assumption 1. *Function f is coercive, continuously differentiable on $\mathbb{R}^N$, and has a $\mathcal{L}$-Lipschitzian gradient with $\mathcal{L} > 0$, i.e.*

$$(\forall(\boldsymbol{x},\boldsymbol{y}) \in (\mathbb{R}^N)^2) \qquad \|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\| \leqslant \mathcal{L}\|\boldsymbol{x}-\boldsymbol{y}\|. \tag{17}$$

Assumption 1 ensures the existence of solutions for Problem (1) (by coercivity). Moreover, (17) in Assumption 1 guarantees the existence of a quadratic function (2) satisfying (3), setting $\boldsymbol{A} : \boldsymbol{x} \mapsto \mathcal{L}\boldsymbol{I}_N$. Another direct consequence is

$$(\forall \mathcal{S} \in \mathbb{T})(\forall(\boldsymbol{x},\boldsymbol{y}) \in (\mathbb{R}^N)^2) \qquad \|\nabla_{(\mathcal{S})} f(\boldsymbol{x}) - \nabla_{(\mathcal{S})} f(\boldsymbol{y})\| \leqslant \mathcal{L}\|\boldsymbol{x}-\boldsymbol{y}\|, \tag{18}$$

since $\|\nabla_{(\mathcal{S})} f(\boldsymbol{x}) - \nabla_{(\mathcal{S})} f(\boldsymbol{y})\| \leqslant \|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\|$ for all $\mathcal{S} \in \mathbb{T}$ and $(\boldsymbol{x},\boldsymbol{y}) \in (\mathbb{R}^N)^2$.

Assumption 2. *(Boundedness of delay) For every $k \in \mathbb{N}$, and every worker $c^k \in [\![1, N]\!]$, the set $\mathcal{S}_c^k$ is not empty and there exists $\tau \in \mathbb{N}^*$ such that*

$$(\forall k \geqslant \tau) \quad [\![1, N]\!] = \bigcup_{j=k-\tau}^{k-1} \mathcal{B}^j, \tag{19}$$

using the notation in (13).

Assumption 2 gives an upper bound on the delay τ. Each of the block of variables should be updated within a time frame of at most τ iterations and thus the workers must follow a certain regularity. Such a condition follows a similar goal than quasi-cyclic rule frequently assumed in block coordinate methods [32, 40]. From Assumption 2, we deduce the following Proposition, which appears fundamental for the rest of our convergence study. It guarantees that, for a given $k \in \mathbb{N}$, the vector treated by worker c^k before its feedback (i.e the $(k+1)$-th master's reception) is not "too old" compared to the iteration index.

Proposition 3.1. *Under Assumption 2, for every $k \geqslant \tau$, the index ι_k given in (11) belongs to $[\![k-\tau+1, k]\!]$.*

Proof. Let $k \geqslant \tau$, where $\tau > 0$ is defined in Assumption 2. Inequality $\iota_k \leqslant k$ directly comes from Definition (11). We prove the lower bound on ι_k by contradiction. Let us suppose that $\iota_k \leqslant k - \tau$. Two situations may arise.

Case 1: $\iota_k = 0$. By definition, $c^0, \dots, c^{k-1} \neq c^k$ and an easy induction gives $\mathcal{S}^0_{c^k} = \dots = \mathcal{S}^k_{c^k}$. Non-overlap rule (5) with $c^0, \dots, c^{k-1} \neq c^k$ yields

$$(\forall j \in [\![0, k-1]\!]) \quad \mathcal{S}^j_{c^k} \cap \mathcal{S}^j_{c^j} = \mathcal{S}^k_{c^k} \cap \mathcal{B}^j \tag{20}$$

$$= \varnothing. \tag{21}$$

Since $\mathcal{S}^k_{c^k}$ is non empty by Assumption 2, condition (20) ensures the existence of some $i_k \in [\![1, N]\!]$ verifying $i_k \notin \bigcup_{j=0}^{k-1} \mathcal{B}^j$ contradicting $\bigcup_{j=k-\tau}^{k-1} \mathcal{B}^j = [\![1, N]\!]$, as $k \geqslant \tau$.

Case 2: $\iota_k > 0$. We have $c^{\iota_k - 1} = c^k$ and a finite induction leads to

$$(\forall j \in [\![\iota_k, k]\!]) \quad \mathcal{S}^{\iota_k}_{c^k} = \mathcal{S}^{\iota_k}_{c^{\iota_k - 1}} = \mathcal{S}^j_{c^{\iota_k - 1}} = \mathcal{S}^j_{c^k}. \tag{22}$$

Majoration $\iota_k \leqslant k - \tau$ implies that

$$(\forall j \in [\![k-\tau, k]\!]) \quad \mathcal{S}^{\iota_k}_{c^k} = \mathcal{S}^j_{c^k}. \tag{23}$$

Non-overlap rule (5) with $c^{k-\tau}, \dots, c^{k-1} \neq c^k$ then gives

$$(\forall j \in [\![k-\tau, k-1]\!]) \quad \mathcal{S}^j_{c^k} \cap \mathcal{S}^j_{c^j} = \mathcal{S}^{\iota_k}_{c^k} \cap \mathcal{B}^j \tag{24}$$

$$= \varnothing. \tag{25}$$

Since $\mathcal{S}^{\iota_k}_{c^k}$ is non empty, Condition (24) thus ensures the existence of $i_k \in [\![1, N]\!]$ verifying $i_k \notin \bigcup_{j=k-\tau}^{k-1} \mathcal{B}^j$ which contradicts $\bigcup_{j=k-\tau}^{k-1} \mathcal{B}^j = [\![1, N]\!]$.

□

Assumption 3. *(Curvature of majorizing matrix)*

(i) The mapping $\boldsymbol{A} : \boldsymbol{x} \in \mathbb{R}^N \mapsto \boldsymbol{A}(\boldsymbol{x}) \in \mathbb{S}^N_{++}$ *is such that* (3) *holds. Moreover, there exists* $\overline{\nu} > 0$ *such that, for all* $\mathcal{S} \in \mathbb{T}$ *and* $\boldsymbol{v} \in \mathbb{R}^{|\mathcal{S}|}$,

$$0 \prec \boldsymbol{A}_{(\mathcal{S})}(\boldsymbol{v}) \preceq \overline{\nu} \boldsymbol{I}_{|\mathcal{S}|}. \tag{26}$$

(ii) There exists $\underline{\nu} > 0$ *such that, for all* $k \in \mathbb{N}$,

$$\boldsymbol{\Gamma}^k_c = \boldsymbol{A}_{(\mathcal{B}^k)}(\boldsymbol{x}^{\iota_k}) - \frac{1}{2}\boldsymbol{A}_{(\mathcal{B}^k)}(\boldsymbol{x}^k) \succeq \left(\frac{\mathcal{L}\sqrt{\tau}(1+\tau)}{2} + \underline{\nu} \right) \boldsymbol{I}_{|\mathcal{B}^k|}. \tag{27}$$

Assumption 3(i) is standard in optimization literature dealing with MM methods involving quadratic surrogates [16]. Assumption 3(ii) assumes that the spectrum of the difference between delayed and exact majorizing matrices of the partial quadratic majoring functions is strictly greater than a certain constant. This hypothesis controls the length of the MM increments performed by each worker. It aims at ensuring consistency between the asynchronous updates, by directly relating the worst-case curvature of the function f (parameterized by the Lipschitz constant $\mathcal{L}$) and the worst-case communication delay (parameterized by the constant τ). Condition (27) is key to ensure a condition descent for the general process generated by BD3MG scheme (see subsection 4.1). Assumption 3(ii) becomes redundant with Assumption 3(i) in the case when no delay occurs (i.e., $\tau = 0$). A detailed constructive example on how to meet Assumption 3(ii) will be provided in our experimental Section 5.

3.2. Technical lemmas

We conclude this section by presenting some preliminary results that will be useful for our convergence analysis.

Lemma 3.1. *Under Assumption 2, for every $k \geqslant \tau$,*

$$\left\|\boldsymbol{x}^k - \boldsymbol{x}^{\iota_k}\right\|^2 \leqslant \tau \sum_{j=k-\tau+1}^{k} \left\|\boldsymbol{x}^j - \boldsymbol{x}^{j-1}\right\|^2. \tag{28}$$

Proof. Let $k \in \mathbb{N}$. If $\iota_k = k$, inequality (28) is trivial. For the rest of the proof we thus suppose $\iota_k \leqslant k-1$. According to the definition of the euclidean norm we have

$$\left\|\boldsymbol{x}^k - \boldsymbol{x}^{\iota_k}\right\|^2 = \sum_{n=1}^{N} \left(x_n^k - x_n^{\iota_k}\right)^2. \tag{29}$$

Then, for all $n \in [\![1, N]\!]$, the Jensen's inequality leads to

$$\left(x_n^k - x_n^{\iota_k}\right)^2 = \left(\sum_{j=\iota_k+1}^{k} \left(x_n^j - x_n^{j-1}\right)\right)^2 \leqslant (k-\iota_k) \sum_{j=\iota_k+1}^{k} \left(x_n^j - x_n^{j-1}\right)^2. \tag{30}$$

Moreover, Proposition 3.1 ensures that ι_k belongs to $[\![k-\tau+1, k]\!]$. As a consequence

$$\left(\forall n \in [\![1, N]\!]\right) \quad \left(x_n^k - x_n^{\iota_k}\right)^2 \leqslant \tau \sum_{j=k-\tau+1}^{k} \left(x_n^j - x_n^{j-1}\right)^2. \tag{31}$$

We then replace (31) in (29), which yields

$$\left\|\boldsymbol{x}^k - \boldsymbol{x}^{\iota_k}\right\|^2 \leqslant \tau \sum_{j=k-\tau+1}^{k} \sum_{n=1}^{N} \left(x_n^j - x_n^{j-1}\right)^2. \tag{32}$$

Relation (28) directly comes from the identification of the inner sum of (32) as $\left\|\boldsymbol{x}^j - \boldsymbol{x}^{j-1}\right\|^2$ for all $j \in [\![k-\tau+1, k]\!]$. □

Lemma 3.1 provides a bound on the residual between $\boldsymbol{x}^k$ and the delayed vector $\boldsymbol{x}^{\iota_k}$ updated by worker c^k at iteration $k \in \mathbb{N}$. The right term in (28) can be understood as the extra information available to the master, when compared to the one available to worker c^k. This Lemma will allow to establish a descent condition on the BD3MG process in the next Section.

Lemma 3.2. *Under Assumptions 1 and 3(i), for every $k \in \mathbb{N}$,*

$$\|\nabla_{(\mathcal{B}^k)} f(\boldsymbol{x}^{\iota_k})\|^2 \leqslant \overline{\nu}^2 \|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|^2. \tag{33}$$

Proof. Let $k \in \mathbb{N}$. Let us analyse the quantity $f(\boldsymbol{x}^{\iota_k}) - Q_{\mathcal{B}^k}(\boldsymbol{x}^{k+1}_{(\mathcal{B}^k)}, \boldsymbol{x}^{\iota_k})$.

On the one hand, function $\Psi_k : \alpha \in \mathbb{R} \mapsto Q_{\mathcal{B}^k}(\boldsymbol{x}^k_{(\mathcal{B}^k)} - \boldsymbol{D}^k \alpha \boldsymbol{e}, \boldsymbol{x}^{\iota_k})$ with $\boldsymbol{e} = (1,0)^\top$ is a second degree convex polynomial with a unique minimizer that reads

$$\widehat{\alpha}_k = \frac{\|\nabla_{(\mathcal{B}^k)} f(\boldsymbol{x}^{\iota_k})\|^2}{\|\nabla_{(\mathcal{B}^k)} f\left(\boldsymbol{x}^{\iota_k}\right)\|^2_{\boldsymbol{A}_{(\mathcal{B}^k)}(\boldsymbol{x}^{\iota_k})}}. \tag{34}$$

Since $\boldsymbol{u}^k$ is a minimizer of $Q_{\mathcal{B}^k}(\boldsymbol{x}^{\iota_k}_{(\mathcal{B}^k)} + \boldsymbol{D}^k ., \boldsymbol{x}^{\iota_k}) = Q_{\mathcal{B}^k}(\boldsymbol{x}^k_{(\mathcal{B}^k)} + \boldsymbol{D}^k ., \boldsymbol{x}^{\iota_k})$, with $\boldsymbol{x}^{\iota_k}_{(\mathcal{B}^k)} = \boldsymbol{x}^k_{(\mathcal{B}^k)}$ by Equation (11), we deduce that

$$Q_{\mathcal{B}^k}(\boldsymbol{x}^{k+1}_{(\mathcal{B}^k)}, \boldsymbol{x}^{\iota_k}) \leqslant \Psi_k(\widehat{\alpha}_k) = f(\boldsymbol{x}^{\iota_k}) - \frac{1}{2}\widehat{\alpha}_k \|\nabla_{(\mathcal{B}^k)} f(\boldsymbol{x}^{\iota_k})\|^2. \tag{35}$$

From Assumption 3(i), $\widehat{\alpha}_k$ verifies $\widehat{\alpha}_k \geqslant \overline{\nu}^{-1}$. Equation (35) can thus be rewritten as

$$f(\boldsymbol{x}^{\iota_k}) - Q_{\mathcal{B}^k}(\boldsymbol{x}^{k+1}_{(\mathcal{B}^k)}, \boldsymbol{x}^{\iota_k}) \geqslant \frac{1}{2\overline{\nu}} \|\nabla_{(\mathcal{B}^k)} f(\boldsymbol{x}^{\iota_k})\|^2. \tag{36}$$

On the other hand, using (15) from Definition (2), and Equation (16) yield

$$\begin{aligned}
&f(\boldsymbol{x}^{\iota_k}) - Q_{\mathcal{B}^k}(\boldsymbol{x}^{k+1}_{(\mathcal{B}^k)}, \boldsymbol{x}^{\iota_k}) \\
&= \left\langle \nabla_{(\mathcal{B}^k)} f(\boldsymbol{x}^{\iota_k}) \ , \ \boldsymbol{x}^{k+1}_{(\mathcal{B}^k)} - \boldsymbol{x}^{\iota_k}_{(\mathcal{B}^k)} \right\rangle + \frac{1}{2} \left\| \boldsymbol{x}^{k+1}_{(\mathcal{B}^k)} - \boldsymbol{x}^{\iota_k}_{(\mathcal{B}^k)} \right\|^2_{\boldsymbol{A}_{(\mathcal{B}^k)}(\boldsymbol{x}^{\iota_k})} \\
&= \left\langle \nabla_{(\mathcal{B}^k)} f(\boldsymbol{x}^{\iota_k}) \ , \ \boldsymbol{x}^{k+1}_{(\mathcal{B}^k)} - \boldsymbol{x}^{k}_{(\mathcal{B}^k)} \right\rangle + \frac{1}{2} \left\| \boldsymbol{x}^{k+1}_{(\mathcal{B}^k)} - \boldsymbol{x}^{k}_{(\mathcal{B}^k)} \right\|^2_{\boldsymbol{A}_{(\mathcal{B}^k)}(\boldsymbol{x}^{k})} \\
&= \frac{1}{2} \left\| \boldsymbol{x}^{k+1}_{(\mathcal{B}^k)} - \boldsymbol{x}^{k}_{(\mathcal{B}^k)} \right\|^2_{\boldsymbol{A}_{(\mathcal{B}^k)}(\boldsymbol{x}^{\iota_k})}.
\end{aligned} \tag{37}$$

The combination of (36) and (37) leads to

$$\|\nabla_{(\mathcal{B}^k)} f(\boldsymbol{x}^{\iota_k})\|^2 \leqslant \overline{\nu} \left\| \boldsymbol{x}^{k+1}_{(\mathcal{B}^k)} - \boldsymbol{x}^{k}_{(\mathcal{B}^k)} \right\|^2_{\boldsymbol{A}_{(\mathcal{B}^k)}(\boldsymbol{x}^{\iota_k})}. \tag{38}$$

Finally, Equation (33) comes using Assumption 3(i), and in particular,

$$\left\| \boldsymbol{x}^{k+1}_{(\mathcal{B}^k)} - \boldsymbol{x}^{k}_{(\mathcal{B}^k)} \right\|^2_{\boldsymbol{A}_{(\mathcal{B}^k)}(\boldsymbol{x}^{\iota_k})} \leqslant \overline{\nu} \left\| \boldsymbol{x}^{k+1}_{(\mathcal{B}^k)} - \boldsymbol{x}^{k}_{(\mathcal{B}^k)} \right\|^2 \tag{39}$$

$$= \overline{\nu} \|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|^2. \tag{40}$$

□

Lemma 3.2 generalizes the decreasing behavior of standard MM schemes [17, 32] to the asynchronous context. It is not directly invoked in our main convergence proof but serves as an intermediary to show the following technical result.

Lemma 3.3. *Under Assumptions 1 and 3(i), for all $k \geqslant 2\tau$,*

$$\|\nabla f(\boldsymbol{x}^k)\| \leqslant \mathcal{L}\tau \sum_{j=k-2\tau+1}^{k} \|\boldsymbol{x}^j - \boldsymbol{x}^{j-1}\| + \overline{\nu} \sum_{j=k-\tau}^{k} \|\boldsymbol{x}^{j+1} - \boldsymbol{x}^j\|. \tag{41}$$

Proof. Let $k \geqslant 2\tau$. Assumption 2 allows us to bound the gradient of f at $\boldsymbol{x}^k$, as

$$\|\nabla f(\boldsymbol{x}^k)\|^2 \leqslant \sum_{\ell=k-\tau}^{k-1} \|\nabla_{(\mathcal{B}^\ell)} f(\boldsymbol{x}^k)\|^2 \leqslant \left(\sum_{\ell=k-\tau}^{k-1} \|\nabla_{(\mathcal{B}^\ell)} f(\boldsymbol{x}^k)\| \right)^2. \tag{42}$$

Let us extract the root of the above terms, and use triangular and gradient-Lipschitz inequalities, leading to

$$\begin{aligned} \|\nabla f(\boldsymbol{x}^k)\| &\leqslant \sum_{\ell=k-\tau}^{k-1} \|\nabla_{(\mathcal{B}^\ell)} f(\boldsymbol{x}^k) - \nabla_{(\mathcal{B}^\ell)} f(\boldsymbol{x}^{\iota_\ell})\| + \sum_{\ell=k-\tau}^{k-1} \|\nabla_{(\mathcal{B}^\ell)} f(\boldsymbol{x}^{\iota_j})\| \\ &\leqslant \mathcal{L} \sum_{\ell=k-\tau}^{k-1} \|\boldsymbol{x}^k - \boldsymbol{x}^{\iota_\ell}\| + \sum_{j=k-\tau}^{k-1} \|\nabla_{(\mathcal{B}^j)} f(\boldsymbol{x}^{\iota_j})\|. \end{aligned} \tag{43}$$

For all $\ell \in [\![k-\tau, k-1]\!]$, by Proposition 4.1, $\iota_\ell \geqslant \ell - \tau + 1 \geqslant k - 2\tau$. Thus,

$$\|\boldsymbol{x}^k - \boldsymbol{x}^{\iota_\ell}\| \leqslant \sum_{j=k-2\tau+1}^{k} \|\boldsymbol{x}^j - \boldsymbol{x}^{j-1}\|. \tag{44}$$

The right term of (44) does not depend on index ℓ. Using (44) and inequality (33) finally proves the result. □

Lemma 3.3 is useful as it provides a bound on the gradient at step $\boldsymbol{x}^k$ only depending on the $2\tau + 1$ past iterates $\boldsymbol{x}^k, \ldots, \boldsymbol{x}^{k-2\tau}$.

Lemma 3.4. *Let $(u^k)_{k\in\mathbb{N}}$ and $(v^k)_{k\in\mathbb{N}}$ be two sequences of positive real numbers. If there exists $P \in \mathbb{N}$ and $k^* \geqslant P$ such that*

$$(\forall k \geqslant k^*) \quad u^k \leqslant r \sum_{i=k-P}^{k-1} u^j + v^{k-1}, \tag{45}$$

with $r < 1/P$ and $\sum_{k=0}^{+\infty} v^k < +\infty$, then $\sum_{k=0}^{+\infty} u^k < +\infty$.

Proof. Summing (45) from k^* to $n \geqslant k^*$ leads to

$$\sum_{k=k^*}^{n} u^k \leqslant r \sum_{k=k^*}^{n} \sum_{j=k-P}^{k-1} u^j + \sum_{k=k^*}^{n} v^{k-1}, \tag{46}$$

with

$$\sum_{k=k^*}^{n} \sum_{j=k-P}^{k-1} u^j = \sum_{k=k^*}^{n} \sum_{j=1}^{P} u^{k-j} = \sum_{j=1}^{P} \sum_{k=k^*-j}^{n-j} u^k \leqslant \sum_{j=1}^{P} \sum_{k=0}^{n} u^k. \tag{47}$$

Plugging (47) into (46), yields

$$\sum_{k=k^*}^{n} u^k \leqslant rP \sum_{k=k*}^{n} u^k + \left(rP \sum_{k=0}^{k^*-1} u^k + \sum_{k=k^*}^{n} v^{k-1} \right) \leqslant rP \sum_{k=k*}^{n} u^k + \left(rP \sum_{k=0}^{k^*-1} u^k + \sum_{k=0}^{+\infty} v^k \right), \tag{48}$$

that is $(1-rP)\sum_{k=k*}^{n} u^k \leqslant rP \sum_{k=0}^{k^*-1} u^k + \sum_{k=0}^{+\infty} v^k$. With $0 < 1-rP < 1$, we deduce the summability of $\left(u^k\right)_{k\in\mathbb{N}}$. □

Lemma 3.4 is a technical result to ensure the convergence of some real series. Several variants of inequality (45) have been used to prove the finite length of iterative processes and then their convergence [24, 8].

4. Convergence results

Let us now state our main theoretical results, that relate to the convergence properties of BD3MG iterates. Our proof line is reminiscent of [24, 8] and follows similar steps that we summarize hereafter. First, starting from the majoration property (3) and using Lemma 3.1, we will establish a descent inequality. The latter is the key point of the rest of the proof. In particular, it will allow to show convergence of $\left(f(\boldsymbol{x}_k)\right)_{k\in\mathbb{N}}$. Then, Lemma 3.3 will ensure that $\left(\nabla f(\boldsymbol{x}_k)\right)_{k\in\mathbb{N}}$ converges to $\mathbf{0}_N$, and usual topological properties will serve to show that the set of cluster points $\mathcal{C}(\boldsymbol{x}^k)_{k\in\mathbb{N}})$ of $(\boldsymbol{x}^k)_{k\in\mathbb{N}}$ lies in the set of stationary point of f. Finally, we will exhibit a Lyapunov function from our descent inequality and will resort to the Kurdyka-Łojasewicz (KL) inequality [8] to prove our main theorem, showing the convergence of the BDM3G iterates and providing a rate of convergence.

4.1. Descent inequality

Proposition 4.1. *Under Assumptions 1-2-3, there exists a positive sequence* $(\nu_k)_{k\geqslant\tau}$ *such that*

$$(\forall k \geqslant \tau) \quad f(\boldsymbol{x}^{k+1}) + \nu_{k+1} \leqslant f(\boldsymbol{x}^k) + \nu_k - \underline{\nu}\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|^2. \tag{49}$$

Proof. By definition of the majorant function (3), for every $k \in \mathbb{N}$,

$$f(\boldsymbol{x}^{k+1}) \leqslant f(\boldsymbol{x}^k) + \left\langle \nabla_{(\mathcal{B}^k)} f(\boldsymbol{x}^k),\ \boldsymbol{x}^{k+1}_{(\mathcal{B}^k)} - \boldsymbol{x}^k_{(\mathcal{B}^k)} \right\rangle + \frac{1}{2}\|\boldsymbol{x}^{k+1}_{(\mathcal{B}^k)} - \boldsymbol{x}^k_{(\mathcal{B}^k)}\|^2_{\boldsymbol{A}_{(\mathcal{B}^k)}(\boldsymbol{x}^k)}. \tag{50}$$

Decomposing the scalar product term then yields, for every $k \in \mathbb{N}$,

$$f(\boldsymbol{x}^{k+1}) \leqslant f(\boldsymbol{x}^k) + R_k + \left\langle \nabla_{(\mathcal{B}^k)} f(\boldsymbol{x}^{\iota_k}),\ \boldsymbol{x}^{k+1}_{(\mathcal{B}^k)} - \boldsymbol{x}^k_{(\mathcal{B}^k)} \right\rangle + \frac{1}{2}\|\boldsymbol{x}^{k+1}_{(\mathcal{B}^k)} - \boldsymbol{x}^k_{(\mathcal{B}^k)}\|^2_{\boldsymbol{A}_{(\mathcal{B}^k)}(\boldsymbol{x}^k)}, \tag{51}$$

$$\text{with } R_k = \left\langle \nabla_{(\mathcal{B}^k)} f(\boldsymbol{x}^k) - \nabla_{(\mathcal{B}^k)} f(\boldsymbol{x}^{\iota_k}),\ \boldsymbol{x}^{k+1}_{(\mathcal{B}^k)} - \boldsymbol{x}^k_{(\mathcal{B}^k)} \right\rangle.$$

Let τ defined as in Assumption 2. A majoration of R_k for every $k \geqslant \tau$ comes by using successively Cauchy-Schwartz inequality, $\mathcal{L}$ gradient-Lipschitz inequality from Assumption 1, and Lemma 3.1:

$$\begin{aligned}(\forall k \geqslant \tau) \quad R_k &\leqslant \mathcal{L}\|\boldsymbol{x}^k - \boldsymbol{x}^{\iota_k}\| \, \|\boldsymbol{x}^{k+1}_{(\mathcal{B}^k)} - \boldsymbol{x}^k_{(\mathcal{B}^k)}\| \\ &\leqslant \frac{\mathcal{L}}{2\sqrt{\tau}}\|\boldsymbol{x}^k - \boldsymbol{x}^{\iota_k}\|^2 + \frac{\mathcal{L}\sqrt{\tau}}{2}\|\boldsymbol{x}^{k+1}_{(\mathcal{B}^k)} - \boldsymbol{x}^k_{(\mathcal{B}^k)}\|^2, \\ &\leqslant \frac{\mathcal{L}\sqrt{\tau}}{2}\sum_{j=k-\tau+1}^{k}\|\boldsymbol{x}^j - \boldsymbol{x}^{j-1}\|^2 + \frac{\mathcal{L}\sqrt{\tau}}{2}\|\boldsymbol{x}^{k+1}_{(\mathcal{B}^k)} - \boldsymbol{x}^k_{(\mathcal{B}^k)}\|^2. \end{aligned} \tag{52}$$

We then set, for all $k \geqslant \tau$, $\nu_k = \frac{\mathcal{L}\sqrt{\tau}}{2}\sum_{j=k-\tau+1}^{k}(j-k+\tau)\|\boldsymbol{x}^j - \boldsymbol{x}^{j-1}\|^2$. Since $\boldsymbol{x}^{k+1}_{(\mathcal{B}^k)} - \boldsymbol{x}^k_{(\mathcal{B}^k)} = \boldsymbol{x}^{k+1} - \boldsymbol{x}^k$, (52) also reads

$$\begin{aligned}(\forall k \geqslant \tau) \quad R_k &\leqslant \nu_k - \nu_{k+1} + \frac{\mathcal{L}\tau\sqrt{\tau}}{2}\|\boldsymbol{x}^{k+1}_{(\mathcal{B}^k)} - \boldsymbol{x}^k_{(\mathcal{B}^k)}\|^2 + \frac{\mathcal{L}\sqrt{\tau}}{2}\|\boldsymbol{x}^{k+1}_{(\mathcal{B}^k)} - \boldsymbol{x}^k_{(\mathcal{B}^k)}\|^2, \\ &= \nu_k - \nu_{k+1} + \frac{\mathcal{L}\sqrt{\tau}(1+\tau)}{2}\|\boldsymbol{x}^{k+1}_{(\mathcal{B}^k)} - \boldsymbol{x}^k_{(\mathcal{B}^k)}\|^2. \end{aligned} \tag{53}$$

Moreover, Equation (16) ensures that

$$(\forall k \geqslant \tau) \quad \left\langle \nabla_{(\mathcal{B}^k)} f(\boldsymbol{x}^{\iota_k}),\ \boldsymbol{x}^{k+1}_{(\mathcal{B}^k)} - x^k_{(\mathcal{B}^k)} \right\rangle = -\|\boldsymbol{x}^{k+1}_{(\mathcal{B}^k)} - \boldsymbol{x}^k_{(\mathcal{B}^k)}\|^2_{\boldsymbol{A}_{(\mathcal{B}^k)}(\boldsymbol{x}^{\iota_k})}. \tag{54}$$

Replacing both (53) and (54) in (51) gives, for all $k \geqslant \tau$,

$$\begin{aligned} f(\boldsymbol{x}^{k+1}) + \nu_{k+1} &\leqslant f(\boldsymbol{x}^k) + \nu_k + \frac{\mathcal{L}\sqrt{\tau}(1+\tau)}{2}\|\boldsymbol{x}^{k+1}_{(\mathcal{B}^k)} - \boldsymbol{x}^k_{(\mathcal{B}^k)}\|^2 - \|\boldsymbol{x}^{k+1}_{(\mathcal{B}^k)} - \boldsymbol{x}^k_{(\mathcal{B}^k)}\|^2_{\boldsymbol{\Gamma}^k_c} \\ &= f(\boldsymbol{x}^k) + \nu_k - \|\boldsymbol{x}^{k+1}_{(\mathcal{B}^k)} - \boldsymbol{x}^k_{(\mathcal{B}^k)}\|^2_{\boldsymbol{\Gamma}^k_c - \frac{\mathcal{L}\sqrt{\tau}(1+\tau)}{2}\boldsymbol{I}_{|\mathcal{B}^k|}}, \end{aligned} \tag{55}$$

with $\boldsymbol{\Gamma}^k_c$ defined in Assumption 3(ii). (49) is a direct consequence of Assumption 3(ii) remarking that $\|\boldsymbol{x}^{k+1}_{(\mathcal{B}^k)} - \boldsymbol{x}^k_{(\mathcal{B}^k)}\| = \|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|$. □

4.2. General behaviour

We now state our first convergence Theorem for BD3MG algorithm.

Theorem 4.1. *Under Assumptions 1-2-3, sequence* $\left(f(\boldsymbol{x}^k)\right)_{k\in\mathbb{N}}$ *generated by BD3MG converges to a finite limit* f^∞*. Moreover,* $\left(\nabla f(\boldsymbol{x}^k)\right)_{k\in\mathbb{N}}$ *converges to* $\mathbf{0}_N$*.*

Proof. Coercivity of f (Assumption 1) and (49) guarantee that $\left(f(\boldsymbol{x}^k) + \nu_k\right)_{k\in\mathbb{N}}$ is a decreasing and lower-bounded sequence. It is thus converging to a real value f^∞. Equation (49) then directly leads to $\sum_{k=0}^{+\infty}\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|^2 < +\infty$. On the first hand, using the same notation $(\nu_k)_{k\in\mathbb{N}}$ introduced in our proof of Proposition 4.1, we have

$$(\forall k \geqslant \tau) \quad \nu_k \leqslant \frac{\mathcal{L}\tau\sqrt{\tau}}{2}\sum_{j=k-\tau+1}^{+\infty}\|\boldsymbol{x}^j - \boldsymbol{x}^{j-1}\|^2. \tag{56}$$

Thus, the sequence $(\nu_k)_{k\in\mathbb{N}}$ converges to 0 and so, by Proposition 4.1, $\left(f(\boldsymbol{x}^k)\right)_{k\in\mathbb{N}}$ goes to f^∞. On the other hand, Lemma 3.3 gives

$$(\forall k \geqslant 2\tau) \quad \|\nabla f(\boldsymbol{x}^k)\| \leqslant \mathcal{L}\tau \sum_{j=k-2\tau+1}^{+\infty} \|\boldsymbol{x}^j - \boldsymbol{x}^{j-1}\| + \sum_{j=k-\tau}^{+\infty} \|\boldsymbol{x}^{j+1} - \boldsymbol{x}^j\|, \tag{57}$$

which enables to conclude that $\left(\nabla f(\boldsymbol{x}^k)\right)_{k\in\mathbb{N}}$ converges to $\mathbf{0}_N$.

□

Proposition 4.2. *Under Assumptions 1-2-3, $\mathcal{C}\left((\boldsymbol{x}^k)_{k\in\mathbb{N}}\right)$, defined as the set of accumulation points of $\left(\boldsymbol{x}^k\right)_{k\in\mathbb{N}}$, is non empty , compact and is a subset of the set of stationary points of f. Moreover, f takes value f^∞ on $\mathcal{C}\left((\boldsymbol{x}^k)_{k\in\mathbb{N}}\right)$.*

Proof. Coercivity of f (by Assumption 1) and convergence of $\left(f(\boldsymbol{x}^k)\right)_{k\in\mathbb{N}}$ to f^∞ (by Theorem 4.1) show that $\left(\boldsymbol{x}^k\right)_{k\in\mathbb{N}}$ is a bounded sequence and $\mathcal{C}\left((\boldsymbol{x}^k)_{k\in\mathbb{N}}\right)$ is non empty and compact. Convergence of $\left(\nabla f(\boldsymbol{x}^k)\right)_{k\in\mathbb{N}}$ to $\mathbf{0}_N$ (by Theorem 4.1) guarantees that every point $\boldsymbol{x}^* \in \mathcal{C}\left((\boldsymbol{x}^k)_{k\in\mathbb{N}}\right)$ is a stationary point of f. Moreover, using again $\left(f(\boldsymbol{x}^k)\right)_{k\in\mathbb{N}}$ converging to f^∞ yields $f^\infty = f(\boldsymbol{x}^*)$ for every $\boldsymbol{x}^* \in \mathcal{C}\left((\boldsymbol{x}^k)_{k\in\mathbb{N}}\right)$ which concludes the proof. □

4.3. Lyapunov-based asymptotical analysis

In order to establish the convergence of the iterates of BD3MG, we will follow an analysis relying on the use of a Lyapunov function. Such proof technique has also been used in [24, 74, 71]. The idea is to exhibit a function, related to the loss function f but non necessarily equals to it, that decreases monotonically along the iterative process. Given (49), a natural choice is

$$L : \boldsymbol{Z} = \begin{pmatrix} \boldsymbol{Z}_0 \\ \vdots \\ \boldsymbol{Z}_\tau \end{pmatrix} \in \mathbb{R}^{(\tau+1)N} \mapsto f(\boldsymbol{Z}_0) + \frac{\mathcal{L}\sqrt{\tau}}{2} \sum_{\ell=1}^{\tau} (\tau - \ell + 1)\|\boldsymbol{Z}_\ell - \boldsymbol{Z}_{\ell-1}\|^2. \tag{58}$$

Let us denote, for every $k \geqslant \tau$, $\boldsymbol{Z}^k = \begin{pmatrix} \boldsymbol{x}^k \\ \vdots \\ \boldsymbol{x}^{k-\tau} \end{pmatrix} \in \mathbb{R}^{(\tau+1)N}$, with $\boldsymbol{x}^k$ the k-th BD3MG iterate. Then, the descent condition from Proposition 4.1 can be rewritten as

$$(\forall k \geqslant \tau) \quad L(\boldsymbol{Z}^{k+1}) \leqslant L(\boldsymbol{Z}^k) - \underline{\nu}\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|^2. \tag{59}$$

The structure of L allows to build an upper bound of its gradient norm along the iterates, where the bound depends only on the differences of the past iterates:

Lemma 4.1. *There exists $\rho > 0$ such that*

$$(\forall k \geqslant \tau) \quad \|\nabla L(\boldsymbol{Z}^k)\| \leqslant \rho \sum_{j=k-2\tau+1}^{k} \|\boldsymbol{x}^j - \boldsymbol{x}^{j-1}\|. \tag{60}$$

Proof. Function L is differentiable. The expression of its gradient is

$$\left(\forall \boldsymbol{Z} \in \mathbb{R}^{(\tau+1)N}\right) \quad \nabla L(\boldsymbol{Z}) = \boldsymbol{g}_0 + \mathcal{L}\sqrt{\tau}\sum_{\ell=1}^{\tau}(\tau-\ell+1)\boldsymbol{g}_\ell, \tag{61}$$

$$\text{where } \boldsymbol{g}_0 = \begin{pmatrix} \nabla f(\boldsymbol{Z}_0) \\ \boldsymbol{0}_{\tau N} \end{pmatrix} \text{ and } \left(\forall \ell \in [\![1,\tau]\!]\right) \quad \boldsymbol{g}_\ell = \begin{pmatrix} \boldsymbol{0}_{(\ell-1)N} \\ \boldsymbol{Z}_{\ell-1} - \boldsymbol{Z}_\ell \\ \boldsymbol{Z}_\ell - \boldsymbol{Z}_{\ell-1} \\ \boldsymbol{0}_{(\tau-\ell)N} \end{pmatrix}. \tag{62}$$

Let us apply twice the Jensen inequality for the square of the norm and then the majoration $\tau-\ell+1 \leqslant \tau$ for $1 \leqslant \ell \leqslant \tau$. This yields

$$\begin{aligned}\left(\forall \boldsymbol{Z} \in \mathbb{R}^{(\tau+1)N}\right) \quad \|\nabla L(\boldsymbol{Z})\|^2 &\leqslant 2\|\boldsymbol{g}_0\|^2 + 2(\mathcal{L}\tau)^2\sum_{\ell=1}^{\tau}(\tau-\ell+1)^2\|\boldsymbol{g}_\ell\|^2 \\ &= 2\|\nabla f(\boldsymbol{Z}_0)\|^2 + 4(\mathcal{L}\tau)^2\sum_{\ell=1}^{\tau}(\tau-\ell+1)^2\|\boldsymbol{Z}_\ell - \boldsymbol{Z}_{\ell-1}\|^2 \\ &\leqslant 2\|\nabla f(\boldsymbol{Z}_0)\|^2 + 4\mathcal{L}^2\tau^4\sum_{\ell=1}^{\tau}\|\boldsymbol{Z}_\ell - \boldsymbol{Z}_{\ell-1}\|^2. \end{aligned} \tag{63}$$

Using $\sqrt{a^2+b^2} \leqslant a+b$ for the two quantities at the right of (63) and then standard norm majoration inequalities, we get:

$$\left(\forall \boldsymbol{Z} \in \mathbb{R}^{(\tau+1)N}\right) \quad \|\nabla L(\boldsymbol{Z})\| \leqslant \sqrt{2}\|\nabla f(\boldsymbol{Z}_0)\| + 2\mathcal{L}\tau^2\sum_{\ell=1}^{\tau}\|\boldsymbol{Z}_\ell - \boldsymbol{Z}_{\ell-1}\|. \tag{64}$$

The application of (64) to sequence $(\boldsymbol{Z}^k)_{k\in\mathbb{N}}$ leads to

$$(\forall k \geqslant \tau) \quad \|\nabla L(\boldsymbol{Z}^k)\| \leqslant \sqrt{2}\|\nabla f(\boldsymbol{x}_k)\| + 2\mathcal{L}\tau^2\sum_{j=k-\tau+1}^{k}\|\boldsymbol{x}^j - \boldsymbol{x}^{j-1}\|. \tag{65}$$

By Lemma 3.3 and (65), we finally deduce that

$$\begin{aligned}(\forall k \geqslant 2\tau) \quad \|\nabla L(\boldsymbol{Z}^k)\| &\leqslant \sqrt{2}\mathcal{L}\tau\sum_{j=k-2\tau+1}^{k}\|\boldsymbol{x}^j - \boldsymbol{x}^{j-1}\| + \sqrt{2}\overline{\nu}\sum_{j=k-\tau}^{k}\|\boldsymbol{x}^{j+1} - \boldsymbol{x}^j\| \\ &\quad + 2\mathcal{L}\tau^2\sum_{j=k-\tau+1}^{k}\|\boldsymbol{x}^j - \boldsymbol{x}^{j-1}\| \\ &\leqslant \left(\sqrt{2}L\tau + \sqrt{2}\overline{\nu} + 2L\tau^2\right)\sum_{j=k-2\tau+1}^{k}\|\boldsymbol{x}^j - \boldsymbol{x}^{j-1}\|, \end{aligned} \tag{66}$$

which concludes the proof taking $\rho = \sqrt{2}L\tau + \sqrt{2}\overline{\nu} + 2L\tau^2$. □

The following analysis makes use of recent theoretical results around the KL inequality [3, 8] that we recall hereafter. For every $\zeta > 0$, we denote by Φ_ζ the set of concave functions $\varphi\colon [0,\zeta) \mapsto \mathbb{R}_+$ verifying :

- $\varphi(0) = 0$.
- $\varphi \in C^1((0, \zeta))$ and is continuous in 0.
- $\forall s \in (0, \zeta), \varphi'(s) > 0$.

We are then ready to introduce the so-called KL property. [3, 8]

Definition 4.1. *[KL property] A differentiable function $g : \mathbb{R}^d \to \mathbb{R}$, with $d \geqslant 1$, satisfies the Kurdyka-Łojasiewicz (KL) property on $E \subset \mathbb{R}^d$ if, for every $\boldsymbol{z} \in E$ and every bounded neighborhood V of $\boldsymbol{z}$, there exist $\zeta > 0$ and $\varphi \in \Phi_\zeta$ such that every $\boldsymbol{x} \in E \cap \{\boldsymbol{x} \text{ s.t. } |g(\boldsymbol{x}) - g(\boldsymbol{z})| < \zeta\}$,*

$$\|\nabla g(\boldsymbol{x})\| \varphi'(|g(\boldsymbol{x}) - g(\boldsymbol{z})|) \geqslant 1. \tag{67}$$

We also recall the following Lemma:

Lemma 4.2. *[Uniform KL property [8, Lemma 6]] Let C a compact set of $\mathbb{R}^d$ and $g : \mathbb{R}^d \to \mathbb{R}$ a differentiable function satisfying KL property on C and constant on the latter. Then, there exist $\epsilon, \zeta > 0$ and $\varphi \in \Phi_\zeta$ such that every $\overline{\boldsymbol{x}} \in C$ and all $\boldsymbol{x} \in \mathbb{R}^d$ satisfying both $d(\boldsymbol{x}, C) < \epsilon$, $0 < g(\boldsymbol{x}) - g(\overline{\boldsymbol{x}}) < \zeta$, we have*

$$\|\nabla g(\boldsymbol{x})\| \varphi'(|g(\boldsymbol{x}) - g(\overline{\boldsymbol{x}})|) \geqslant 1. \tag{68}$$

Proposition 4.3. *Under Assumptions 1-2-3, if L defined in (58) fulfills the KL property on $\mathbb{R}^{(\tau+1)N}$ then, considering $g = L$, $C = \mathcal{C}\left((\boldsymbol{Z}^k)_{k\in\mathbb{N}}\right)$ with $L(C) = \{f^\infty\}$, there exists ϵ^L, ζ^L and $\phi^L \in \Phi_{\zeta^L}$ such that L satisfies (68).*

Proof. This is a direct consequence of Lemma 4.2. Continuity of L is clear. We still have to verify the compactness of $\mathcal{C}\left((\boldsymbol{Z}^k)_{k\in\mathbb{N}}\right)$ and that L is constant valued on that set. $\mathcal{C}\left((\boldsymbol{Z}^k)_{k\in\mathbb{N}}\right)$ is closed. Moreover, it is straightforward to show that this set is included in the Cartesian product $\left[\mathcal{C}\left((\boldsymbol{x}^k)_{k\in\mathbb{N}}\right)\right]^{\tau+1}$, where $\mathcal{C}\left((\boldsymbol{x}^k)_{k\in\mathbb{N}}\right)$ is compact. $\mathcal{C}\left((\boldsymbol{Z}^k)_{k\in\mathbb{N}}\right)$ is thus bounded and, finally, it is compact.
Let $\boldsymbol{Z} \in \mathcal{C}\left((\boldsymbol{Z}^k)_{k\in\mathbb{N}}\right)$. We have $L(\boldsymbol{Z}^k) = f(\boldsymbol{x}^k) + \nu_k$ for all $k \in \mathbb{N}$. From our proof of Theorem 4.1, it follows that sequence $\left(L(\boldsymbol{Z}^k)\right)_{k\in\mathbb{N}}$ converges to f^∞. Continuity of L finally ensures that $f^\infty = L(\boldsymbol{Z})$. This proves that f is constant valued on $\mathcal{C}\left((\boldsymbol{Z}^k)_{k\in\mathbb{N}}\right)$ (and equals to f^∞). □

4.4. Convergence of the iterates

We are now ready to state our second convergence Theorem for BD3MG algorithm, characterizing the convergence of $(\boldsymbol{x}_k)_{k\in\mathbb{N}}$.

Theorem 4.2. *Let assume that Assumptions 1-2-3 hold. Assume furthermore that the Lyapunov function L in (58) satisfies the KL property on $\mathbb{R}^{(\tau+1)N}$. Then, sequence $(\boldsymbol{x}^k)_{k\in\mathbb{N}}$ is of finite length, i.e :*

$$\sum_{k=0}^{+\infty} \|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\| < +\infty, \tag{69}$$

and converges to a stationary point of f.

Proof. Let us start considering the case when there exists some $k_0 \in \mathbb{N}$ where $L(\boldsymbol{Z}^{k_0}) = f^\infty$. Since $\left(L(\boldsymbol{Z}^k)\right)_{k\in\mathbb{N}}$ is decreasing sequence converging to f^∞ (see proof of Proposition 4.3), it follows that $L(\boldsymbol{Z}^k) = f^\infty$ for all $k \geqslant k_0$. (59) then gives

$$(\forall k \geqslant k_0) \quad \|\boldsymbol{x}^{k+1} - \boldsymbol{x}_k\|^2 \leqslant \underline{\nu}^{-1}\left(L(\boldsymbol{Z}^k) - L(\boldsymbol{Z}^{k+1})\right) = 0, \tag{70}$$

ensuring that $(\boldsymbol{x}^k)_{k\in\mathbb{N}}$ has a finite length and $\boldsymbol{x}^k$, $k \geqslant 0$, is a stationary point of f.

We now suppose that, for all $k \in \mathbb{N}$, $L(\boldsymbol{Z}^{k_0}) \neq f^\infty$. We aim at exhibiting a uniform KL inequality on sequence $\left(L(\boldsymbol{Z}^k)\right)_{k\in\mathbb{N}}$. To do so, let us peruse the quantities $\epsilon^L, \eta^L, \varphi^L$ arising from Proposition 4.3.
On the one hand, the decrease of $\left(L(\boldsymbol{Z}^k)\right)_{k\in\mathbb{N}}$ implies that, for all $k \in \mathbb{N}$, $L(\boldsymbol{Z}^k) > f^\infty$. The set $\mathcal{C}\left((\boldsymbol{x}^k)_{k\in\mathbb{N}}\right)$ is non empty (see proof of Proposition 4.2), so is the set $\mathcal{C}\left((\boldsymbol{Z}^k)_{k\in\mathbb{N}}\right)$. Let $\boldsymbol{Z} \in \mathcal{C}\left((\boldsymbol{Z}^k)_{k\in\mathbb{N}}\right)$ an element of such set i.e., a cluster point of $(\boldsymbol{Z}^k)_{k\in\mathbb{N}}$. From Proposition 4.3, $L(\boldsymbol{Z}) = f^\infty$. Hence, $L(\boldsymbol{Z}^k) - L(\boldsymbol{Z}) > 0$ for all $k \in \mathbb{N}$.
On the other hand, $\left(L(\boldsymbol{Z}^k)\right)_{k\in\mathbb{N}}$ converges to $f^\infty = L(\boldsymbol{Z})$. The boundedness of $(\boldsymbol{x}_k)_{k\in\mathbb{N}}$ also ensures this of $(\boldsymbol{Z}^k)_{k\in\mathbb{N}}$.
We deduce the existence of some $k_1 \geqslant 2\tau$ such that

$$(\forall k \geqslant k_1) \quad 0 < L(\boldsymbol{Z}^k) - L(\boldsymbol{Z}) < \eta^L, \quad d\left(\boldsymbol{Z}^k, \mathcal{C}\left((\boldsymbol{Z}^k)_{k\in\mathbb{N}}\right)\right) < \epsilon^L. \tag{71}$$

From Proposition 4.3, the uniform KL property on L holds i.e.,

$$(\forall k \geqslant k_1) \quad \|\nabla L(\boldsymbol{Z}^k)\| \left(\varphi^L\right)'\left(L(\boldsymbol{Z}^k) - L(\boldsymbol{Z})\right) \geqslant 1. \tag{72}$$

Moreover, setting $\Delta^k = \varphi^L\left(L(\boldsymbol{Z}^k) - L(\boldsymbol{Z})\right) - \varphi^L\left(L(\boldsymbol{Z}^{k+1}) - L(\boldsymbol{Z})\right)$ for all $k \in \mathbb{N}$, concavity of φ^L and (59) ensure that

$$\begin{aligned}(\forall k \geqslant k_1)\ \Delta^k &\geqslant \left(\varphi^L\right)'\left(L(\boldsymbol{Z}^k) - L(\boldsymbol{Z})\right)\left(L(\boldsymbol{Z}^k) - L(\boldsymbol{Z}^{k+1})\right) \\ &\geqslant \overline{\nu}\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|^2 \left(\varphi^L\right)'\left(L(\boldsymbol{Z}^k) - L(\boldsymbol{Z})\right).\end{aligned} \tag{73}$$

The combination of the latter with (72) leads to

$$(\forall k \geqslant k_1) \quad \|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|^2 \leqslant \overline{\nu}^{-1}\Delta^k\|\nabla L(\boldsymbol{Z}^k)\|. \tag{74}$$

By Lemma 4.1, we can upper bound the gradient term in (74). This gives

$$(\forall k \geqslant k_1) \quad \|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|^2 \leqslant \rho\overline{\nu}^{-1}\Delta^k \sum_{j=k-2\tau+1}^{k} \|\boldsymbol{x}^j - \boldsymbol{x}^{j-1}\|. \tag{75}$$

Passing to the root and using the classical identity $\sqrt{ab} \leqslant a/c + bc/4$, with $a = \sum_{j=k-2\tau+1}^{k} \|\boldsymbol{x}^j - \boldsymbol{x}^{j-1}\|$ for all $k \geqslant k_1$, $b = \Delta^k$, both positive for all $k \geqslant k_1$ and some

$c > 0$ is generic, leads to

$$(\forall k \geqslant k_1) \quad \|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\| \leqslant \frac{\sqrt{\rho\overline{\nu}}^{-1/2}}{c} \sum_{j=k-2\tau+1}^{k} \|\boldsymbol{x}^j - \boldsymbol{x}^{j-1}\| + \frac{c\sqrt{\rho\overline{\nu}}^{-1/2}}{4}\Delta^k. \tag{76}$$

Since $(\Delta^k)_{k\in\mathbb{N}}$ is summable (as a telescopic sequence), we can apply Lemma 3.4 with some $c > 2\tau\sqrt{\rho\overline{\nu}}^{-1/2}$ so that $2\tau\frac{\sqrt{\rho\overline{\nu}}^{-1/2}}{c} < 1$. This shows that sequence $(\boldsymbol{x}^k)_{k\in\mathbb{N}}$ has a finite length.

This finite length property entails that $(\boldsymbol{x}^k)_{k\in\mathbb{N}}$ is a Cauchy sequence and thus a converging one. The final conclusion directly comes from Proposition 3.1, ensuring that every accumulation point of $(\boldsymbol{x}^k)_{k\in\mathbb{N}}$ is a stationary point of f. □

4.5. Discussion

Under the KL condition for the Lyapunov function L defined in (58), we were able to demonstrate the convergence of sequence $(\boldsymbol{x}^k)_{k\in\mathbb{N}}$ to a stationary point of f. Let us notice that f satisfying KL property does not necessary imply that L does. Still, our assumption on L can be verified in practice for a wide class of functions f. For instance, following the discussion in [24, section 6], if f is semi-algebraic [7, 8], then the required condition on L in Theorem 4.2 is satisfied, with function $\varphi^L = \kappa(.)^{1-\theta}$ for a some $(\kappa, \theta) \in \mathbb{R}_+^* \times (0, 1)$. Such situation will be met in our experimental settings in Section 5. Extending Theorem 4.2 to any KL function f would be an interesting avenue for future work but up to our knowledge, it does not seem straightforward.

5. Application to 3D image restoration

5.1. Problem statement

5.1.1. Observation model. We focus on the inverse problem of restoring a vectorized 3D volume $\overline{\boldsymbol{x}}$ of size $N = N_{\mathsf{X}} \times N_{\mathsf{Y}} \times N_{\mathsf{Z}}$ given blurry and noisy observation $\boldsymbol{y} \in \mathbb{R}^N$. We consider a depth-variant 3D blur operator $\boldsymbol{H} \in \mathbb{R}^{N\times N}$ associated to kernels with support size $M = M_{\mathsf{X}} \times M_{\mathsf{Y}} \times M_{\mathsf{Z}}$, and additive i.i.d. Gaussian noise with standard deviation $\sigma > 0$, so that the observed volume is related to $\overline{\boldsymbol{x}}$ through,

$$\boldsymbol{y} = \boldsymbol{H}\overline{\boldsymbol{x}} + \boldsymbol{b}, \tag{77}$$

with vector $\boldsymbol{b} \in \mathbb{R}^N$ accounting for the noise. The goal is to solve the inverse problem of estimating $\overline{\boldsymbol{x}}$ given $\boldsymbol{y}$ and $\boldsymbol{H}$. Depth-variant blurs are commonly encountered in 3D microscopy [62, 38, 45, 44], due to optical aberrations. They are particular cases of spatially-variant blurs [12, 53]. The degradation operator $\boldsymbol{H}$ raises specific challenges due to its high computational cost. Several strategies have been investigated in the case of 2D spatially variant blur maps encountered for instance in astronomical imaging [27, 26, 29]. The extension to 3D maps of these methods is however not covered up to our knowledge. This motivates the use of a distributed optimization approach for solving the inverse problem (77).

5.1.2. Objective function We adopt a variational strategy, which consists in seeking for an estimate of $\overline{\boldsymbol{x}}$ that minimizes a penalized least squares criterion f. A hybrid regularization term is employed incorporating prior knowledge on the smoothness and the range of the sought solution. The objective function reads:

$$(\forall \boldsymbol{x} \in \mathbb{R}^N) \quad f(\boldsymbol{x}) = \sum_{s=1}^{S} f_s(\boldsymbol{L}_s \boldsymbol{x}), \tag{78}$$

where, for every $s \in \{1, \ldots, S\}$, $\boldsymbol{L}_s \in \mathbb{R}^{P_s \times N}$, $P_s \in \mathbb{N}^*$, and f_s is a function from $\mathbb{R}^{P_s}$ to $\mathbb{R}$. $f_1 \circ \boldsymbol{L}_1$ represents the data fidelity term while the other terms are regularization terms. Here, we set $S = 4$ and

- $P_1 = N$, $\boldsymbol{L}_1 = \boldsymbol{H}$, $f_1 = \frac{1}{2}\| \cdot - \boldsymbol{y}\|^2$,
- $P_2 = N$, $\boldsymbol{L}_2 = \boldsymbol{I}_N$, $f_2 = \eta\, d^2_{[x_{\min}, x_{\max}]^N}$,
- $P_3 = 2N$, $\boldsymbol{L}_3 = [(\boldsymbol{V}^{\mathsf{X}})^\top\ (\boldsymbol{V}^{\mathsf{Y}})^\top]^\top$, $f_3 = \lambda \sum_{n=1}^{N} \sqrt{[\cdot]_n^2 + [\cdot]_{N+n}^2 + \delta^2}$,
- $P_4 = N$, $\boldsymbol{L}_4 = \boldsymbol{V}^{\mathsf{Z}}$, $f_4 = \kappa\| \cdot \|^2$.

Hereabove, $(\eta, \lambda, \delta, \kappa) \in (0, +\infty)^4$ are hyper-parameters. The linear operators $\boldsymbol{V}^{\mathsf{X}}, \boldsymbol{V}^{\mathsf{Y}}, \boldsymbol{V}^{\mathsf{Z}} \in \mathbb{R}^{N \times N}$ are discrete gradient operators along X (horizontal), Y (vertical), and Z (longitudinal) directions of the 3D volume. Function $d^2_{[x_{\min}, x_{\max}]^N}$ states for the squared distance to set $[x_{\min}, x_{\max}]^N \subset \mathbb{R}^N$, with $(x_{\min}, x_{\max}) \in \mathbb{R}^2$ minimal and maximal bounds on the sought intensity values. The later term can be viewed as an exterior penalty function [18]. ‡.

5.1.3. Majorant mapping. In order to implement BD3MG, we must build a majorant mapping ensuring the majorization condition (3). First, let us notice that, according to (78), the gradient of f reads

$$(\forall \boldsymbol{x} \in \mathbb{R}^N) \quad \nabla f(\boldsymbol{x}) = \sum_{s=1}^{S} \boldsymbol{L}_s^\top \varphi_s(\boldsymbol{L}_s \boldsymbol{x}), \tag{79}$$

with, for every $s \in \{1, \ldots, S\}$, $\varphi_s : \mathbb{R}^{P_s} \to \mathbb{R}^{P_s}$ the gradient operator of f_s. Then, function f fits within the class of half-quadratic majorizing constructions initially introduced in [34, 35] and later analysed in [55, 2, 16]. A general structure for the majorant mapping of (78) is thus

$$(\forall \boldsymbol{x} \in \mathbb{R}^N) \quad \boldsymbol{A}(\boldsymbol{x}) = \sum_{s=1}^{S} \boldsymbol{L}_s^\top \underset{1 \leqslant p \leqslant P_s}{\mathrm{Diag}} \left\{ [\omega_s(\boldsymbol{L}_s \boldsymbol{x})]_p \right\} \boldsymbol{L}_s, \tag{80}$$

‡ Function $\boldsymbol{x} \in \mathbb{R}^N \mapsto d^2_E(\boldsymbol{x})$ is 2-Lipschitz differentiable a soon as E is non-empty closed and convex set [4]. Denoting by p_E the orthogonal projection operator, its gradient then corresponds to $\boldsymbol{x} \in \mathbb{R}^N \mapsto 2(\boldsymbol{x} - P_E(\boldsymbol{x}))$.

where, for every $s \in \{1, \dots, S\}$, $\omega_s : \mathbb{R}^{P_s} \to]0, +\infty[^{P_s}$ is a majorizing potential that depends on the properties of $(f_s)_{1 \leqslant s \leqslant S}$ [20, Tab. I]. In our case, for every $s \in \{1, \dots, 4\}$, each of these terms is f_s is β_s-Lipschitz differentiable with

$$\begin{cases} \beta_1 = 1, \\ \beta_2 = 4\eta, \\ \beta_3 = \lambda\delta^{-1}, \\ \beta_4 = 2\kappa. \end{cases} \tag{81}$$

Then, from descent lemma [6], a valid choice is $\omega_s(\cdot) = \alpha\beta_s \mathbf{1}_{P_s}$ with some $\alpha \geqslant 1$ [16]. We adopt this simple strategy for functions f_1, f_2 and f_4, which yields

$$\begin{cases} \omega_1(\cdot) = \alpha \mathbf{1}_N \\ \omega_2(\cdot) = 4\alpha\eta \mathbf{1}_N \\ \omega_4(\cdot) = 2\alpha\kappa \mathbf{1}_N. \end{cases} \tag{82}$$

Regarding function f_3, a more sophisticated majorization is adopted, inherited from half-quadratic strategies [2, 55]:

$$(\forall \boldsymbol{v} \in \mathbb{R}^{2N}) \quad \omega_3(\boldsymbol{v}) = \lambda \begin{bmatrix} \left(1/\sqrt{v_n^2 + v_{N+n}^2 + \delta^2}\right)_{1 \leqslant n \leqslant N} \\ \left(1/\sqrt{v_n^2 + v_{N+n}^2 + \delta^2}\right)_{1 \leqslant n \leqslant N} \end{bmatrix}. \tag{83}$$

A quadratic majorant function satisfying (3) for a given block $\mathcal{S} \in \mathbb{T}$ can then be obtained using (2) with (79) and (80).

5.1.4. Distributed implementation We implement BD3MG algorithm as presented in Section 2.4. Our code is available at https://github.com/mathieuchal/BD3MG. We split the 3D volume into 2D slices along the depth axis $z \in \{1, \dots, N_{\mathsf{Z}}\}$, and consider each 2D slice as an individual block upon which workers can compute an update. Assuming a lexicographic ordering of the voxels, this means that the following partition is adopted:

$$\mathbb{T} = \left\{ [\![(i-1)N_{\mathsf{X}}N_{\mathsf{Y}} + 1, iN_{\mathsf{X}}N_{\mathsf{Y}}]\!] \mid 1 \leqslant i \leqslant N_{\mathsf{Z}} \right\}. \tag{84}$$

BD3MG is implemented on a star graph of workers with a specific master aggregating the current solution. For a given number of active cores $C_{\text{tot}} = C + 1$ of the computer (or of the cluster), one is used as the master process to manage the computation split between the workers while all the $C (= C_{\text{tot}} - 1)$ others, are computing updates asynchronously on planar blocks (i.e., Algorithm 2). We initially set, for every $c \in \{1, \dots, C\}$, $\mathcal{S}_c^0$ corresponding to the index set of the $((c-1)\lfloor \frac{N_{\mathsf{Z}}}{C} \rfloor + 1)$-th 2D slice in the volume. Then, at each iteration k, the master requires worker c^k to process the 2D slice with index set $\mathcal{S}_{c^k}^{k+1}$, by applying a first-in, first-out basis. The worker c^k hence computes the update

for the 2D slice that has been modified the longest time ago, assuming it is available (i.e., not processed in the same time by another worker). A cyclic block update is used as default choice, if several blocks are available (this typically arises in the first iterations). Furthermore, the master controls that each slice has been updated at least once every τ iterations. Regarding data exchange, as emphasized in Section 2.5, in practice, it is not necessary to share the full vector $\boldsymbol{x}$ with all the workers. Consider a worker update associated to the block $\mathcal{S} \in \mathbb{T}$. The worker has to compute $\nabla_{(\mathcal{S})} f(\boldsymbol{x})$ and $\boldsymbol{A}_{(\mathcal{S})}(\boldsymbol{x})$. Because of the structure of (78), these quantities actually only depend on a subpart of vector $\boldsymbol{x}$, defined by $(x_n)_{n \in \mathbb{V}_{\mathcal{S}}}$, with $\mathbb{V}_{\mathcal{S}} \subset [\![1, N]\!]$ a set which has low cardinality compared to the full volume size N. Let us explicit this set for our practical example. The key ingredients to account for are (i) the presence of null entries in the linear operators $(\boldsymbol{L}_s)_{1 \leqslant s \leqslant S}$, (ii) the (almost) separability of operators $(\varphi_s, \omega_s)_{1 \leqslant s \leqslant S}$. We introduce the following sets, for every $s \in \{1, \ldots, S\}$,

$$(\forall n \in \{1, \ldots, N\}) \quad \mathrm{col}_{n,s} = \left\{ p \in \{1, \ldots, P_s\} \quad \text{s.t.} \quad (\boldsymbol{L}_s)_{p,n} \neq 0 \right\}, \tag{85}$$

$$(\forall p \in \{1, \ldots, P_s\}) \quad \mathrm{row}_{p,s} = \left\{ n \in \{1, \ldots, N\} \quad \text{s.t.} \quad (\boldsymbol{L}_s)_{p,n} \neq 0 \right\}, \tag{86}$$

Moreover the separable structures of $(\varphi_s, \omega_s)_{1 \leqslant s \leqslant S}$ ensure that for every $s \in \{1, \ldots, S\}$ and $p \in \{1, \ldots, P_s\}$, there exists a subset $\mathcal{V}_{s,p} \subset [\![1, P_s]\!]$ as well as two functions $\widetilde{\varphi}_{s,p} : \mathbb{R}^{|\mathcal{V}_{s,p}|} \to \mathbb{R}$ and $\widetilde{\omega}_{s,p} : \mathbb{R}^{|\mathcal{V}_{s,p}|} \to (0, +\infty)$ such that

$$(\forall \boldsymbol{v} \in \mathbb{R}^{P_s}) \quad \begin{cases} \left[\varphi_s(\boldsymbol{v})\right]_p = \widetilde{\varphi}_{s,p}(\boldsymbol{v}_{(\mathcal{V}_{s,p})}), \\ \left[\omega_s(\boldsymbol{v})\right]_p = \widetilde{\omega}_{s,p}(\boldsymbol{v}_{(\mathcal{V}_{s,p})}). \end{cases} \tag{87}$$

Considering this, we can now rewrite the expressions $\nabla_{(\mathcal{S})} f(\boldsymbol{x})$ and $\boldsymbol{A}_{(\mathcal{S})}(\boldsymbol{x})$ as

$$(\forall \boldsymbol{x} \in \mathbb{R}^N) \quad \nabla_{(\mathcal{S})} f(\boldsymbol{x}) = \Big(\left[\nabla f(\boldsymbol{x})\right]_i \Big)_{i \in \mathcal{S}}, \tag{88}$$

with, for every $i \in \mathcal{S}$,

$$\big[\nabla f(\boldsymbol{x})\big]_i = \sum_{s=1}^{S} \Big[\boldsymbol{L}_s^\top \varphi_s(\boldsymbol{L}_s \boldsymbol{x})\Big]_i, \tag{89}$$

$$= \sum_{s=1}^{S} \sum_{p=1}^{P_s} (\boldsymbol{L}_s)_{p,i} \big[\varphi_s(\boldsymbol{L}_s \boldsymbol{x})\big]_p, \tag{90}$$

$$= \sum_{s=1}^{S} \sum_{p \in \mathrm{col}_{i,s}} (\boldsymbol{L}_s)_{p,i} \big[\varphi_s(\boldsymbol{L}_s \boldsymbol{x})\big]_p, \tag{91}$$

$$= \sum_{s=1}^{S} \sum_{p \in \mathrm{col}_{i,s}} (\boldsymbol{L}_s)_{p,i} \widetilde{\varphi}_{s,p}([\boldsymbol{L}_s \boldsymbol{x}]_{(\mathcal{B}_{s,p})}), \tag{92}$$

$$= \sum_{s=1}^{S} \sum_{p \in \mathrm{col}_{i,s}} (\boldsymbol{L}_s)_{p,i} \widetilde{\varphi}_{s,p} \left(\left[\sum_{n=1}^{N} (\boldsymbol{L}_s)_{\ell,n} x_n \right]_{\ell \in \mathcal{V}_{s,p}} \right), \tag{93}$$

$$= \sum_{s=1}^{S} \sum_{p \in \mathrm{col}_{i,s}} (\boldsymbol{L}_s)_{p,i} \widetilde{\varphi}_{s,p} \left(\left[\sum_{n \in \mathrm{row}_{\ell,s}} (\boldsymbol{L}_s)_{\ell,n} x_n \right]_{\ell \in \mathcal{V}_{s,p}} \right), \tag{94}$$

Similar computation shows that, for every $(i,j) \in \mathcal{S}^2$,

$$\big[\boldsymbol{A}(\boldsymbol{x})\big]_{i,j} = \sum_{s=1}^{S} \sum_{p \in \mathrm{col}_{i,s} \cap \mathrm{col}_{j,s}} (\boldsymbol{L}_s)_{p,i} (\boldsymbol{L}_s)_{p,j} \widetilde{\omega}_{s,p} \left(\left[\sum_{n \in \mathrm{row}_{\ell,s}} (\boldsymbol{L}_s)_{\ell,n} x_n \right]_{\ell \in \mathcal{V}_{s,p}} \right). \tag{95}$$

Hence, (94)-(95) reflect the fact that the only coordinates of the vector $\boldsymbol{x}$ that are manipulated to compute the gradient and majorant mapping related to block $\mathcal{S}$, belong to $\mathbb{V}_{\mathcal{S}}$ where

$$\mathbb{V}_{\mathcal{S}} = \bigcup_{i \in \mathcal{S}} \bigcup_{s \in \{1,\dots,S\}} \bigcup_{p \in \mathrm{col}_{i,s}} \bigcup_{\ell \in \mathcal{V}_{s,p}} \mathrm{row}_{\ell,s}. \tag{96}$$

Since matrices $(\boldsymbol{L}_s)_{1 \leqslant s \leqslant S}$ are very sparse and functions $(\varphi_s, \omega_s)_{1 \leqslant s \leqslant S}$ close to separable ones, the cardinality of the involved sets in (96) is small so that the memory load for each communication in between master and worker is limited §.

5.1.5. Validity of Assumptions. Let us discuss the validity of Assumptions 1, 2 and 3 for the considered problem and implementation.

§ Let us point out that it is not necessary for the worker to have access to the full operators $(\boldsymbol{L}_s)_{1 \leqslant s \leqslant S}$ but only to the entries involved in expressions (94)-(95). In particular, in our implementation, only the kernel blurs related to the depths in a neighborhood of the 2D slice are shared, reducing again the memory load (see for instance `apply_PSFvar3Dz_block.py` in our shared code folder).

Assumption 1. Function f in (78) is differentiable. Moreover, it has a $\mathcal{L}$-Lipschitzian gradient with $\mathcal{L} = \sum_{s=1}^{S} \beta_s |||\boldsymbol{L}_s|||^2$, where $|||\cdot|||$ denotes the spectral norm over matrices and $(\beta_s)_{1\leqslant s\leqslant S}$ were given in the previous subsection. According to [64, Prop. 2.5], a sufficient condition for f to be coercive is $\mathrm{Ker}(\boldsymbol{H}) = \{\boldsymbol{0}_N\}$. This latter is verified in our experiments, since $\boldsymbol{H}$ is a full-rank operator. Thus, Assumption 1 holds.

Assumption 2. This assumption relates to the practical implementation of BD3MG and requires every subset of variables to be updated within a finite number of iterations. In practice, we introduced a safety check in the master loop, that introduces an idle time if a slice has not been updated within the last τ iterations with τ a predefined value. In our implementation, each worker is in average in charge of $\frac{N_Z}{C}$ 2D slices, of the volume. We thus set $\tau = 2\left\lceil \frac{N_Z}{C} \right\rceil$, that is each worker is allowed to spend, in average twice more time to update one slice than another. Given our block selection rule, with balanced computational load per slide, and relying on first-in, first-out, this situation could only arise if a worker experienced a major delay, which never occurred in our experiments.

Assumption 3. This assumption relates to the majorant mapping. To check this assumption, we proceed in three steps. On the one hand, we have,

$$(\forall \boldsymbol{x} \in \mathbb{R}^N) \quad \boldsymbol{A}(\boldsymbol{x}) \succeq \boldsymbol{L}_2^\top \operatorname*{Diag}_{1\leqslant p\leqslant P_2} \left\{\omega_2(\boldsymbol{L}_2\boldsymbol{x})\right\} \boldsymbol{L}_2 \succeq \alpha\eta \boldsymbol{I}_N. \tag{97}$$

On the other hand, according to definition (80) and those of $\omega_1, \ldots, \omega_4$

$$(\forall \boldsymbol{x} \in \mathbb{R}^N) \quad \boldsymbol{A}(\boldsymbol{x}) \preceq \left(\sum_{s=1}^{S} |||\boldsymbol{L}_s|||^2 \max_{1\leqslant p\leqslant P_s} [\omega_s(\boldsymbol{L}_s\boldsymbol{x})]_p \right) \boldsymbol{I}_N \preceq \overline{\nu} \boldsymbol{I}_N, \tag{98}$$

with

$$\overline{\nu} = \alpha \left(\sum_{s=1}^{S} \beta_s |||\boldsymbol{L}_s|||^2 \right). \tag{99}$$

Considering (97), (98)-(99) and the fact that any sub-matrix $\mathcal{M}_{(\mathcal{S})}$ ($\mathcal{S} \subset [\![1, N]\!]$) of a (symmetric) positive matrix $\boldsymbol{M}$ remains positive, the chosen mapping $\boldsymbol{A}$ thus respects

conditions imposed by Assumption 3(i). Moreover, for all $(\boldsymbol{x}, \boldsymbol{y}) \in (\mathbb{R}^N)^2$,

$$\begin{aligned}
&\boldsymbol{A}(\boldsymbol{x}) - \frac{1}{2}\boldsymbol{A}(\boldsymbol{y}) \\
&= \sum_{s=1}^{S} (\boldsymbol{L}_s)^\top \operatorname*{Diag}_{1\leqslant p\leqslant P_s} \left\{ \left([\omega_s(\boldsymbol{L}_s\boldsymbol{x})]_p - \frac{1}{2}[\omega_s(\boldsymbol{L}_s\boldsymbol{y})]_p \right) \right\} \boldsymbol{L}_s \qquad (100) \\
&= \frac{\alpha}{2} \sum_{s\in\{1,2,4\}} (\boldsymbol{L}_s)^\top \boldsymbol{L}_s + (\boldsymbol{L}_3)^\top \operatorname*{Diag}_{1\leqslant p\leqslant P_3} \left\{ \left([\omega_3(\boldsymbol{L}_3\boldsymbol{x})]_p - \frac{1}{2}[\omega_3(\boldsymbol{L}_3\boldsymbol{y})]_p \right) \right\} \boldsymbol{L}_3 \\
&\geq \frac{\alpha}{2} \boldsymbol{L}_2^\top \boldsymbol{L}_2 + (\boldsymbol{L}_3)^\top \operatorname*{Diag}_{1\leqslant p\leqslant P_3} \left\{ \left([\omega_3(\boldsymbol{L}_3\boldsymbol{x})]_p - \frac{1}{2}[\omega_3(\boldsymbol{L}_3\boldsymbol{y})]_p \right) \right\} \boldsymbol{L}_3 \\
&\geq \eta(\alpha)\boldsymbol{I}_N \text{ with } \eta(\alpha) = \frac{\alpha}{2} - \frac{8\lambda}{2\delta}, \qquad (101)
\end{aligned}$$

as $|||\boldsymbol{L}_3|||^2 = 8$. Under the same previous remark on the block positivity preservation, Assumption 3(ii) is verified considering α large enough (i.e so as for $\eta(\alpha)$ to strictly exceed bound $\frac{\mathcal{L}\sqrt{\tau}(1+\tau)}{2}$). In practice, we opt for $\alpha = 1.1 \times \left(\mathcal{L}\sqrt{\tau}(1+\tau) + \frac{8\lambda}{\delta} \right)$. The associated $\underline{\nu}$ in (27) is $\underline{\nu} = \eta(\alpha) - \frac{\mathcal{L}\sqrt{\tau}(1+\tau)}{2}$).

Convergence result In a nutshell, Assumptions 1-2-3 are fulfilled in our experiments, so that Theorem 4.1 holds. Moreover, function f is semi-algebraic, hence so is the Lyapunov function L (see discussion in Sec. 4.5). Thus, Theorem 4.2 holds.

5.2. Comparative analysis on a controlled scenario

We first set $\overline{\boldsymbol{x}}$ as the 3D microscopic image `FlyBrain`‖ with size $N = N_{\mathsf{X}} \times N_{\mathsf{Y}} \times N_{\mathsf{Z}} = 256 \times 256 \times 57$. The linear operator $\boldsymbol{H}$ models a 3D depth-varying Gaussian blur. For each depth $z \in \{1, \dots, N_{\mathsf{Z}}\}$, the blur kernel is characterized by different variance and rotation parameters $(\sigma_{\mathsf{X}}(z), \sigma_{\mathsf{Y}}(z), \sigma_{\mathsf{Z}}(z), \varphi_{\mathsf{Y}}(z), \varphi_{\mathsf{Z}}(z))$, following the model from [72]. In practice, the values of these five parameters are chosen randomly through a uniform distribution over $[0,3] \times [0,3] \times [0,4] \times [0,2\pi] \times [0,2\pi]$, sampled independently for every z. The support of the blur is then truncated to reach a kernel size of $M = M_{\mathsf{X}} \times M_{\mathsf{Y}} \times M_{\mathsf{Z}} = 5 \times 5 \times 11$, which appears large enough to avoid spurious ringing effects. A zero-mean white Gaussian noise with standard deviation $\sigma = 4 \times 10^{-2}$ is then added to the blurred volume. The regularization parameters $(\lambda, \delta, \kappa, \eta) = (1, 1, 10^{-1}, 10^{-3})$ are chosen empirically so as to maximize the Signal-to-Noise Ratio (SNR) of the restored volume. Moreover, we set $(x_{\min}, x_{\max}) = (0, 1)$, equal to the range of the ground truth image. In order to illustrate the acceleration induced by the proposed BD3MG, we run a comparative analysis between different versions of the optimization scheme, in the spirit of an ablation study. Namely, we propose to compare BD3MG with three methods listed hereafter.

‖ https://imagej.nih.gov/ij/images/

- The 3MG algorithm [16, 17] is considered as the baseline. At each iteration, this algorithm builds the majorant mapping as in Sec. 5.1.3 and computes memory gradient updates on the full volume, without any parallelization.
- The Asynchronous Block Gradient Descent (ABGD) algorithm implements the method from [56]. It performs parallel asynchronous gradient descent updates over the slices of the volume. We adopt here the same parallelization settings as for our BD3MG. Updates correspond to the standard gradient descent on the selected planar blocks, using a fixed step-size μ ensuring convergence of the iterative scheme, namely $\mu = 0.99/(1 + \kappa + 2\lambda/\delta + 2\kappa)$.
- The BP3MG algorithm from [10, 32] runs a synchronous version of BD3MG algorithm. The master process carries out the main loop of [10, Alg 4.3]. At each iteration $k \in \mathbb{N}$, it selects C block indices (following a cyclic rule) and sends to each worker $c \in [\![1, C]\!]$ the required data allowing it to update $\mathcal{S}_c^k$, the associated block. Workers process their block in parallel, wait for each other to finish their tasks, combine their respective updates into a unique vector $(\boldsymbol{x}^j)_{j\in\mathcal{S}_1^k\cup\ldots\cup\mathcal{S}_C^k}$ and finally send the latter to the master. The majorant mapping is set as a block diagonal matrix, allowing synchronous parallel updates, as described in [10]. This approach could be interpreted as a special case of BD3MG with a single worker (potentially composed of several subworkers) sending its update (potentially composed of several sub-updates) to the central process $\mathcal{S}^k = \{\mathcal{S}_c^k\}_{c\in C}$. Thanks to the specific structure of the majorant mapping in BP3MG, there is no mismatch in information between central process and workers in this synchronous version, the delay vector i_k always equals k. Nonetheless, the block diagonal form of the majorant mapping of BP3MG is at the price of a lower quality of approximation of the cost function, which might result in slower convergence.

All methods are implemented in *Python* using the built-in *Multiprocessing* library as well as *Numpy* and *Scipy* for both data manipulation and scientific computing. The experiments of this section are conducted on an Intel® Xeon(R) W-2135 CPU with $C_{\text{tot}} = 12$ cores clocked at 3.70GHz. All the versions were initialized with $\boldsymbol{x}^0 = \mathbf{0}_N$ leading to an initial value $f(\boldsymbol{x}^0) = 91292.92$. For every iteration $k \in \mathbb{N}^*$, we monitor the cost function $f(\boldsymbol{x}^k)$, the normalized increment $\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|/\|\boldsymbol{x}^k\|$, the signal to noise ratio (SNR, in dB) defined as

$$\text{SNR} = 20\log_{10}\left(\frac{\|\overline{\boldsymbol{x}}\|}{\|\overline{\boldsymbol{x}} - \boldsymbol{x}^k\|}\right), \tag{102}$$

and the reconstruction error $\|\overline{\boldsymbol{x}} - \boldsymbol{x}^k\|$. The evolution of these metrics along time for the tested algorithms is displayed on Figure 2. We then set a stopping criterion $\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\| \leqslant \widetilde{\varepsilon}\|\boldsymbol{x}^k\|$. The obtained solution is denoted as $\boldsymbol{x}^{\text{f}}$. We display in Table 1 the metrics for the stopping criterion threshold $\widetilde{\varepsilon} = 10^{-3}$. Table 1 and Figure 2 show that BD3MG exhibits a faster practical convergence than its competitors. Both BD3MG and ABGD are asynchronous distributed schemes, and the former implements an accelerated

version of the gradient descent involved in the latter. The MM metric and the subspace scheme in BD3MG act as catalizers, improving the convergence rate compared to ABGD which relies on a simple steepest descent with fixed stepsize. BP3MG and BD3MG are based on the same inherent optimization scheme 3MG. However, BP3MG uses a simplified block diagonal majorant mapping, and imposes synchronous updates, which might yield idle times. These differences can explain why BD3MG converges faster than BP3MG. Finally, 3MG does not exploit the multicore structure of the computing architecture, and thus shows higher computational time.

Slices of the reconstructed volume are displayed in Figure 3, revealing fine details of the image recovered by the restoration procedure.

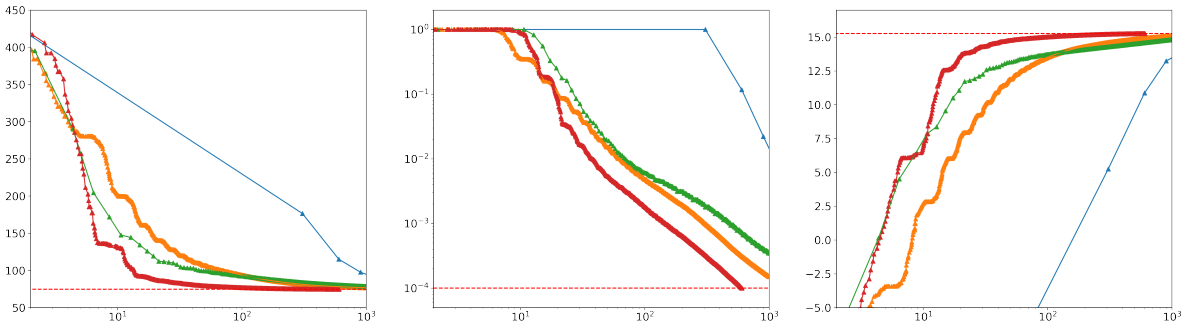


Figure 2: Evolution of quantitative metrics along time (in seconds), for algorithms 3MG (blue), ABGD (orange), BP3MG (green) and BD3MG (red), for `FlyBrain` restoration. Evolution of reconstruction error $\|\boldsymbol{x}^k - \overline{\boldsymbol{x}}\|$ (left), relative increment $\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|/\|\boldsymbol{x}^k\|$ (middle), and SNR in dB (right).

Version	//	Asy.	MM	SNR (dB)	$f(\boldsymbol{x}^{\mathrm{f}})$	Error	Time (× Acc.)
3MG	✗	✗	✓	14.72	1266.04	79.28	1683.79 (-)
ABGD	✓	✓	✗	15.11	1268.80	76.73	305.76 (5.51)
BP3MG	✓	✗	✓	15.21	1264.08	75.33	489.99 (3.44)
BD3MG	✓	✓	✓	**15.26**	**1261.59**	**74.72**	**147.16 (11.44)**

Table 1: Characteristics and performances of compared algorithms on the `Flybrain` restoration task, for reaching the stopping criterion with $\widetilde{\varepsilon} = 10^{-3}$. "//" = Parallel, "Asy." = Asynchronous, "MM" = Majorize-Minimize scheme. Time is in seconds and "× Acc." is the acceleration ratio with respect to 3MG running time.

5.3. Effect of an imbalanced computing power

In order to further demonstrate the advantages of BD3MG over its synchronous counterpart BP3MG, we tested the methods under different computing environments by synthetically modeling stochastic delays in the computing loop of workers. More

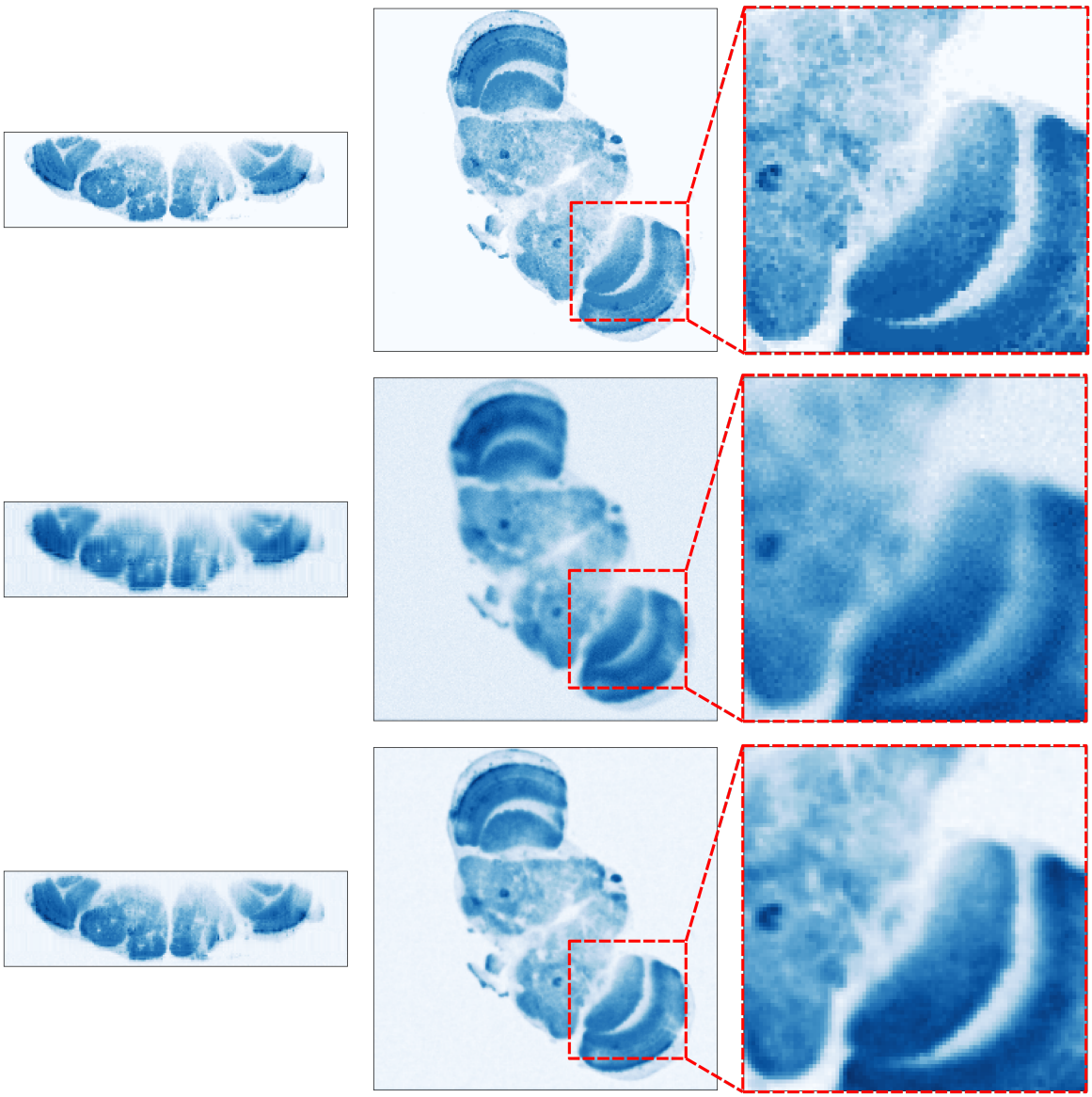


Figure 3: Restoration results of `Flybrain`: ground truth volume (top), degraded version (middle), and results of BD3MG restoration (bottom). Visual comparisons along the $\mathsf{X}-\mathsf{Z}$ axis (left) the $\mathsf{X}-\mathsf{Y}$ axis (middle) and zoomed details (right). The optimization process recovers fine details of the original volume that were lost in its degraded version.

specifically, the same restoration task and computer characteristics than in the previous section is considered, again with $C_{\text{tot}} = 12$ active cores. We introduce artificial perturbation in the computing environment by randomly "freezing" some worker processes for a certain amount of time (i.e., delay) following the three scenarios below:

- **Type I**: One of the workers is consistently affected by a delay that follows a uniform distribution $\mathcal{U}([0,1])$ (in sec.). The other cores are not affected by any delay.
- **Type II**: Two worker cores are not affected by any delay while the others 9 agents

are delayed in the following fashion:

3 cores hold a delay following a uniform distribution $\mathcal{U}([0,1])$.
3 cores hold a delay following a uniform distribution $\mathcal{U}([0,0.5])$.
3 cores hold a delay following a uniform distribution $\mathcal{U}([0,.25])$.

- **Type III**: All worker cores are affected by a delay that follows a uniform distribution $\mathcal{U}([0,1])$.

Method (Scenario)	SNR (dB)	$f(\boldsymbol{x}^f)$	Time (s.)
3MG (no delay)	18.1	1 247.0	1 683.8
BP3MG (Type I)	17.9	1 247.1	623.1
BD3MG (Type I)	18.7	1 246.0	**211.3**
BP3MG (Type II)	17.9	1 247.1	707.9
BD3MG (Type II)	18.7	1 246.0	**220.7**
BP3MG (Type III)	17.9	1 247.1	752.8
BD3MG (Type III)	18.7	1 246.0	**219.9**

Table 2: Performances of BP3MG and BD3MG under imbalanced computed power, for reaching the stopping criterion with $\widetilde{\varepsilon} = 5 \times 10^{-4}$ for `Flybrain` restoration. We additionally provide results for the vanilla 3MG algorithm for sake of comparison.

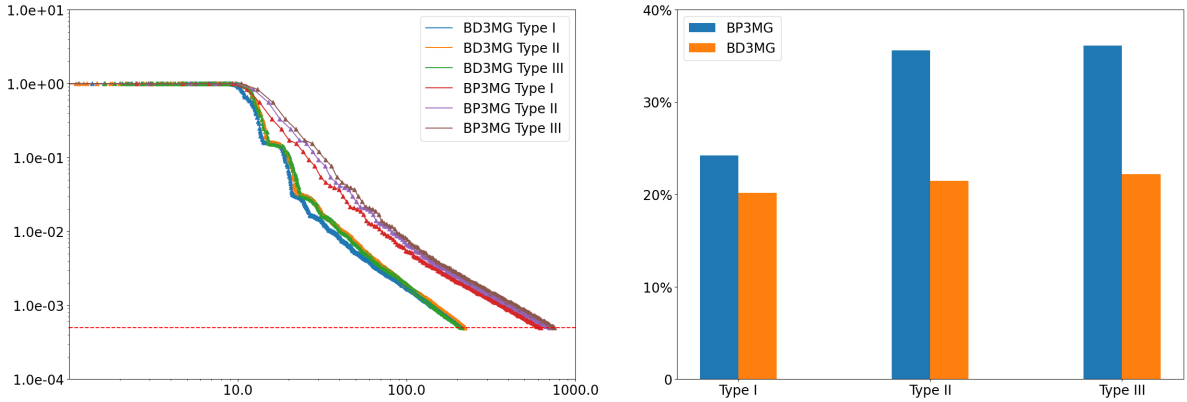


Figure 4: Numerical comparisons between BD3MG and BP3MG for `FlyBrain` restoration under imbalanced computing power: evolution of the relative increment $\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|/\|\boldsymbol{x}^k\|$ along time (in sec.) for each of the three experimental settings in log-log scale (left), and averaged ratio of workers CPU idle time over the entire optimization process for each scenario (right).

The results are summarized in Table 2 and Figure 4. We also report the results of a plain, not delayed, 3MG implementation, for the sake of comparison. In all three scenarios, BD3MG outperforms its synchronous version BP3MG, in terms of computation time while reaching similar final criterion value and SNR. The criteria decrease is faster

for BD3MG which can be explained by two main differences with BP3MG. First, the majorant mapping of BD3MG performs a tighter approximation of the cost function than BP3MG, thus leading intrinsically to improved convergence rate. Second, BD3MG is asynchronous by essence and thus it is resilient to communication delays as soon as they are bounded, as shown by our convergence analysis. In contrast, BP3MG simply waits for all workers to finalize their update, to force the synchronicity, which yields slowdown in case of delayed workers. A more efficient and dynamic handling of the workload is performed in BD3MG, as shown in Figure 4 where CPU idle time is consistently lower for BD3MG than for BP3MG. We note that in asymmetric settings such as (Type II) and (Type III), BD3MG proved to be particularly efficient in reducing the synchronicity constraint of BP3MG for "fast" workers. The comparable results for BD3MG on all three scenarios further suggest that the proposed algorithm is robust to an imbalance in the computing power of workers. Moreover, despite the delayed feedbacks of the workers, both BP3MG and BD3MG remain competitive with respect to the vanilla 3MG, which shows the great interest of a parallel friendly algorithmic structure in this context.

5.4. Scalability assessment.

In order to assess the scalability properties of BD3MG, we further analyse the speed-up generated by the number of cores available. We consider the restoration problem of the 3D image `Aneurysm`¶ of size $N = N_{\mathsf{X}} \times N_{\mathsf{Y}} \times N_{\mathsf{Z}} = 256 \times 256 \times 154$, under the same degradation operator and noise level than in the previous example. Figure 5 presents the acceleration ratio between the required computation time for a single active worker versus the computation time of up to 30 active workers in reaching the stopping criterion with $\widetilde{\varepsilon} = 10^{-3}$. The regularization parameters are set empirically to $(\lambda, \delta, \kappa, \eta) = (1, 1, 10^{-1}, 10^{-3})$ to maximize the final SNR and the same blur kernel than in the previous subsection is used. The computations were performed using HPC resources from the **Oscar** - Ocean State Center for Advanced Resources of the Center for Computation and Visualization, Brown University. The hardware is an Intel Corei9 CPU with up to 48 physical cores at 3.3 GHz GHz and 300G of RAM. Results found in Figure 5 illustrate the great potential of scalability of the proposed algorithm. As the number of core increases, a mild saturation effect is observed (in agreement with Amdahl's law [59]).

5.5. Application to real data from multiphoton microscopy

We finally illustrate the performance of BD3MG on a restoration task of real multiphoton microscopy data specifically acquired for this experiment. Multiphoton microscopy is an interesting solution for the 3D and submicrometric characterization of biomedical structures, it is label-free and contactless [36]. Such a solution takes

¶ http ://www.mathworks.com/matlabcentral/fileexchange /25987-showvol-isosurface-render/

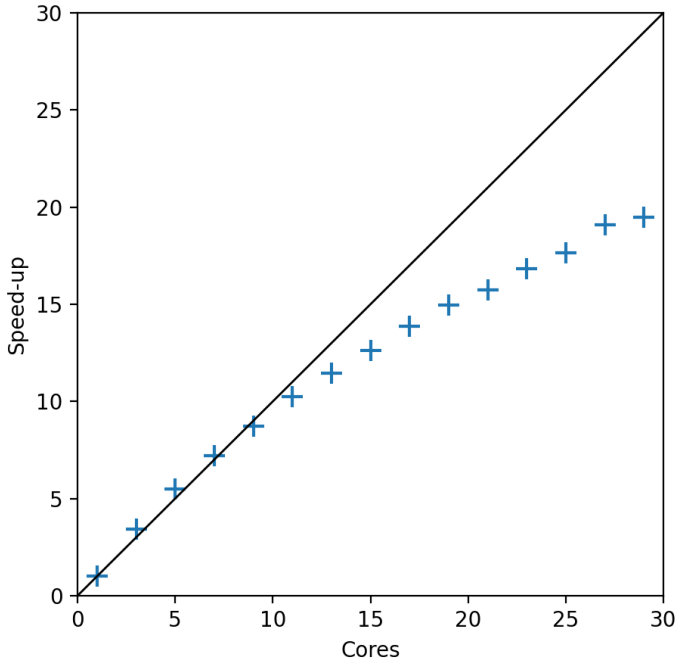


Figure 5: Speed-up ratio of the computation time for 1 to 30 cores for BD3MG for the restoration of `Aneurysm`.

advantage of optical sectioning, an optical property resulting from the nonlinear optical processes involved. 3D images are produced with sub-micrometer resolution without slicing the sample. We use an instrumental solution for acquisition rested on a commercial system from Olympus (BX61WI) coupled with a multiphoton water immersion objective (Olympus XLPLN25XWMP, 25×, NA 1.05). A laser system, emitting femtosecond pulses centred at 810 nm with 10 nm of spectral bandwidth, is used for production of the nonlinear phenomena of second harmonic generation (SHG) and two-photon fluorescence (TPF). The biomedical sample is made of a whole mouse muscle, the Extensor digitorum longus (EDL), isolated from tendon to tendon. Sub-micrometric fluorescent microspheres emitting in the green range are included into the EDL and spread homogeneously all along the whole muscle structure. Under such an experimental protocol, the production of two 3D images is obtained. The first channel contains the SHG from the myosin of the muscle and the second channel displayed the TPF of microspheres used for calibrating the instrumental PSF. A hundred of 2D image slices of SHG and TPF are produced, with 0.1 μm resolution along depth axis Z and $0.049\mu\mathrm{m}\times0.049\mu\mathrm{m}$ resolution over $\mathsf{X}-\mathsf{Y}$ horizontal-vertical axis. The acquisition recording starts 140 μm under the sample surface for a total sample thickness of 180 µm. For this range of depth, the imaging of biological samples is degraded by scattering effects. Both raw volumes (i.e., SHG and TPF) dimension have $2048\times2048\times100$ voxels, from which we extract a subpart with size $N = N_{\mathsf{X}} \times N_{\mathsf{Y}} \times N_{\mathsf{Z}} = 256 \times 256 \times 100$ voxels

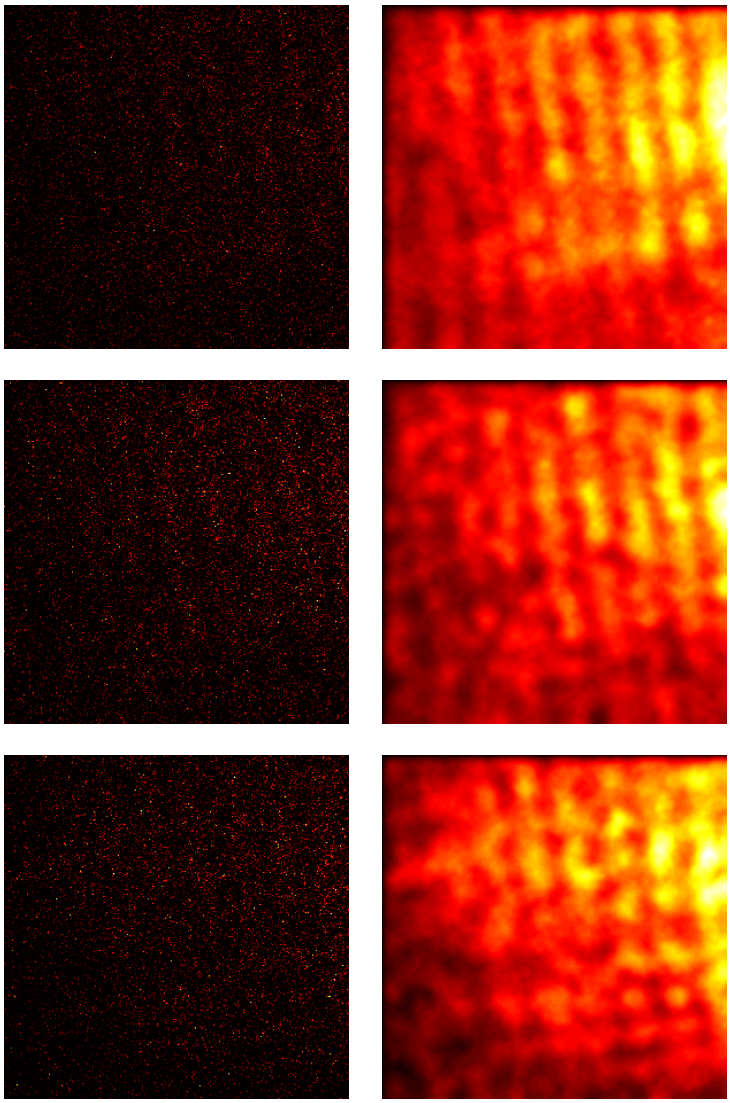


Figure 6: Slices $(12,5\mu m \times 12,5\mu m)$ for depths $z = 5, 25$ and 70 (from top to bottom) of the original acquisition (left) and after restoration (right). The comparisons show that the definition of the muscular structure has been enhanced by the reconstruction.

for the purpose of our study.

We follow the computational pipeline FAMOUS previously introduced in [48]. We estimate a depth-variant Gaussian PSF field within the 3D microscopic volume by applying the 3D Gaussian fitting algorithm FIGARO from [22] to volume of interests extracted from the second image channel, displaying fluorescence of calibrated microbeads. Each volume of interest is selected through an automatic search of connected components within a filtered and binarized version of the observed volume. Then, FIGARO method is ran, yielding parameters (i.e., mean, covariance, scaling, shift) of a 3D Gaussian shape. This allows to build, through a simple interpolation strategy, a model for a depth-variant PSF with truncated support of size $M =$

$M_{\mathsf{X}} \times M_{\mathsf{Y}} \times M_{\mathsf{Z}} = 21 \times 21 \times 21$ (see more details in [48, Sec.2.4]). Since no ground truth is available, the regularization parameters $(\lambda, \delta, \kappa, \eta) = (10^2, 2, 10, 10^{-3})$, are selected by retrospective visual inspection. The reconstruction shown in Figure 6 exhibits clear contrasts and sharpness properties. Comparative videos of the original and restored volume are available at https://github.com/mathieuchal/BD3MG. The native signal from the raw image was presenting a high level of noise and blur due to the presence of scattering elements all along the 140 µm of sample depth. Thanks to the proposed restoration strategy, the localisation of the myosin in the muscle sample is made possible, and the spatial organization of this protein into the down side of the EDL is revealed. The volume restoration took 305 seconds and $\sim$ 2000 iterations on a $C_{\text{tot}} = 12$ cores setting, when setting $\widetilde{\varepsilon} = 10^{-3}$.

6. Conclusion

In this paper, we have presented a new block distributed Majorize-Minimize algorithm, BD3MG, devised to tackle large-size differentiable optimization problems met in a wide range of applications. Our main contribution lies in a distributed asynchronous formulation that allows for delays in the current solution computed between workers, while securing convergence guarantees under mild assumptions. Our new algorithm BD3MG has been tested in the context of 3D image restoration with depth-variant blur. Experimental results underlined the speedup potential of this method and its concrete applicability in the field of fluorescence microscopy. Future work will be dedicated to extension to more general distributed graph topologies.

Declarations

Funding: This research work received funding support from the European Research Council Starting Grant MAJORIS ERC-2019-STG-850925.
Competing interests: The authors have no relevant financial or non-financial interests to disclose.

Bibliography

[1] F. Abboud, M. Stamm, E. Chouzenoux, J.-C. Pesquet, and H. Talbot. Distributed algorithms for proximity operator computation with applications to video processing. *Digital Signal Processing*, 2022. To appear.

[2] M. Allain, J. Idier, and Y. Goussard. On global and local convergence of half-quadratic algorithms. *IEEE Transactions on Image Processing*, 15(5):1130–1142, May 2006.

[3] H. Attouch, J. Bolte, P. Redont, and A. 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.

[4] H. Bauschke and P. L. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer, 2011.

[5] M. Bertero. *Introduction to Inverse Problems in Imaging*. CRC Press, 2017.

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

[7] J. Bochnak, M. Coste, and M.-F. Roy. *Real Algebraic Geometry*, volume 36. Springer Science & Business Media, 2013.

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

[9] S. Bonettini, M. Prato, and S. Rebegoldi. A block coordinate variable metric linesearch based proximal gradient method. *Computational Optimization and Applications*, 71(1):5–52, 2018.

[10] S. Cadoni, E. Chouzenoux, J.-C. Pesquet, and C. Chaux. A block parallel majorize-minimize memory gradient algorithm. In *Proceedings of the 23rd IEEE International Conference on Image Processing (ICIP 2016)*, pages 3194–3198, Phoenix, AZ, 25-28 Sep. 2016.

[11] L. Cannelli, F. Facchinei, G. Scutari, and V. Kungurtsev. Asynchronous optimization over graphs: Linear convergence under error bound conditions. *IEEE Transactions on Automatic Control*, 66(10):4604–4619, 2020.

[12] A. Chakrabarti, T. Zickler, and W. T. Freeman. Analyzing spatially-varying blur. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2010)*, pages 2512–2519, San Francisco, CA, USA, 13-18 June 2010.

[13] M. Chalvidal and E. Chouzenoux. Block distributed 3MG algorithm and its application to 3D image restoration. In *Proceedings of the IEEE International Conference on Image Processing (ICIP 2020)*, pages 938–942, Abu Dhabi, United Arab Emirates (virtual), 25-28 Oct. 2020.

[14] F. Chorobura and I. Necoara. Random coordinate descent methods for nonseparable composite optimization. Technical report, 2022. https://arxiv.org/abs/2203.14368.

[15] E. Chouzenoux and J.-B. Fest. SABRINA: a stochastic subspace majorization-minimization algorithm. Technical report, 2021. https://optimization-online.org/2021/08/8573/.

[16] E. Chouzenoux, J. Idier, and S. Moussaoui. A majorize–minimize strategy for subspace optimization applied to image restoration. *IEEE Transactions on Image Processing*, 20(6):1517–1528, 2010.

[17] E. Chouzenoux, A. Jezierska, J.-C. Pesquet, and H. Talbot. A majorize-minimize subspace approach for $\ell_2 - \ell_0$ image regularization. *SIAM Journal on Imaging Sciences*, 6(1):563–591, Jan 2013.

[18] E. Chouzenoux, S. Martin, and J.-C. Pesquet. A local MM subspace method for solving constrained variational problems in image recovery. *Journal of Mathematical Imaging and Vision*, 2022. to appear.

[19] E. Chouzenoux and J.-C. Pesquet. Convergence rate analysis of the majorize-minimize subspace algorithm. *IEEE Signal Processing Letters*, 23(9):1284–1288, Sep. 2016.

[20] E. Chouzenoux and J.-C. Pesquet. A stochastic majorize-minimize subspace algorithm for online penalized least squares estimation. *IEEE Transactions on Signal Processing*, 65(18):4770–4783, 2017.

[21] E. Chouzenoux, J.-C. Pesquet, and A. Repetti. A block coordinate variable metric forward-backward algorithm. *Journal of Global Optimization*, pages 1–29, Feb. 2016.

[22] E. Chouzenoux, T. Tsz-Kit Lau, C. Lefort, and J.-C. Pesquet. Optimal multivariate Gaussian fitting with applications to PSF modeling in two-photon microscopy imaging. *Journal of Mathematical Imaging and Vision*, 61(7):1037–1050, Sept. 2019.

[23] J. Chung, S. Knepper, and J. G. Nagy. *Large-Scale Inverse Problems in Imaging*, pages 47–90. Springer New York, New York, NY, 2015.

[24] D. Davis. The asynchronous PALM algorithm for nonsmooth nonconvex problems. Technical report, 2016. https://arxiv.org/abs/1604.00526.

[25] D. Davis, M. Udell, and B. Edmunds. The sound of APALM clapping: Faster nonsmooth nonconvex optimization with stochastic asynchronous palm. In *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS 2016)*, Barcelona, Spain, 5-10 Dec. 2016.

[26] L. Denis, E. Thiébaut, F. Soulez, J. M. Becker, and R. Mourya. Fast approximations of shift-variant blur. *International Journal of Computer Vision*, 115:253–278, 2015.

[27] L. Denis, E. Thiébaut, and F. Soulez. Fast model of space-variant blurring and its application to deconvolution in astronomy. In *2011 18th IEEE International Conference on Image Processing*, pages 2817–2820. IEEE, 2011.

[28] L. Duval, E. Chouzenoux, A. Repetti, M. Q. Pham, and J.-C. Pesquet. Euclid in a taxicab: Sparse blind deconvolution with smoothed l1/l2 regularization. *IEEE Signal Processing Letters*, 22(5):539–543, May 2015.

[29] P. Escande and P. Weiss. Sparse wavelet representations of spatially varying blurring operators. *SIAM Journal on Imaging Sciences*, 8:2976–3014, 2015.

[30] T. Fan and T. Murphey. Majorization minimization methods for distributed pose graph optimization with convergence guarantees. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2020)*, pages 5058–5065, Las Vegas, USA, 2020.

[31] J. A. Fessler. Grouped coordinate descent algorithms for robust edge-preserving image restoration. In T. J. Schulz, editor, *Image Reconstruction and Restoration II, International Society for Optics and Photonics*, volume 3170, pages 184–194, 1997.

[32] J.-B. Fest and E. Chouzenoux. Convergence analysis of block majorize-minimize subspace approaches. Technical report, 2021. https://optimization-online.org/2021/12/8711/.

[33] A. Florescu, E. Chouzenoux, J.-C. Pesquet, P. Ciuciu, and S. Ciochina. A majorize-minimize memory gradient method for complex-valued inverse problems. *Signal Processing*, 103:285–295, 2014.

[34] D. Geman and G. Reynolds. Constrained restoration and the recovery of discontinuities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(3):367–383, 1992.

[35] D. Geman and C. Yang. Nonlinear image recovery with half-quadratic regularization. *IEEE transactions on Image Processing*, 4(7):932–946, 1995.

[36] W. Göbel, B. M. Kampa, and F. Helmchen. Imaging cellular network dynamics in three dimensions using fast 3d laser scanning. *Nature methods*, 4(1):73–79, 2007.

[37] D. Grishchenko, F. Iutzeler, J. Malick, and M.-R. Amini. Asynchronous distributed learning with sparse communications and identification. Technical report, 2018. https://arxiv.org/abs/1812.03871.

[38] S. B. Hadj and L. Blanc-Féraud. Modeling and removing depth variant blur in 3d fluorescence microscopy. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 689–692. IEEE, 2012.

[39] R. Hannah and W. Yin. On unbounded delays in asynchronous parallel fixed-point algorithms. *Journal of Scientific Computing*, 76(1):299–326, Dec 2017.

[40] M. Hong, M. Razaviyayn, Z. Luo, and J. Pang. A unified algorithmic framework for block-structured optimization involving big data: With applications in machine learning and signal processing. *IEEE Signal Processing Magazine*, 33(1):57–77, Jan 2016.

[41] S. Horváth, M. Sanjabi, L. Xiao, P. Richtárik, and M. Rabbat. Fedshuffle: Recipes for better use of local work in federated learning. Technical report, 2022. https://arxiv.org/abs/2204.13169.

[42] D. R. Hunter and K. Lange. A tutorial on MM algorithms. *The American Statistician*, 58(1):30–37, 2004.

[43] M. W. Jacobson and J. A. Fessler. An expanded theoretical treatment of iteration-dependent majorize-minimize algorithms. *IEEE Transactions on Image Processing*, 16(10):2411–2422, Oct. 2007.

[44] A. Jezierska, H. Talbot, and J.C.Pesquet. Spatially variant psf modeling in confocal macroscopy. In *Proceedings of the 15th IEEE International Symposium on Biomedical Imaging (ISBI 2018)*, pages 489–492. Washington, DC, USA, 4-7 April 2018.

[45] B. Kim and T. Naemura. Blind depth-variant deconvolution of 3d data in wide-field fluorescence microscopy. *Scientific reports*, 5(1):1–9, 2015.

[46] J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. Technical report, 2016. https://arxiv.org/abs/1610.02527.

[47] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Proceedings of the 13th International Conference on Neural Information Processing Systems (NIPS 2000)*, page 535541, Denver, Colorado, 2000.

[48] C. Lefort, M. Chalvidal, A. Parente, V. Blanquet, H. Massias, L. Magnol, and E. Chouzenoux. FAMOUS: a fast instrumental and computational pipeline for multiphoton microscopy applied to 3d imaging of muscle ultrastructure. *Journal of Physics D: Applied Physics*, 54(27):274005, 2021.

[49] Z. Li, Z. Dong, Z. Liang, and Z. Ding. Surrogate-based distributed optimisation for expensive black-box functions. *Automatica*, 125:109407, 2021.

[50] X. Lian, Y. Huang, Y. Li, and J. Liu. Asynchronous parallel stochastic gradient for nonconvex optimization. Technical report, 2015. https://arxiv.org/abs/1506.08272.

[51] N. Loizou and P. Richtárik. Revisiting randomized gossip algorithms: General framework, convergence rates and novel block and accelerated protocols. *IEEE Transactions on Information Theory*, 67(12):8300–8324, 2021.

[52] K. Mishchenko, F. Iutzeler, and J. Malick. A distributed flexible delay-tolerant proximal gradient algorithm. *SIAM Journal on Optimization*, 30(1):933–959, 2020.

[53] J. Nagy and D. O'Leary. Restoring images degraded by spatially variant blur. *SIAM Journal on Scientific Computing*, 19(4):1063–1082, 1998.

[54] Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *Université catholique de Louvain, Center for Operations Research and Econometrics (CORE), CORE Discussion Papers*, 22, Jan. 2010.

[55] M. Nikolova and M.-K. Ng. Analysis of half-quadratic minimization methods for signal and image recovery. *SIAM J. Sci. Comput.*, 27:937–966, 2005.

[56] F. Niu, B. Recht, C. Re, and S. Wright. Hogwild: A lockfree approach to parallelizing stochastic gradient descent. In *Proceedings of the 25th Conference on Advances in Neural Information Processing Systems (NIPS 2011)*, pages 693–701, Granada, Spain, 12-17 Dec. 2011.

[57] J. Nutini, M. Schmidt, I. H. Laradji, M. Friedlander, and H. Koepke. Coordinate descent converges faster with the Gauss-Southwell rule than random selection. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning (ICML 2015)*, volume 37, page 1632–1641, 2015.

[58] A. Onose, R. E. Carrillo, A. Repetti, J. D. McEwen, J.-T. Thiran, J.-C. Pesquet, and Y. Wiaux. Scalable splitting algorithms for big-data interferometric imaging in the SKA era. *Monthly Notices of the Royal Astronomical Society*, 462(4):4314–4335, 2016.

[59] D. A. Patterson, J. L. Hennessy, and D. Goldberg. *Computer architecture: a quantitative approach*, volume 2. Morgan Kaufmann San Mateo, CA, 1990.

[60] J.-C. Pesquet and A. Repetti. A class of randomized primal-dual algorithms for distributed optimization. *Journal of Nonlinear and Convex Analysis*, 16(12):2353–2490, 2014.

[61] M. Prato, A. La Camera, S. Bonettini, S. Rebegoldi, M. Bertero, and P. Boccacci. A blind deconvolution method for ground based telescopes and Fizeau interferometers. *New Astronomy*, 40:1–13, 2015.

[62] C. Preza and J.-A. Conchello. Depth-variant maximum-likelihood restoration for three-dimensional fluorescence microscopy. *JOSA A*, 21(9):1593–1601, 2004.

[63] P. Richtárik and M. Takáč. Parallel coordinate descent methods for big data optimization. *Mathematical Programming*, 156(1-2):433–484, Apr 2015.

[64] M. C. Robini and Y. Zhu. Generic half-quadratic optimization for image reconstruction. *SIAM Journal on Imaging Sciences*, 8(3):1752–1797, 2015.

[65] S. Sotthivirat and J. A. Fessler. Image recovery using partitioned-separable paraboloidal surrogate coordinate ascent algorithms. *IEEE Transactions on Signal Processing*, 11(3):306–317, 2002.

[66] Y. Sun, P. Babu, and D. P. Palomar. Majorization-minimization algorithms in signal processing, communications, and machine learning. *IEEE Transactions on Signal Processing*, 65(3):794–816, 2016.

[67] P. Thouvenin, N. Dobigeon, and J.-Y. Tourneret. Partially asynchronous distributed unmixing of hyperspectral images. *IEEE Transactions on Geoscience and Remote Sensing*, 57(4), Apr. 2019.

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

[69] J. Tuck, D. Hallac, and S. Boyd. Distributed majorization-minimization for Laplacian regularized problems. *IEEE/CAA Journal of Automatica Sinica*, 6(1):45–52, 2019.

[70] J. Wang, Z. Charles, Z. Xu, G. Joshi, H. B. McMahan, M. Al-Shedivat, G. Andrew, S. Avestimehr, K. Daly, D. Data, et al. A field guide to federated optimization. Technical report, 2021. https://arxiv.org/abs/2107.06917.

[71] A. Wilson. *Lyapunov Arguments in Optimization*. PhD thesis, University of California, Berkeley, 2018.

[72] O. Wirjadi and T. Breuel. Approximate separable 3D anisotropic Gauss filter. In *Proceedings of the 12nd IEEE International Conference on Image Processing (ICIP 2005)*, volume 2, pages 149–52, Genoa, Italy, 11-14 Sept 2005.

[73] T. Yang, X. Yi, J. Wu, Y. Yuan, D. Wu, Z. Meng, Y. Hong, H. Wang, Z. Lin, and K. H. Johansson. A survey of distributed optimization. *Annual Reviews in Control*, 47:278–305, 2019.

[74] S. Zavriev and F. Kostyuk. Heavy-ball method in nonconvex optimization problems. *Computational Mathematics and Modeling*, 4(4):336–341, 1993.

[75] R. Zhang and J. T. Kwok. Asynchronous distributed ADMM for consensus optimization. In *Proceedings of the 31st International Conference on International Conference on Machine Learning (ICML 2014)*, volume 32, 2014.

[76] Z. Zhang, J. T. Kwok, and D.-Y. Yeung. Surrogate maximization/minimization algorithms and extensions. *Machine Learning*, 69:1–33, 2007.

[77] Y. Zheng and Q. Liu. A review of distributed optimization: Problems, models and algorithms. *Neurocomputing*, 483:446–459, 2021.