Stochastic convergence of parallel asynchronous adaptive first-order methods

S. Gratton* and Ph. L. Toint†

29 V 2026

Abstract

A new class of asynchronous adaptive first-order optimization methods is introduced, comprising asynchronous variants of several popular algorithms. Versions of these methods using momentum and/or inexact normalization are also considered. The convergence of methods in the class on non-convex functions is analyzed in a fully stochastic setting, and is shown to be (up to logarithmic factors) of order $\mathcal{O}(1/\sqrt{t})$ under reasonable assumptions. Numerical experiments suggest that such asynchronous adaptive algorithms are very relevant in heterogeneous large-scale machine learning systems.

Keywords: Unconstrained nonconvex optimization, first-order methods, global rate of convergence, asynchronous methods, parallel computing, heterogeneous deep learning problems.

1 Introduction

[55] [38]

We consider the problem of finding a first-order critical point for the optimization problem

$$\min_{X\in\mathbb{R}^N} F(X) = \mathbb{E}[f(X,\xi)] \tag{1.1}$$

where f is a smooth, possibly nonconvex function of X and ξ is a suitably defined random variable. In other words, we are seeking X such that $\nabla_X \mathbb{E}[f(X,\xi)] = 0$. Motivated by the growing importance of very large models in deep learning, a substantial literature has been devoted to the study of asynchronous algorithms for solving (1.1). By far the most popular are stochastic asynchronous steepest descent methods (also called asynchronous gradient methods), whose modern efficient lock-free framework has been pioneered by the Hogwild! method [43]. This method, like most of the methods we discuss in this paper, maintains a vector $\widehat{X}$ of values for the problem's variables in a memory shared by a number of processes/processors, each of which sequentially

1. selects a subset or block of the variables from $\widehat{X}$,
2. computes the gradient of f at $\widehat{X}$ with respect to the variables of the block (this requires reading the vector $\widehat{X}$ from shared memory),
3. computes a gradient step (with method-specific stepsize) in the subspace spanned by this block using the block gradient, resulting in new values of the block variables,
4. updates the values of $\widehat{X}$ corresponding to the block with these new values.

*Université de Toulouse, INP, IRIT, Toulouse, France. Email: serge.gratton@enseeiht.fr. Work partially supported by 3IA Artificial and Natural Intelligence Toulouse Institute (ANITI), French "Investing for the Future - PIA3" program under the Grant agreement ANR-19-PI3A-0004"

†Université de Toulouse, INP, IRIT, Toulouse, France, and NAXYS, University of Namur, Namur, Belgium. Email: philippe.toint@unamur.be

The whole operation is asynchronous because each process/processor does not wait for other processes to complete their tasks (it is *lock-free*). Of course, this description is extremely synthetic and methods of the type just described may still differ on a number of issues.

Geometry. The first is to define how a particular block of variables is selected in Step 1. This selection can be random or deterministic, and is characterized by the size of the allowed "overlap" between blocks selected by different processes, ranging from totally separable (the blocks associated with running processes are disjoint), partially separable (limited overlap between running blocks is allowed — for an introduction to partial separability, see [19, 20, 11]), or arbitrary (the size of the overlap is not constrained).

Atomicity. The next issue is the management of the read/write operations between local processes and shared memory. Because of the asynchronous nature of the whole algorithm, it may happen that different processes attempt to read or write the same component of $\widehat{X}$ at the same time. To cope with this problem, it is generally assumed that the reading and writing operations are *atomic* at some granularity level, meaning that reading/writing an atomic component of $\widehat{X}$ by a process cannot be interrupted by another process. We will see below that this concept also applies to other quantities than $\widehat{X}$, and that, in addition to purely hardware and message passing considerations, it also can be limited by the internal structure of the object to read/write, or by the norm used to measure it (for instance, objects like positive definite matrices need to maintain their positive definite nature).

Consistency. In addition, it may well happen that a process is reading parts of $\widehat{X}$ while other parts are being updated by other processes. This blurs the notion of algorithm's iterates, because it is possible that a block gradient is computed for a value of the variables corresponding to an incomplete, ongoing update of $\widehat{X}$. Most algorithms (but not all) allow these *inconsistent* read/writes.

Delays. In addition, the asynchrony of the whole algorithm implies that when a block gradient is computed, it uses the values of $\widehat{X}$ read from shared memory, but these values may result from writes (by other blocks) somewhat in the past. The time difference between the moment $\widehat{X}$ is updated and the moment where it is used for computing a block gradient is called a *delay*. The analysis of all algorithms considered here imposes some kind of upper bound on such delays, either explicitly or in expectation.

Choice of stepsize. Finally, once the block gradient is computed, yielding a descent direction, a stepsize along this direction must be defined. Algorithms in the literature either assume that the stepsize is a "suitably small" constant, or that it is given by a (user) predefined decreasing sequence converging to zero.

Having brushed the landscape, we now review chronologically significant proposals in the literature. The pioneering Hogwild! method proposed in [43] was instrumental in introducing lock-free variants of asynchronous steepest-descent. It applies to strongly convex functions with partially separable geometry, for which block gradients are computable. Its stochastic convergence analysis assumes that the gradient oracles are bounded and that the constant stepsize can be chosen small enough compared to the gradient's Lipschitz constant. Inconsistent reads are allowed. This is not the case for the AsySG-con [33] which, as the end of its name suggests, insists on coherent reads, but applies to possibly nonconvex functions with full-size gradients without any separability requirement. Its analysis assumes bounded gradients and predefined diminishing stepsizes. The ADrock algorithm [44] is of a slightly different type, as it is better viewed as a fixed point method, but it faces similar issues. It uses full gradient vectors and requires that stepsizes decrease to zero. Its analysis holds for convex functions with bounded gradients but without any specific separability. The KroMagnon algorithm [36] is similar to Hogwild! in that it is a gradient-based method, allows block gradient evaluations and requires a partially separable geometry. Establishing its rate of convergence needs convex functions but no bound on the gradients is necessary. The requirement on partially separable geometry is relaxed for the versions of Hogwild! studied in [42, 41] and boundedness of the gradients is no longer requested. Instead of assuming a small enough stepsize as in the original version, the convergence analysis requires a stepsize decreasing to zero. ASAGA [32]

	Problem		Algorithm's characteristics				Assumptions		
	convex	separable geometry	method type	stoch.	shared info	cons. read	step size	bounded gradients	bounded delay
Hogwild! [43]	strong	partially	gradient	yes	$\widehat{X}$	no	small	yes	yes
AsySG-con [33]	no	no	gradient	yes	$\widehat{X}$	yes	$\searrow 0$	no	yes
"SGD" [9]	yes	no	gradient	yes	$\widehat{X}$	no	$\searrow 0$	yes	in $\mathbb{E}$
ARock [44]	yes	no	fixed point	yes	$\widehat{X}$	no	small	yes	yes
KroMagnon [36]	yes	partially	gradient	yes	$\widehat{X}$	no	small	no	yes
Hogwild! [42, 41]	strong	relaxed	gradient	yes	$\widehat{X}$	no	$\searrow 0$	no	yes
ASAGA [32]	strong	partially	gradient	yes	$\widehat{X}, (\widehat{G})$	no	small	no	yes
FDG [55]	no	yes	gradient	yes	$\widehat{X}$	no	small	yes	yes
AsyFLEXA [8]	no	no	convex approx.	no	$\widehat{X}$	no	small	no	yes
PASSM [29, 30]	no	yes	gradient+mom	yes	$\widehat{X}$	no	$\searrow 0$	in $\mathbb{E}$	in $\mathbb{E}$
DADAO [38]	yes	no	gradient	yes	$\widehat{X}$	no	small	no	yes
this paper	no	yes	prec. grad.+mom	yes	$\widehat{X}, (\widehat{\Pi})$	no	adapt.	no	yes
this paper	no	yes	prec. grad.+mom	yes	$\widehat{X}, (\widehat{\Pi}, \widehat{M})$	no	adapt.	no	yes

Table 1: A summary of the relevant problem class, algorithm's characteristics and conditions of analysis for stochastic asynchronous gradient methods. Brackets around items in the "shared info" columns indicate the items are not mandatory.

is another gradient-based method requiring, as is the case for the original Hogwild!, partially separable geometry, strongly convex functions and a sufficiently small stepsize. However, there is no need to assume bounded gradients. The method occasionally evaluates the full gradient (in shared memory) in order to reduce the variance of the estimates. The FDG algorithm of [55] partitions the variables into blocks and interestingly proposes to exploit the structure of backpropagation in neural networks to compute the gradients with respect to the variables of a block independently from those of another, in turns leading to an asynchronous delayed-gradient method. The analysis stops with a descent lemma requiring bounded gradients and knowledge of the gradient's Lipschitz constant. The AsyFLEXA [8] deterministic algorithm is again slightly different in that it relies on convex local models (instead of purely linear ones as is the case for most gradient-based methods) and applies to possibly nonconvex functions with block gradient evaluations without separability requirement. Once more, the analysis is carried out for small enough stepsize, but without assuming bounded gradients. The reference [30] discusses several variants of a stochastic gradient-based method with momentum, as seen in the context of continuous differential equations. We report here on PASSM, a variant for potentially nonconvex totally separable functions with block gradient evaluations, which appears to perform best. Bounded gradients are required for its analysis and its stepsizes follow a geometrically decreasing sequence. Interestingly, a lock is required for writing the updated variables in shared memory. One should also mention contributions of [39] and [14]. The first introduces the concept of "elastic consistency", formalizing the conditions on delays for non-adaptive gradient methods and showing that such conditions are necessary for convergence of this type of methods. The second introduces general recurrence occurring in many non-adaptive asynchronous stochastic gradient methods and uses them to sharpen existing convergence results, in particular by considering average delays rather than maximum ones. Finally, the proposal of [38] considers an asynchronous method for stricly convex functions with known Lipschitz constant, putting special emphasis on communication costs.

Table 1 on this page provides an easy-to-read summary of the above discussion, where column "stoch." indicates whether the method is stochastic, and "cons. read" indicates whether the method requires consistent reading of shared memory. In the "method type" column, the abbreviations "convex approx", "prec. grad." and "mom" stand for "convex approximation", "preconditioned gradient" and "momentum", respectively. Finally, the abbreviation "adapt." in the "stepsize" column refers to adaptive stepsize, the technique we will develop below.

All the algorithms we have reviewed so far exhibit two (in our view, significant) limitations. The first and most important one is that pure gradient steps can lead to very slow convergence

on (even moderately) ill-conditioned problems. This issue has long been addressed by the use of steepest-descent steps (a norm-dependent concept, at variance with gradient steps) and (adaptive) preconditioning (see [7, Section 9.4], for instance). While these techniques have successfully been applied for solving problem (1.1) in the synchronous case, for instance in methods like AdaGrad [12, 37, 50], Adam [27], Shampoo [22, 48] or Muon [26, 46, 54] or in the distributed case [34], they haven't yet (to the authors' knowledge) been considered for asynchronous optimization. The second limitation is that theoretical analysis of many of the published algorithms ignore the use of momentum, despite its widespread nature (see [34], for instance).

The main contribution of the present paper is to propose and analyze a (to the authors' knowledge, first ever) class of algorithms containing several *stochastic asynchronous adaptively preconditioned gradient methods*, possibly *using momentum*, for the nonconvex case. In particular, AdaNorm, full and diagonal AdaGrad as well as adaptive variants of Shampoo and Muon are covered. Like for most recent proposals, bounded gradients are not necessary for its analysis. The proposed algorithm builds on the ADPREC framework for (synchronous) stochastic first-order methods [18], which is itself based on the use of general dual norms and covers a large class of stochastic adaptive gradient-based methods. The present proposal takes advantage of its powerful generality, and is defined in the geometric setting of totally separable problems, as it is the case for PASSM. While this might be seen as theoretically restrictive, it still covers the very important case of layer-wise preconditioning in training large neural networks (see [53, 16, 45, 2] for instance), hence justifying our interest. As a side-benefit, our proposal also extends the theory presented in [18] in allowing *inexact step normalization strategies*, a clear advantage in practice for methods like Muon where the normalization is performed using a possibly inexact Newton-Schultz iteration [28, 5, 3, 48, 34]. The proposed addition of momentum to asynchronous steepest-descent methods (the second line for our paper in Table 1) is also useful to enhance numerical performance, but it may come with a (mostly theoretical) cost. For the common momentum definition of using the gradient, assuming that the step-size is sufficiently small may be necessary to prove convergence, but the resulting degraded convergence rate may improved by adopting a suitable strategy for the momentum parameter. Remarkably, no assumption on the step-size is required for the momentum-less variant, or for a definition of the momentum using the momentum itself.

The paper is organized as follows. The momentum-less asynchronous framework is presented in Section 2, where it is shown to be a special case of an easier-to-analyze delayed framework, whose rate of convergence is studied in Section 3. Two different variants of momentum are introduced and the resulting rates of convergence examined in Section 4. Numerical experiments illustrating the use of asynchronous block-wise adaptive Muon/AdaGrad are presented in Section 6. Finally, some conclusions and perspectives are discussed in Section 7 while Appendix 1 and Appendix 2 give the details of the convergence proofs for the methods of Sections 2 and 3, respectively.

Notations: In what follows, $\|\cdot\|_E$ denotes the Euclidean norm. If $\|\cdot\|$ is a norm on some space $\mathcal{S}$ endowed with an inner product $\langle\cdot,\cdot\rangle$, its dual norm $\|\cdot\|_D$ is defined by $\|x\|_D = \max_{y\in\mathcal{S}}\langle y, x\rangle/\|y\|$. The symbol I_n denotes the identity matrix of dimension n and 0_n the zero vector of size n. If x is a vector, $[x]_i$ denotes its i-th component and $[x]_{\mathcal{I}}$ the set of the components with indices in $\mathcal{I}$. If $\mathcal{S}$ is a set, $|\mathcal{S}|$ denotes its cardinal.

2 ASADPREC: a stochastic asynchronous first-order framework

The basis of our approach is to assume that the problem variables come in disjoint *blocks* of homogeneous nature. In deep learning applications, this occurs for instance when some variables are layer weights and other thresholds or biases of activation nodes. The first may be scalar and the second matrices [53, 16]. In line with [18], we assume that there are L blocks of variables, and problem (1.1) is therefore cast in the Cartesian product space

$$\mathbb{R}^N = \prod_{\ell=1}^{L} \mathbb{R}^{n_\ell \times m_\ell} = \prod_{\ell=1}^{L} \mathbb{R}^{d_\ell}.$$

We denote by $\mathcal{I}_\ell$ the set of indices of the variables in block ℓ. For $X \in \mathbb{R}^n$, $[X]_{\mathcal{I}_\ell}$ is then the subset of variables in X belonging to block ℓ. Following [18], we then specify, for each block, a particular norm $\|\cdot\|_\ell$ and its dual $\|\cdot\|_{\ell,*}$. The inner product on $\mathbb{R}^N$ is then defined as

$$\langle U, V\rangle = \sum_{\ell=1}^{L} \langle U_\ell, V_\ell\rangle_F \quad \text{where} \quad \langle A, B\rangle_F = \mathrm{tr}(A^T B)$$

and the norm on $\mathbb{R}^N$ given by

$$\|V\|^2 = \sum_{\ell=1}^{L} \|V_\ell\|_\ell^2 \tag{2.1}$$

whose dual norm is

$$\|U\|_*^2 = \sum_{\ell=1}^{L} \|U_\ell\|_{\ell,*}^2. \tag{2.2}$$

Because of the structure of the variables' space, it is natural to avoid mixing variables of different blocks, and this therefore defines the geometry of the asynchronous minimization method we are going to construct. Moreover, as is shown by the practical success of methods like Muon for matrix variables, different minimization methods may be appropriate for different types of variables. For each block, we therefore define a collection of methods applicable to variables in the block[1]. As was already the case in [18], each such *method* for block ℓ is specified by a preconditioner's update operator $\mathcal{U} : \mathbb{R}^{d_\ell} \to \mathbb{R}^{d_\ell}$ and a normalization operator $\mathcal{N} : \mathbb{R}^{d_\ell} \to \mathbb{R}^{d_\ell}$. The collection of methods applicable for block ℓ is then given by $\mathcal{M}_\ell = \{(\mathcal{U}_1, \mathcal{N}_1), \ldots, (\mathcal{U}_{a_\ell}, \mathcal{N}_{\hat{m}_\ell})\}$.

While this way to describe first-order adaptively preconditioned gradient methods may seem abstract, it was shown in [18] that it covers a fairly representative set of popular methods, such as AdaNorm, full and diagonal Adagrad, Shampoo and Muon[2] (see [18, Section 4.5] for details).

We now specify the ASADPREC[3] algorithmic *framework* on the next page, meaning that its description can be instantiated in a number of different specific algorithms. In what follows, we use the shorthand "ASADPREC algorithm" to refer to any algorithm belonging to the ASADPREC framework.

Some comments on this algorithm are useful at this stage.

1. We have stated the framework in a form as general as possible. It is however important to observe that *significant simplifications occur if*, as is often practical, *the computing architecture allocates to each processor a set of blocks disjoint from that of other processors.* In that setting, the preconditioner $\widehat{\Pi}_\ell$ may be stored locally on the processor in charge of block ℓ and Steps 6 and 9 are unnecessary. Moreover, this design also ensures that not two processors can work on the same block at the same time, so that Steps 2, 3 , 4 and 11 can be replaced by a simpler Step 3 where the block selection is restricted to those assigned to the current processor. Thus all steps in brackets can be ignored and Step 3 simplified, thereby significantly reducing interprocessor communication costs and resulting in a much leaner operation.
2. Pushing asynchronicity to the component- (rather than block-) level is difficult for Π_ℓ if one wishes to allow for non-diagonal preconditioners (as in full AdaGrad or Shampoo), because inconsistently read symmetric matrices may fail to be positive-definite.
3. No writing conflict can occur in Steps 9 and 10 because we have assumed a totally separable geometry, with no overlap between blocks.

[1] For instance, AdaNorm [50] and Adagrad [12, 37] are applicable methods for blocks with scalar variables. Shampoo [22] and Muon [26] are applicable for blocks with matrix variables.

[2] An adaptive version of Muon [54] for block ℓ is, for example, given by setting $\|\cdot\|_\ell$ to the spectral matrix norm, $\|\cdot\|_*$ to its dual, the nuclear norm, and defining $\mathcal{N}_\ell(G) = UV^T$ where the singular value decomposition of G is given by $U\Sigma V^T$, and $\mathcal{U}(G) = (\|G\|_*/\sqrt{d_\ell})I_{d_\ell}$.

[3] ASynchronous ADaptive PREConditioned gradient

Algorithm 2.1: ASADPREC

Given: a starting state $\widehat{X}_0$ and constants $\eta, \varsigma > 0$.
For $\ell \in \{1, \ldots, L\}$, set $\widehat{\Pi}_\ell = \varsigma I_\ell$ and $[\widehat{F}]_\ell = 1$.
For each processor $p \in \{1, \ldots, P\}$ in parallel, continuously

1. read $X \leftarrow \widehat{X}$ (block-atomic, inconsistent)
2. (read $F \leftarrow \widehat{\mathcal{F}}$) (component-atomic)
3. select a block ℓ such that $[F]_\ell = 1$ and an associated method $(\mathcal{U}_\ell, \mathcal{N}_\ell) \in \mathcal{M}_\ell$.
4. (update $[\widehat{F}]_\ell = 0$) (component-atomic)
5. draw ξ_ℓ and set $\widetilde{G}_\ell = [\nabla_X f(X, \xi_\ell)]_{\mathcal{I}_\ell}$
6. (read $\Pi_\ell^- \leftarrow \widehat{\Pi}_\ell$) (block-atomic)
7. $\Pi_\ell = \Pi_\ell^- + \mathcal{U}_\ell(\widetilde{G}_\ell)^2$,
8. $Z_\ell = \Pi_\ell^{-1/2} \widetilde{G}_\ell$,
9. (update $\widehat{\Pi}_\ell \leftarrow \Pi_\ell$) (block-atomic)
10. update $[\widehat{X}]_{\mathcal{I}_\ell} \leftarrow [X]_{\mathcal{I}_\ell} - \eta \, \|Z_\ell\|_{*,\ell} \mathcal{N}_\ell(Z_\ell)$ (block-atomic)
11. (update $[\widehat{F}]_\ell = 1$) (component-atomic)

4. The computation of the block's gradient $\widetilde{G}_{k(\ell_t)}$ in Step 4 may or may not require the computation of the complete gradient $\nabla f(X_{k(\ell_t)}, \xi_{k(\ell_t)})$, depending on the particular form of the objective function and the computer architecture.
5. The general framework of ASADPREC is reminiscent of the block-Jacobi [4, 15] setting for solving linear systems of equations.

Our analysis of ASADPREC's global rate of convergence is based on the view that, from an analytical standpoint, it can be viewed as a synchronous algorithmic framework with delays. To establish this interpretation, we define a *global iterate* $\widehat{X}_t$ $(t = 0, \ldots)$ as the state of the shared memory $\widehat{X}$ at each moment t where $\widehat{X}$ is modified by a block-atomic update. Because of the read operation of Step 1 is atomic, we have that $X = \widehat{X}_t$ and, after Step 5, $\widetilde{G}_\ell = [\nabla_X f(\widehat{X}_t, \xi_\ell)]_{\mathcal{I}_{\ell_t}}$. If ℓ_t is the index then selected at Step 3, the algorithm's loop on block ℓ_t continues, starting from $(\widehat{X}_t, [\nabla_X f(\widehat{X}_t, \xi_{k(\ell_t)})]_{\mathcal{I}_{\ell_t}})$. Now consider the interval t to $t+1$ and all steps being computed in parallel in this interval. Ignore all such steps whose computation does not terminate at $t+1$, and focus on the (unique, with probability one) block q_t which terminates at $t+1$. The computation of this step was started at some time $t - \tau_{t,q_t}$ (where the nonnegative integer τ_{t,q_t} is called a *delay*) using $\widehat{X}_{t-\tau_{t,q_t}}$ and the approximate block gradient $[\nabla_X f(\widehat{X}_{t-\tau_{t,q_t}}, \xi_{t-\tau_{t,q_t}})]_{\mathcal{I}_{q_t}}$. Moreover, because the selection process at Step 3 prevents two different processors to work on block q_t at the same time, we have that $[\widehat{X}_t]_{\mathcal{I}_{q_t}} = [\widehat{X}_{t-\tau_{t,q_t}}]_{\mathcal{I}_{q_t}}$. We may then (formally) consider that this step on block q_t was started at time t from $\widehat{X}_t$, but using the "delayed approximate block gradient" $[\nabla_X f(\widehat{X}_{t-\tau_{t,q_t}}, \xi_{k(q_{t-\tau_{t,q_t}})})]_{\mathcal{I}_{q_t}}$. Thus the sequence of overlapping step computations for different blocks is equivalent to a sequence where, in each interval from t to $t+1$, only the block steps terminating at the end of the interval are considered and are deemed to have started at t using a delayed approximate block gradient. This is exactly how the DADPREC algorithm described on the following page proceeds, at each iteration selecting a subset of blocks for which a complete step is computed using a approximate, *possibly delayed*, block gradient. We may thus see the sequence of iterates $\widehat{X}_t$ generated by an ASADPREC algorithm as being instead generated by a DADPREC algorithm, thereby identifying $\widehat{X}_t$ for ASADPREC with X_t for DADPREC. We also identify $\widetilde{G}_{\ell_t}$, Π_{ℓ_t} and Z_{ℓ_t} in the former with $\widetilde{G}_{t,\ell}$, $\Pi_{t,\ell}$ and $Z_{t,\ell}$ in the latter.

Note that Step 4 of DADPREC is purely formal, because that it does not involve any computation. Note also that $\widetilde{G}_{t,\ell}$ in DADPREC is a fairly general approximation of the true block gradient, as it may include delays and other errors, such as resulting from sampling.

Algorithm 2.2: DADPREC

Given: a starting point X_0 and constants $\eta, \varsigma > 0$. Set $\Pi_{-1,\ell} = \varsigma I_\ell$ for $\ell \in \{1, \ldots, L\}$.
For $t = 0, 1, \ldots$

1. Draw ξ_t and compute $\widetilde{G}_t$.
2. Select a nonempty "active" block set $\mathcal{A}_t \subseteq \{1, \ldots, L\}$.
3. For $\ell \in \mathcal{A}_t$, select $(\mathcal{U}_{t,\ell}, \mathcal{N}_{t,\ell}) \in \mathcal{M}_\ell$ and compute

$$\Pi_{t,\ell} = \Pi_{t-1,\ell} + \mathcal{U}_{t,\ell}(\widetilde{G}_{t,\ell})^2, \tag{2.3}$$

$$Z_{t,\ell} = \Pi_{t,\ell}^{-1/2} \widetilde{G}_{t,\ell}, \tag{2.4}$$

$$X_{t+1,\ell} = X_{t,\ell} - \eta \, \|Z_{t,\ell}\|_{*,\ell} \mathcal{N}_{t,\ell}(Z_{t,\ell}). \tag{2.5}$$

4. For $\ell \notin \mathcal{A}_t$, (formally) set

$$\Pi_{t,\ell} = \Pi_{t-1,\ell}, \tag{2.6}$$

$$Z_{t,\ell} = 0_{n_\ell \times m_\ell} \tag{2.7}$$

$$X_{t+1,\ell} = X_{t,\ell} \tag{2.8}$$

The reason for introducing the DADPREC framework is that its convergence analysis can be derived, with moderate effort, from that of the proposal in [18]. We describe the necessary assumptions and the results obtained (for DADPREC and thus for ASADPREC) in the next section, and postpone to the Appendix the detailed description of the modifications of the analysis of [18].

All assumptions and results will thus be formulated using the notation of the DADPREC framework, but may directly be interpreted in the context of ASADPREC.

As is clear from the algorithms' statements, this analysis does not include momentum. Variants including momentum will be considered in Section 4.

3 Convergence of the momentum-less DADPREC

Our analysis of the DADPREC framework starts by formulating standard conditions on the problem (1.1).

Assumption 1 (Boundedness) *There exists a constant $f_{\rm low}$ such that $F(X) \geq f_{\rm low}$ for all $X \in \mathbb{R}^N$.*

Defining $G(X) = \nabla_X F(X)$, we then require the following smoothness assumption.

Assumption 2 (Smoothness) *The objective function f is continuously differentiable and has a Lipschitz continuous gradient, that is there exists a constant $L_G \geq 0$ such that, for all $X, Y \in \mathbb{R}^N$, $\|G(X) - G(Y)\|_* \leq L_G \|X - Y\|$.*

We now introduce conditions that define the class of methods $(\mathcal{U}_\ell, \mathcal{N}_\ell)$ for which we develop our theory. These are defined for a given block $\ell \in \{1, \ldots, L\}$ and depends on the choice of dual norm $\|\cdot\|_{\ell,*}$ for this block. Again, we stress that *these abstract conditions do hold for several popular methods* for specific choices of the methods $(\mathcal{U}, \mathcal{N}) \in \mathcal{M}_\ell$, including AdaNorm, full and diagonal Adagrad, as well as adaptive variants of Shampoo and Muon [18, Section 4.5].

Assumption 3 (Structural identities) *For each $t \geq 0$ and $\ell \in \mathcal{A}_t$, we have that, for all $(\mathcal{U}, \mathcal{N}) \in \mathcal{M}_\ell$,*

$$\|Z_{t,\ell}\|_{*,\ell} \, \langle G_{t,\ell}, \mathcal{N}(Z_{t,\ell}) \rangle_F = \mathrm{tr}\big(\Pi_{t,\ell}^{-1/2} \mathcal{U}(\widetilde{G}_{t,\ell})^2\big), \tag{3.1}$$

and

$$\|Z_{t,\ell}\|_{*,\ell}^2 = \mathrm{tr}\big(\Pi_{t,\ell}^{-1} \mathcal{U}(\widetilde{G}_{t,\ell})^2\big). \tag{3.2}$$

Assumption 4 (Gradient-preconditioner compatibility) *There exists a constant $\kappa_\circ > 0$ such that, for all $\mathcal{U}$ in $(\mathcal{U},\mathcal{N}) \in \mathcal{M}_\ell$ and all G,*

$$\|G\|^2_{*,\ell} \leq \kappa_\circ \ \mathrm{tr}\big(\mathcal{U}(G)^2\big). \tag{3.3}$$

We also need to consider stochastic conditions on the quality of the gradient oracle.

Assumption 5 (Unbiased oracle) *The block gradient oracle is unbiased, that is*

$$\mathbb{E}_t\Big[\widetilde{G}_{t,\ell}\Big] = G_{t,\ell} \tag{3.4}$$

for all $t \geq 0$ and all $\ell \in \mathcal{A}_t$.

Assumption 6 (Cumulative variance) *There exists a constant $\omega \geq 0$ and a sequence $\{\nu_t\}_{k\geq 0}$, such that, for all $t \geq 0$,*

$$\sum_{j=0}^{t} \sum_{\ell\in\mathcal{A}_j} \mathbb{E}\Big[\|\widetilde{G}_{j,\ell} - G_{j,\ell}\|^2_*\Big] \leq \nu_t^2 + \omega^2 \sum_{j=0}^{t} \sum_{\ell\in\mathcal{A}_j} \mathbb{E}\big[\|Z_{j,\ell}\|^2_{*,\ell}\big]\,. \tag{3.5}$$

Observe that Assumption 5 is made at the block level, and is weaker than assuming the more standard condition that $\mathbb{E}_t\Big[\widetilde{G}_t\Big] = G_t = G(X_t)$. Similarly, Assumption 6 is also made at the block level. Moreover (3.5) requires a bound on the *cumulative* variance of all block gradient oracles over all past iterates, an approach also more general than assuming conditional variance at every iteration. In particular, it allows large variance at early iterations provided later iterations compensate. Trade-offs between different realizations and/or different blocKs are also theoretically possible.

We finally impose a minimal condition on the asynchronous nature of the algorithm.

Assumption 7 (Bounded delays) *There exists an integer $\tau \geq 0$ such that $\tau_{t,\ell} \leq \tau$ for all $t \geq 0$ and all $\ell \in \{1,\ldots,L\}$.*

For the purpose of analysis, we also define, for $t \geq 0$ and $\ell \in \{1,\ldots,L\}$,

$$\mathcal{J}_{t,\ell} = \{j \in \{0,\ldots,t\} \mid \ell \in \mathcal{A}_j\},$$

the set of indices of iterations at which block ℓ is effectively updated, and immediately note that the summations $\sum_{j=0}^{t}\sum_{\ell\in\mathcal{A}_j}$ and $\sum_{\ell=1}^{L}\sum_{j\in\mathcal{J}_{t,\ell}}$ are interchangeable (for instance in (3.5)).

A general theorem stating the global rate of convergence of any DADPREC (and thus ASADPREC) algorithm may now be stated under these assumptions.

Theorem 3.1 Suppose that the DADPREC algorithm is applied to problem (1.1) and that Assumptions 1 to 7 hold. Then, for all $t \geq 0$,

$$\min_{j\in\{0,\ldots,t\}} \mathbb{E}[\|G_j\|_*] \leq \frac{1}{t+1}\sum_{j=0}^{t}\mathbb{E}[\|G_j\|_*] \leq \frac{\kappa_a + \kappa_b\,\sqrt{N\log(\Theta_t)} + \kappa_c\,\Theta_t}{\sqrt{t+1}} \tag{3.6}$$

with

$$\kappa_a = L_G\eta\sqrt{\tau L\kappa_0}, \quad \kappa_b = L_G\eta\sqrt{2\tau L} \quad \text{and} \quad \kappa_c = \kappa_\circ\tau\sqrt{L},$$

$$\Theta_t \stackrel{\rm def}{=} \max\left[\kappa_\Theta, 12\sqrt{N}\,\nu_t\,\sqrt{\max\left[1,\log\left(12\sqrt{N}\,\nu_t\right)\right]}\right] \tag{3.7}$$

with

$$\kappa_\Theta = \max\left[e^{\max[1,\frac{\kappa_0}{2N}]},\ \mathbb{E}[f(X_0)] - f_{\rm low} + \eta\varsigma\,N, 24N\left(\omega + \frac{L_G}{\eta}\right)\log\left(24N\left(\omega+\frac{L_G}{\eta}\right)\right)\right]$$

and $\kappa_0 = -\sum_{\ell=1}^{L} d_\ell\log(d_\ell) - N\log(\varsigma)$.

Proof. See Appendix 1. □

Note that, since we prove (in Appendix) that each preconditioner $\Pi_{t,\ell}$, is bounded by the (in expectation) very slowly growing Θ_t (see (3.7)), and since Z_t is the preconditioned approximate gradient, condition (3.5) is akin to a (undelayed) cumulative affine variance condition of the form

$$\sum_{j=0}^{s}\sum_{\ell=1}^{L}\mathbb{E}_j\left[\|\widetilde{G}_{j,\ell}-G_{j,\ell}\|^2\right] \leq \nu_t^2+\omega^2\sum_{j=0}^{s}\sum_{\ell=1}^{L}\mathbb{E}_j\left[\|G_{j,\ell}\|_{*,\ell}^2\right],$$

whose iteration-wise variant has already been used in analysis of first-order methods (see [13, 49] or, for the stronger " affine-* " version, [1]). In particular, the "strong growth" assumption motivated by over-parametrized problems and used in [49] to derive an improved convergence rate for AdaGrad is essentially subsumed (at the cumulative level) for "preconditioned cumulative strong growth" (that is if $\nu_t = 0$ is assumed in (3.5)).
As stated, the rate of convergence of DADPREC is dominated by the term in $\kappa_c\Theta_t/\sqrt{t+1}$ in (3.6), where Θ_t depends on ν_t, the bound on the cumulative variance of the block gradient oracles, which may itself depend on t. The next corollary describes what can be said if one makes a more specific assumption on the oracle (block) variance at each iteration.

Corollary 3.2 Suppose that the DADPREC algorithm is applied to problem (1.1) Assumptions 1 to 5 and 7 hold. Suppose also that, for all $t \geq 0$ and $\ell \in \mathcal{A}_t$,

$$\mathbb{E}_t\left[\|\widetilde{G}_{t,\ell}-G_{t,\ell}\|_*^2\right] \leq \frac{\sigma_\ell^2}{|\mathcal{J}_{t,\ell}|^\alpha}+\omega^2\mathbb{E}_t\left[\|Z_{t,\ell}\|_*^2\right] \tag{3.8}$$

for some $\sigma_\ell \geq 0$, $\ell \in \mathcal{A}_t$ and $\alpha, \omega > 0$. Then, if $\psi_t = \max_{\ell\in\{1,\ldots,L\}}|\mathcal{J}_{t,\ell}|$,

$$\frac{1}{t+1}\sum_{j=0}^{t}\mathbb{E}[\|G_j\|_*] = \begin{cases} \mathcal{O}\left(\dfrac{\psi_t^{1-\alpha}\sqrt{\log(\psi_t)}}{\sqrt{t+1}}\right) & \text{if} \quad \alpha < 1, \\ \mathcal{O}\left(\dfrac{\sqrt{\log(\psi_t)\log(\log(t+1))}}{\sqrt{t+1}}\right) & \text{if} \quad \alpha = 1, \\ \mathcal{O}\left(\dfrac{1}{\sqrt{t+1}}\right) & \text{if} \quad \alpha > 1. \end{cases} \tag{3.9}$$

Proof. See Appendix 1. □

From the ASADPREC point of view, $|\mathcal{J}_{t,\ell}|$ corresponds to $k(\ell_t)$, and therefore counts the number of updates to block t, while ψ_t is their maximum taken on all blocks. We have that $\psi_t c \leq t+1$ in general, but if, at each iteration t, $|\mathcal{A}_t| \ll L$ (a not unusual situation), then $\psi_t \ll t+1$, suggesting an acceleration of the convergence rate compared with the undelayed algorithm (where $\psi_k = t+1$).

Observe that the continuity of the bound expressed in Theorem 3.1 as a function of ν_t is lost in the statement of Corollary 3.2, because the constants hidden in the $\mathcal{O}(\cdot)$ notation depend on α. In particular, this formulation does not support taking the limit for α tending to one. Observe also that we could weaken (3.8) by requiring that

$$\mathbb{E}_t\left[\|\widetilde{G}_{t,\ell}-G_{t,\ell}\|^2\right] \leq \frac{\sigma_\ell^2}{t^\alpha}+\omega^2\|Z_{t-1,\ell}\|_*^2 \tag{3.10}$$

for $\ell \in \mathcal{A}_t$ with the convention that $Z_{-1,\ell} = 0$, since, obviously,

$$\sum_{\ell=1}^{L}\sum_{j=0}^{t-1}\mathbb{E}_j\left[\|Z_{j,\ell}\|_*^2\right] \leq \sum_{\ell=1}^{L}\sum_{j=0}^{t}\mathbb{E}_j\left[\|Z_{j,\ell}\|_*^2\right].$$

The advantage of (3.10) over (3.8) is that the right-hand-side of this new condition is now measurable at iteration t.

As was noted in [18], although the rates of convergence obtained in Corollary 3.2 under the bias and variance Assumptions 5 and 6 are reasonable, they do not quite match, for high-variance regimes, the best obtained so far for (momentum-less, synchronous) AdaGrad and AdaNorm [49]. However they recover (in order) the best $\mathcal{O}(\sqrt{\log(k+1)\log(\log(k+1))/(k+1)})$ rate in the preconditioned cumulative strong growth context[4]. Importantly, they do so for a large class of asynchronous algorithms.

4 Convergence of DADPREC with momentum

Adding momentum to first-order methods has often been instrumental in improving numerical performance[5]. It is therefore of interest to consider asynchronous versions of such methods. We investigate two versions of this idea in this section, which share the common framework ASADPREC.M stated below for some given sequences $\{\mu_k\}$ of momentum parameters with $0 \leq \mu_k \leq \mu_{\max} < 1$ for all $k \geq 0$, with k indexing the number of updates to block ℓ.

Algorithm 4.1: ASADPREC.M

Given: a starting state $\widehat{X}_0$ and constants $\eta, \varsigma > 0$.
For $\ell \in \{1, \ldots, L\}$, set $\widehat{\Pi}_{-1,\ell} = \varsigma I_\ell$.
For each processor $p \in \{1, \ldots, P\}$ in parallel, continuously

1. read $X \leftarrow \widehat{X}_t$ (block-atomic, inconsistent)
2. (read $F \leftarrow \widehat{F}$) (component-atomic)
3. select a block ℓ such that $[F]_\ell = 1$ and an associated method $(\mathcal{U}_\ell, \mathcal{N}_\ell) \in \mathcal{M}_\ell$.
4. (update $[\widehat{F}]_\ell = 0$) (component-atomic)
5. (read $\Pi_\ell^- \leftarrow \widehat{\Pi}_\ell$) (block-atomic)
6. (read $M_\ell^- \leftarrow \widehat{M}_\ell$) (block-atomic)
7. draw ξ_ℓ and set $\widetilde{G}_\ell = [\nabla_X f(X, \xi_\ell)]_{\mathcal{I}_\ell}$
8. (read $k \leftarrow [\widehat{K}]_\ell$) (component-atomic)
9. compute M_ℓ and Π_ℓ, using $\mathcal{U}_\ell$ and k.
10. $Z_\ell = \Pi_\ell^{-1/2} M_\ell$,
11. (update $\widehat{\Pi}_\ell \leftarrow \Pi_\ell$) (block-atomic)
12. (update $\widehat{M}_\ell \leftarrow M_\ell$) (block-atomic)
13. (update $[\widehat{F}]_\ell = 1$) (component-atomic)
14. (update $[\widehat{K}]_\ell = k + 1$) (component-atomic)
15. update $[\widehat{X}]_{\mathcal{I}_\ell} \leftarrow [X]_{\mathcal{I}_\ell} - \eta \, \|Z_\ell\|_{*,\ell} \, \mathcal{N}_\ell(Z_\ell)$ (block-atomic)

The two variants are specified by detailing Step 9 either as

$$\left.\begin{array}{rcl} \Pi_\ell & = & \Pi_\ell^- + \mathcal{U}_\ell(\widetilde{G}_\ell)^2, \\ M_\ell & = & \left\{ \begin{array}{ll} \mu_k M_\ell^- + (1-\mu_k)\widetilde{G}_\ell & \text{if} \quad k > 0, \\ \widetilde{G}_\ell & \text{if} \quad k = 0, \end{array} \right. \end{array}\right\} \text{(ASADPREC.MG)}$$

[4] That is not only in the case where $\sigma_\ell = 0$ for each ℓ, but, more generally, in the case where the "inaccuracy budget" ν_t^2 is finite.

[5] Although, as far as the authors are aware, this has not so far been translated in improved convergence theory.

or as

$$\left.\begin{array}{rcl} M_\ell & = & \left\{ \begin{array}{ll} \mu_k M_\ell^- + (1-\mu_k)\widetilde{G}_\ell & \text{if} \quad k > 0, \\ \widetilde{G}_\ell & \text{if} \quad k = 0 \end{array} \right. \\ \Pi_\ell & = & \Pi_\ell^- + \mathcal{U}_\ell(M_\ell)^2. \end{array} \right\} \text{(ASADPREC.MM)}$$

Note that, in both cases, the momentum $\widehat{M_\ell}$ and the index $[\widehat{K}]_\ell$ must in general be read from and updated in shared memory, unless, again, specific blocks are affected to specific processors, in which case they can be maintained in local memory. *All bracketed steps may thus be omitted in this context.*

Fortunately, the convergence theory can be adapted to cover both variants. This is achieved again by considering DADPREC.MM and DADPREC.MG, the obvious extensions of DADPREC corresponding to ASADPREC.MM and ASADPREC.MG, respectively, where (2.3) and (2.4) are now replaced by

$$\begin{array}{rcl} M_{t,\ell} & = & \mu_{t,\ell} M_{\pi(t-1,\ell),\ell} + (1-\mu_{t,\ell})\widetilde{G}_{t,\ell}, \\ \Pi_{t,\ell} & = & \Pi_{t-1,\ell} + \mathcal{U}_{t,\ell}(M_{t,\ell})^2, \\ Z_{t,\ell} & = & \Pi_{t,\ell}^{-1/2} M_{t,\ell} \end{array}$$

for DADPREC.MM,

$$\begin{array}{rcl} \Pi_{t,\ell} & = & \Pi_{t-1,\ell} + \mathcal{U}_{t,\ell}(\widetilde{G}_{t,\ell})^2, \\ M_{t,\ell} & = & \mu_{t,\ell} M_{\pi(t-1,\ell),\ell} + (1-\mu_{t,\ell})\widetilde{G}_{t,\ell}, \\ Z_{t,\ell} & = & \Pi_{t,\ell}^{-1/2} M_{t,\ell} \end{array}$$

for DADPREC.MG.

Because the update formula for $Z_{t,\ell}$ above now involves the momentum $M_{t,\ell}$ instead of $\widetilde{G}_{t,\ell}$, we reformulate Assumption 3 to reflect this change[6] as follows.

Assumption 8 (Structural identities with momentum) *For each $t \geq 0$ and $\ell \in \mathcal{A}_t$, we have that, for all $(\mathcal{U}, \mathcal{N}) \in \mathcal{M}_\ell$,*

$$\|Z_{t,\ell}\|_{*,\ell} \, \langle M_{t,\ell}, \mathcal{N}(Z_{t,\ell}) \rangle_F = \mathrm{tr}\big(\Pi_{t,\ell}^{-1/2} \mathcal{U}(M_{t,\ell})^2\big), \tag{4.1}$$

and

$$\|Z_{k(\ell)}\|_{*,\ell}^2 = \mathrm{tr}\big(\Pi_{k(\ell)}^{-1} \mathcal{U}(M_{k(\ell)})^2\big). \tag{4.2}$$

Armed with this reformulated assumption, it is now possible to establish that the expected global rate of convergence of a DADPREC.MM algorithm is essentially identical to that of the version without momentum.

Theorem 4.1 Suppose that an DADPREC.MM algorithm is applied to problem (1.1) and that Assumptions 1, 2, and 4 to 8 hold. Then the conclusions of Theorem 3.1 and Corollary 3.2 do hold.

Proof. See Appendix 2. □

We note at this point that our momentum definition and convergence analysis for the DADPREC.MM algorithm do not make the assumption that the stepsize parameter η is sufficiently small, in contrast with previous proofs of convergence where results assume that η is small enough, in particular smaller than a multiple of the (usually unknown) Lipschitz constant L_G [21, 25, 52]. Establishing convergence results for the DADPREC.MG variant unfortunately requires additional assumptions. In particular, the stepsize η has to be chosen small enough. This alternative theory hinges on the fact that $\mathcal{U}(\widetilde{G}_{k,\ell})$ can be viewed as a perturbation of $\mathcal{U}(M_{k,\ell})$, and therefore that the structural relations of Assumption 8 are themselves perturbed when the DADPREC.MG algorithm is used. Fortunately, it remains possible to bound these perturbations by a mix of variance-related and second-order terms, the first of which potentially affecting the resulting global convergence rate. The final outcome is given by the following theorem and corollary.

[6]The reformulation obviously still holds for AdaNorm, AdaGrad, Shampoo and Muon [18].

Theorem 4.2 Suppose that a DADPREC.MG algorithm is applied to problem (1.1). Suppose that Assumptions 1, 2, 4, 5, 7 and 8 hold. Suppose also that there exists constants $\kappa_\square, \kappa_\diamond > 0$ such that

$$\mathcal{U}(U+V)^2 \preceq \kappa_\square \mathcal{U}(U)^2 + \kappa_\square \mathcal{U}(V)^2 \quad \text{and} \quad \mathrm{tr}\Big(\mathcal{U}(U)^2\Big) \leq \kappa_\diamond \|U\|_{*,\ell}^2 \tag{4.3}$$

for all $t \geq 0$ and $\ell \in \mathcal{A}_t$ and all $U, V \in \mathcal{M}_\ell$, and that

$$\text{either} \quad \eta \leq \frac{1-\mu_{\max}}{L_G}\sqrt{\frac{\varsigma}{6\kappa_\square\kappa_\diamond\tau L}} \quad \text{or} \quad \sum_{j=0}^{\infty} \|Z_j\|_*^2 \leq \kappa_Z \tag{4.4}$$

for some $\kappa_Z \geq 0$. Then

$$\min_{j\in\{0,\ldots,t\}} \mathbb{E}[\|G_j\|_*] \leq \frac{1}{t+1}\sum_{j=0}^{t} \mathbb{E}[\|G_j\|_*] \leq \frac{\kappa_a + \kappa_b\sqrt{N\log(\Theta_t)} + \kappa_c\,\Theta_t}{\sqrt{t+1}} \tag{4.5}$$

with

$$\kappa_a = L_G\eta\sqrt{\tau L\kappa_0}, \quad \kappa_b = L_G\eta\sqrt{2\tau L}, \quad \kappa_c = \kappa_\circ\tau\sqrt{L},$$

and

$$\Theta_t = \max\left[\kappa_\Theta, \frac{3\kappa_{\nu\nu}\theta_t^2}{\eta}, T_t\right], \tag{4.6}$$

where

$$\kappa_\Theta = \max\left[e^{\max\left[1,\frac{\kappa_0}{2N}\right]}, \frac{3\kappa_{\rm gap}}{\eta}, 24N\kappa_\Delta\left(\omega^2 + \frac{L_G}{\eta}\right)\log\left(24N\kappa_\Delta\left(\omega^2+\frac{L_G}{\eta}\right)\right)\right],$$

$$\theta_t^2 = \sum_{\ell=1}^{L}\sum_{j\in\mathcal{J}_{t,\ell}} \max[\mu_{\pi(j-1,\ell),\ell}, \mu_{j,\ell}]^2\,\mathbb{E}\Big[\|\widetilde{G}_{j,\ell} - G_{j,\ell}\|_{*,\ell}^2\Big], \tag{4.7}$$

$$T_t = 12\sqrt{N}\,\kappa_{\nu\Delta}\,\theta_t\sqrt{\max\Big[1, \log\Big(12\sqrt{N}\,\kappa_{\nu\Delta}\,\theta_t\Big)\Big]},$$

$\kappa_{\rm gap} = \mathbb{E}[f(X_0)] - f_{\rm low} + \eta\varsigma N$, $\kappa_\circ$ is defined in Assumption 4, κ_0 in Theorem 3.1, and $\kappa_{\rm gap}$, $\kappa_{\nu\nu}$, $\kappa_{\nu\Delta}$ and κ_Δ are specified in (A.37)–(A.38).

Proof. See Appendix 2. □

As it turns out, assuming (4.3) is not restrictive[7], but supposing (4.4) is definitely less desirable because the value of the Lipschitz constant L_G or that of κ_Z (if it exists) is usually unknown. It is however common practice in the literature [21, 25, 52]. Note the term $\kappa_{\nu\nu}\nu_t^2$ in the middle term of the expression of Θ_t in (4.6), which is the first significant difference between (4.6) and (3.7) and induces a modified convergence rate. The second difference is the presence of the factor $\max[\mu_{\pi(j-1,\ell),\ell}, \mu_{j,\ell}]^2$ in the definition of the cumulative variance, which has the opposite effect of potentially improving the convergence rate. This is expressed by the following corollary, where $|\mathcal{J}_{t,\ell}|$ is the number of updates to block ℓ up to iteration t.

[7] AdaNorm, AdaGrad, Shampoo and Muon continue to be covered, as verified in [18].

Corollary 4.3 Suppose that a DADPREC.MG algorithm is applied to problem (1.1). Suppose that Assumptions 1, 2, 5, 4, 7 and 8 hold. Suppose also that (4.3) and (4.4) hold, that

$$\mathbb{E}_t\left[\|\widetilde{G}_{t,\ell} - G_{t,\ell}\|_*^2\right] \leq \frac{\sigma_\ell^2}{|\mathcal{J}_{t,\ell}|^\alpha} + \omega^2 \mathbb{E}_t\left[\|Z_{t,\ell}\|_{*,\ell}^2\right] \tag{4.8}$$

for some $\alpha > 0$ and all $k \geq 0$ and $\ell \in \{1, \ldots, L\}$, and that, for each $\ell \in \mathcal{A}_t$,

$$\mu_{t,\ell} \in \left[0, \frac{\mu_{\max}}{|\mathcal{J}_{t,\ell}|^\beta}\right] \tag{4.9}$$

for some $\mu_{\max} < 1$ and some $\beta \geq 0$. Then, if $\psi_t = \max_{\ell \in \{1,\ldots,L\}} |\mathcal{J}_{t,\ell}|$,

$$\frac{1}{k+1}\sum_{j=0}^{t} \mathbb{E}[\|G_j\|_*] = \begin{cases} \mathcal{O}\left(\dfrac{\psi_t^{1-\alpha-2\beta}}{\sqrt{t+1}}\right) & \text{if} \quad \alpha + 2\beta < 1, \\ \mathcal{O}\left(\dfrac{\log(\psi_t)}{\sqrt{t+1}}\right) & \text{if} \quad \alpha + 2\beta = 1, \\ \mathcal{O}\left(\dfrac{1}{\sqrt{t+1}}\right) & \text{if} \quad \alpha + 2\beta > 1. \end{cases} \tag{4.10}$$

Proof. See Appendix 2. □

Again, be aware that the constants hidden in the $\mathcal{O}(\cdot)$ depend on α and β, in particular preventing taking limits for α tending to one. The rate of convergence now depends on the value of $\alpha + 2\beta$, indicating how *acting on the momentum parameter can partly alleviate the effect of high gradient oracle's variance*. Because the convergence rate is determined by θ_t, choosing a small momentum parameter in effect reduces the propagation of larger errors in the gradient oracle across iterations. In particular, setting $\mu_{t,\ell}$ to a multiple of $|\mathcal{J}_{t,\ell}|^{-\frac{1}{4}\alpha}$ recovers the rate of the momentum-less variant also for the high-variance regime ($\alpha < 1$). If however $\mu_{t,\ell}$ is kept constant (or bounded away from zero), that is if $\beta = 0$, then (4.8) is stronger than (3.8) and the rate of convergence for $\alpha < 1$ is now $\mathcal{O}\left((k+1)^{\frac{1}{2}-\alpha}\right)$ instead of $\mathcal{O}\left((k+1)^{-\frac{\alpha}{2}}\right)$, requiring in particular that $\alpha > \frac{1}{2}$.

5 Approximate normalization

We have assumed, in our theory, that normalization of the step is exact, in the sense that $\|\mathcal{N}_{t,\ell}(Z)\|_\ell = 1$ for all t, all $\ell \in \mathcal{A}_t$ and all $Z \in \mathbb{R}^{n_\ell \times m_\ell}$. This choice has been made for simplicity, but the reader can readily convince himself/herself that it is not necessary, and that it is sufficient to assume the existence of constants $0 < \kappa_{1,\mathcal{N}} \leq 1 \leq \kappa_{2,\mathcal{N}}$ such that $\|\mathcal{N}_{t,\ell}(Z)\|_\ell \in [\kappa_{1,\mathcal{N}}, \kappa_{2,\mathcal{N}}]$ for all t, $\ell \in \mathcal{A}_t$ and all $Z \in \mathbb{R}^{n_\ell \times m_\ell}$. The constants $\kappa_{1,\mathcal{N}}$ and $\kappa_{2,\mathcal{N}}$ then percolate though all proofs, complicating the expression of relevant factors even more, but without affecting the order of global convergence. This observation is particularly useful for methods involving matrix blocks, such as Shampoo or Muon, where step normalization is typically performed by a possibly inexact Newton-Schulz iteration [28, 5, 3, 48, 34].

6 Numerical experiments

6.1 Experimental objectives

The experiments in this section illustrate the practical behavior of the asynchronous block-parallel adaptive preconditioning method (ASADPREC) introduced in the previous sections. The convergence analysis in Section 3 establishes that the asynchronous block-coordinate iteration converges

under a bounded-delay condition, provided the block preconditioners capture sufficient local curvature to control the inter-block coupling. The experiments are designed to verify that these theoretical predictions hold on concrete machine learning problems, and to quantify the practical trade-offs involved.

The OFFO (Objective-Function-Free Optimization) framework of [18] provides the algorithmic foundation: the parameter vector is decomposed into L blocks, each equipped with its own adaptive preconditioner, and the method requires only gradient evaluations—no function value computations. In the synchronous regime, this reduces to a preconditioned block-Jacobi iteration; in the asynchronous regime, each block is updated independently using a possibly stale snapshot of the other blocks. Classical results on asynchronous chaotic relaxation [10] and asynchronous iterations [4, 15] predict convergence when the spectral radius of the "asynchronous iteration matrix" is less than one—a condition that is favored by weak inter-block coupling and strong local preconditioning. In practice, the throughput gain from asynchronous execution depends on the computational heterogeneity between blocks. When all blocks have equal per-iteration cost, asynchronous execution offers no advantage over the synchronized variant (the barrier imposes no idle time). When block costs differ, the synchronization barrier forces fast blocks to idle while waiting for the slowest block, and the throughput gain from removing the barrier is bounded by an Amdahl-type ratio: the cost of the slowest worker relative to the average. The experiments are designed to explore this relationship across a range of heterogeneity levels. More specifically, the experiments address the following questions:

1. Does asynchronous block-parallel optimization preserve convergence quality (training loss, gradient norm, generalization metric) relative to its synchronous counterpart, as predicted by the bounded-delay convergence theory?
2. What throughput gain does asynchronous execution provide, and how does this gain relate to the measured computational heterogeneity between blocks (Amdahl-type bound)?
3. How does exponential moving average (EMA) momentum interact with the stale reads inherent to asynchronous execution?

The benchmarks are chosen to cover a range of architectural patterns (dense MLPs, sparse embedding models, hybrid architectures) and heterogeneity levels, so that the interplay between block structure, computational cost imbalance, and asynchronous scheduling can be studied in a controlled setting. The goal is *not* to achieve state-of-the-art predictive performance, but to verify the theoretical predictions and to identify the regimes in which asynchronous block parallelism is most beneficial. We consider three algorithmic variants. All three apply the same per-block preconditioned updates and differ only in scheduling and synchronization.

Synchronous single-gradient (SyncSingle): A single mini-batch is sampled and the stochastic gradient $G_k = \nabla F_{\mathcal{B}_k}(x_k)$ is computed once. All L blocks are updated from the same snapshot, using Steps 8 and 10 of the ASADPREC framework. This variant minimizes gradient computation cost but assumes that a single gradient evaluation can be shared across all block solvers.

Synchronous multi-gradient (SyncMulti): Each of the L workers independently computes the full stochastic gradient on the same shared batch and snapshot, then extracts its own block component. Workers are synchronized by barriers: no block update is applied until every worker has completed its gradient computation. The mathematical iterate is identical to that of, SyncSingle, using Steps 8 and 10 of ASADPREC, but the wall-clock cost per iteration includes the overhead of L gradient evaluations and barrier synchronization.

Asynchronous block-owner (ASADPREC): The synchronization barriers are removed. Each block X_ℓ is permanently assigned to a dedicated worker. All workers share access to the full parameter vector $\widehat{X} = (\widehat{X}_1, \ldots, \widehat{X}_L)$. Worker ℓ repeatedly:

(i) reads the full vector $\widehat{X}$ by cloning each block $\widehat{X}_j$ under block j's lock, ensuring that all parameters within a given block come from the same publication epoch (intra-block consistency); different blocks may come from different epochs (inter-block asynchrony);

(ii) samples a local mini-batch $\mathcal{B}_{t,\ell}$, computes the full stochastic gradient $\nabla F_{\mathcal{B}_{t,\ell}}(\widehat{X}_k)$, and extracts the block component $G_{t,\ell} = [\nabla F_{\mathcal{B}_{t,\ell}}(\widehat{X}_k)]_\ell$;
(iii) applies the preconditioned update to $\widehat{X}_\ell$ and publishes the result atomically under block ℓ's lock.

This is the asynchronous block-Jacobi model [4, 15] used by ASADPREC. A bounded-delay throttle pauses any worker whose publication counter exceeds the slowest worker's by more than $\tau_{\max} = 512$.

6.2 Test problems and architectures

Table 2 summarizes the five benchmarks. Each model is decomposed so that one or two blocks use the matrix-valued adaptive Muon preconditioner (Newton–Schulz polar approximation [26] for SVHN, SVD for the others) while the remaining blocks use diagonal AdaGrad.

Dataset	Architecture	n_{train}	p	L	Muon block(s)
FashionMNIST	MLP $784 \to 4096 \to 10$	60,000	3,256,330	4	$W_1 \in \mathbb{R}^{4096\times 784}$
MoE-FMNIST	4 experts (see text)	60,000	3,887,950	6	$W_{1,1} \in \mathbb{R}^{512\times 2048}$, $W_{2,1} \in \mathbb{R}^{512\times 1024}$
CovType	MLP $54 \to 512 \to 256 \to 7$	160,000	161,287	4	$W_2 \in \mathbb{R}^{512\times 256}$
MovieLens 100K	NeuralMF, d=128, h=512	80,000	470,722	5	$W_1 \in \mathbb{R}^{512\times 256}$
Criteo	Hash-embed + MLP $429 \to 256 \to 1$	160,000	634,625	3	$W_1 \in \mathbb{R}^{429\times 256}$
SVHN	MLP $3072 \to 4096 \to 512 \to 10$	73,257	14,689,802	5	$W_1 \in \mathbb{R}^{4096\times 3072}$, $W_2 \in \mathbb{R}^{512\times 4096}$

Table 2: Test problems and block decompositions.

FashionMNIST [51]: A one-hidden-layer MLP with 3.26×10^6 parameters decomposed into 4 blocks: W_1 (Muon), b_1, W_2, b_2 (AdaGrad).

MoE-FMNIST [51]: A Mixture-of-Experts architecture applied to flattened $28\times 2828\times 28$ Fashion-MNIST images, with a backbone MLP ($784 \to 512784 \to 512$), a top-2/4 softmax router, four experts of varying width ($512 \to h \to 512$) with $h \in \{2048, 1024, 256, 64\}$), and an output head ($512 \to 10512 \to 10$), decomposed into 6 blocks: $W_{1,1}$ and $W_{2,1}$ (Muon, Newton–Schulz, 5 iterations), expert 3 and expert 4 full parameters, backbone and expert 1 and 2 biases and second-layer weights, router and output head (all AdaGrad).

CovType [6]: A two-hidden-layer MLP with 1.61×10^5 parameters (200,000 samples, 80/20 split), decomposed into 4 blocks: (W_1, b_1), W_2 (Muon), b_2, (W_3, b_3) (AdaGrad).

MovieLens 100K [24]: A NeuralMF model with 4.71×10^5 parameters (100,000 ratings, 943 users, 1,682 items), decomposed into 5 blocks: P, Q, (b_u, b_i) (sparse row-wise AdaGrad), W_1 (Muon), (c_1, W_2, c_2) (AdaGrad).

Criteo [31]: A click-through-rate model with 6.35×10^5 parameters, decomposed into 3 blocks: embedding table (AdaGrad), W_1 (Muon), (b_1, W_2, b_2) (AdaGrad).

SVHN [40]: A two-hidden-layer MLP with 14.7×10^6 parameters applied to flattened $32\times 32\times 3$ images, decomposed into 5 blocks: W_1 and W_2 (Muon, Newton–Schulz, 5 iterations), b_1, b_2, (W_3, b_3) (AdaGrad). This benchmark features two Muon blocks of different sizes (12.6×10^6 and 2.1×10^6 entries), creating a hierarchy of block costs.

6.2.1 Measured block heterogeneity

The raw block cost ratio ($t_{\max}/t_{\min}$) exceeds 90 on all benchmarks, but the effective heterogeneity is considerably lower. Each worker's per-iteration cost has two components: the gradient computation t_{grad} (identical for all workers, since each computes the full gradient ∇F) and the block update $t_{\text{block},\ell}$. When t_{grad} is large relative to $t_{\text{block},\ell}$, all workers run at similar speeds regardless of block update costs. Table 3 reports per-block computational costs measured on an Apple M3 Pro processor (single-threaded, median over 50 repetitions). The *effective heterogeneity ratio* is defined as $(t_{\text{grad}} + t_{\max})/(t_{\text{grad}} + t_{\min})$.

Dataset	Gradient	Slowest block	Fastest block	$t_{\max}/t_{\min}$	Eff. ratio
FashionMNIST	5.6	105.2 (Muon)	0.004	25,000	19.6
MoE-FMNIST	37.6	451.0 (Muon)	0.035	12,733	5.7
CovType	0.8	0.42 (Muon)	0.004	96	1.5
MovieLens	1.0	5.61 (Muon)	0.011	526	6.5
Criteo	1.8	4.83 (Muon)	0.012	419	3.7
SVHN	23.4	580.2 (Muon)	0.004	134,000	25.8

Table 3: Per-block computational cost (ms) and effective heterogeneity.

On FashionMNIST, the Muon polar factorization (105 ms) far exceeds the gradient (5.6 ms), so the effective ratio remains high (19.6). On CovType, the gradient (0.8 ms) exceeds the slowest block update (0.42 ms), reducing the effective ratio to 1.5×. SVHN exhibits the highest effective ratio (25.8) thanks to the Newton–Schulz iteration on $W_1 \in \mathbb{R}^{4096\times 3072}$ (580 ms) relative to the gradient (23 ms).

6.3 Implementation and experimental protocol

All experiments use PyTorch on a shared-memory CPU architecture (Apple M3 Pro, 12 cores) with `torch.set_num_threads(1)`. The CPython GIL prevents byte-code level parallelism, but PyTorch C++ operations (matrix multiplications, SVD, tensor cloning) release the GIL, allowing concurrent execution across cores. The ASADPREC mode uses per-block locking: each block is read and written under its own lock, ensuring intra-block consistency while allowing inter-block asynchrony. A fixed wall-clock budget of 30 seconds is used for problems CovType, MovieLens and Criteo, a budget of 60 seconds for Fashion-MNIST, and MoE-FMNIST and a budget of 120 seconds for SVHN. All runs use 8 seeds $(1234, \ldots, 1241)$, and identical initial parameters per seed. Throughput is reported in model-equivalent updates per second: throughput $= N_{\text{block}}/(L \cdot T)$.

6.4 Results without momentum

Tables 4 and 5 present the results obtained with the momentum-less ASADPREC in the setting just described. We start by reporting, for all benchmarks the full-dataset training loss and the Euclidean norm of the full training gradient $\|\nabla F(X)\|$ as convergence diagnostics.

	Final training loss gradient $\|\nabla F(X)\|$			Final training loss value $F(X)$		
Dataset	SyncSingle	SyncMulti	ASADPREC	SyncSingle	SyncMulti	ASADPREC
FashionMNIST	0.421	0.343	**0.277**	0.321	0.321	**0.308**
MoE-FMNIST	**0.632**	0.826	0.744	0.309	0.314	**0.279**
CovType	0.568	**0.482**	0.485	0.243	0.260	**0.240**
MovieLens	0.024	0.027	**0.021**	**0.294**	0.340	0.353
Criteo	0.044	0.053	**0.033**	**0.076**	0.112	0.088
SVHN	1.004	**0.979**	1.180	0.840	0.833	**0.819**

Table 4: Convergence diagnostics for momentum-less variants

ASADPREC obtains the best gradient (in norm) for Criteo, FashionMNIST and MovieLens, while there is little difference between SyncSingle and ASADPREC for CovType. The gradient norm

is higher for SVHN and MoE-FMNIST (due to delayed gradients). The best loss value is obtained by ASADPREC for 4/6 benchmarks. SyncSingle remains better for MovieLens and Criteo because it performs only one gradient evaluation per iteration.

We now turn to primary generalization metrics, depending on the benchmark. For classification problems (FashionMNIST, CovType, SVHN), we report the *test accuracy* (fraction of correctly classified test samples). For MovieLens, we report the *root mean square error* (RMSE), defined as RMSE $= \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{r}_i - r_i)^2}$, where $\hat{r}_i$ and r_i are the predicted and true ratings. For Criteo, we report the *area under the ROC curve* (AUC), which measures the probability that a randomly chosen positive example (clicked ad) is ranked higher than a randomly chosen negative example by the model's predicted score; AUC $= 0.5$ corresponds to random ranking and AUC $= 1$ to perfect ranking. Generalization metrics are reported in Table 5, along with the throughput of ASADPREC and the speedup of ASADPREC compared with SyncMulti.

Dataset	metric	SyncSingle	SyncMulti	ASADPREC	Throughput	Speedup
FashionMNIST	acc.(%)	87.00	86.94	**87.34**	10	1.43
MoE-FMNIST	acc.(%)	86.82	86.52	**87.39**	42	1.27
CovType	acc.(%)	89.49	88.82	**89.56**	105	1.27
MovieLens	RMSE	0.930	0.919	**0.916**	89	1.14
Criteo	AUC	0.715	**0.722**	0.717	89	1.17
SVHN	acc.(%)	69.44	69.28	**69.65**	1	1.33

Table 5: Generalization metrics, throughput and speed-up for momentum-less variants

ASADPREC is best on all benchmarks except Criteo.

Globally, Tables 4 and 5 suggest that stale (delayed) gradients do not degrade the quality of the training and that the additional iterations made possible by asynchrony translate into better generalization.

6.5 Results with momentum

The approaches yielding DADPREC.MM/ASADPREC.MM and DADPREC.MG/ASADPREC.MG not only differ in theory, but their practical requirements are also significantly different, in particular for the Shampooand Muon case. Indeed, if the SVD of M_ℓ is given by $M_\ell = U_\ell \Sigma_\ell V_\ell^T$, the Muon recursions[8] are written, in this case, as

$$\begin{aligned} M_\ell &= \mu_k M_\ell^- + (1-\mu_k)\widetilde{G}_\ell \\ \Pi_\ell &= \Pi_\ell^- + \frac{\|M_\ell\|_{*,\ell}^2}{d_\ell}, \\ X_\ell^+ &= X_\ell - \eta \frac{\|M_\ell\|_{*,\ell}}{\sqrt{\Pi_\ell}} U_\ell V_\ell^T \end{aligned}$$

for Muon.MM,

$$\begin{aligned} \Pi_\ell &= \Pi_\ell^- + \frac{\|\widetilde{G}_\ell\|_{*,\ell}^2}{d_\ell}, \\ M_\ell &= \mu_k M_\ell^- + (1-\mu_k)\widetilde{G}_\ell \\ X_\ell^+ &= X_\ell - \eta \frac{\|M_\ell\|_{*,\ell}}{\sqrt{\Pi_\ell}} U_\ell V_\ell^T \end{aligned}$$

for Muon.MG.

The main difference is that Muon.MM only requires $\|M_\ell\|_{*,\ell}$, which can be computed with a single polar decomposition (using SVD or Newton-Schulz iterations), while Muon.MG also requires $\|\widetilde{G}_\ell\|_{*,\ell}$. This potentially requires another decomposition, which can significantly slow down the algorithm. It is therefore of interest to consider faster techniques for computing $\|\widetilde{G}_\ell\|_{*,\ell}$. We have considered both the stochastic Lanczos quadrature [47] and a warm-started power iteration for this task. As it turns out, the first, although theoretically suitable, appears to be too slow for practical use. By contrast, the second, which can only provide a deterministic lower bound, nevertheless works remarkably well. The details of this technique are given in Appendix 3.

When applied to ASADPREC, these considerations lead us to a family of momentum variants, of which we have retained six, whose characteristics are given in Table 6. In this table, β is the rate

[8] Π_ℓ is a scalar in this case.

of momentum decay appearing in (4.9) (with $\mu_{\max} = 0.9$) and the last column indicates whether a warm-started power iteration is used to approximate $\|\widetilde{G}_\ell\|_{*,\ell}$ in the Muon updates.

Variant	type	β	approx. $\|\widetilde{G}_\ell\|_{*,\ell}$
MMconst	MM	0	no
MGconst	MG	0	no
MGconstA	MG	0	yes
MMdecay	MM	0.5	no
MGdecay	MG	0.5	no
MGdecayA	MG	0.5	yes

Table 6: Asynchronous momentum variants and their characteristics

To keep the paper as concise as possible, we only report in Table 7 the values of the final gradients (in norm) for these six variants, and refer the reader to Appendix 3 for more results.

Dataset	No Mom.	MMconst	MGconst	MGconstA	MMdecay	MGdecay	MGdecayA
Fashion-MNIST	0.277	0.551	0.500	0.526	0.255	**0.246**	0.312
MoE-FMNIST	0.744	0.858	**0.656**	1.212	0.663	0.884	0.881
CovType	0.485	**0.246**	0.279	0.265	0.345	0.386	0.755
MovieLens	0.021	0.021	0.036	0.032	**0.015**	0.019	0.025
Criteo	0.033	0.036	**0.008**	0.018	0.046	0.035	0.028
SVHN	1.180	1.430	0.988	1.138	1.103	**0.880**	1.131

Table 7: Final gradient norm for momentum variants

As can be seen from this table, no specific momentum variant appears to dominate, but the momentum-less variant (first column) is never the best.

6.6 Discussion

Globally, our results suggest the following conclusions.

1. *Asynchronous parallelism preserves and improves convergence quality.* DADPREC is the best momentum-less variant for 5/6 benchmarks, and is also best in many case where momentum is used.
2. *Heterogeneity drives the speedup.* The per-block locking protocol—guaranteeing intra-block consistency while allowing inter-block asynchrony—matches the classical asynchronous block-Jacobi model [4, 15]. The effective heterogeneity ratio, which accounts for the shared gradient computation cost, is a better predictor of the asynchronous speedup than the raw block cost ratio.
3. The *cost of momentum is critical for matrix updates* like Muon. This makes the MM approach attractive from this point of view, because its lower cost per iteration allows more iterations in the time budget, resulting in gains in accuracy between +1.8 and +3.6 points compared with the MG approach.
4. The estimation of the nuclear norm of $\widetilde{G}_\ell$ using the *warm-started power iteration is an efficient proxy*, and does produce the best results for some problems.
5. As expected, the effect of decaying momentum (i.e. $\beta > 0$ in (4.9)) is problem-dependent, because of its direct interaction with the variance of the problem's oracle.
6. Although globally promising, our tests did not allow us to select a clear winner among the studied variants.

It should however be stressed that all our experiments were conducted assuming the "full gradient evaluation model", that is the case where $G_{t,\ell} = [\nabla F_{\mathcal{B}_{t,\ell}}(\widehat{X}_k)]_\ell$ cannot be computed on its own, but requires the evaluation of $\nabla F_{\mathcal{B}_{t,\ell}}(\widehat{X}_k)$. Should the evaluation of $G_{t,\ell}$ be possible at a lower

cost[9], the heterogeneity ratios would then be dramatically increased, which is likely to give the asynchronous option used in ASADPREC a substantial advantage.

7 Conclusions and perspectives

We have presented a new class of asynchronous adaptive first-order methods including asynchronous variants of AdaNorm, AdaGrad, Shampoo and Muon, as well as variants thereof using momentum and/or inexact step normalization. We have shown that, under reasonable assumptions, all methods in the class have an global convergence rate (essentially) of order $\mathcal{O}(1/\sqrt{t})$ in expectation.

We also described numerical experiments with these methods, providing motivation for asynchronous adaptive optimization where different blocks use different geometries, different blocks have drastically different computational costs, and synchronization barriers become prohibitively expensive. Such situations naturally appear in modern large-scale machine learning systems mixing diagonal adaptive methods, Our results, although limited, provide empirical support for the claim that asynchronous adaptive block-coordinate optimization methods may be relevant in heterogeneous large-scale machine learning systems.

References

[1] A. Attia and T. Koren. SGD with AdaGrad stepsizes: Full adaptivity with high probability to unknown parameters, unbounded gradients and affine variance. arXiv:2302.08783, 2023.

[2] J. Bernstein and L. Newhouse. Modular duality in deep learning. arXiv:2410.21265v2, 2024.

[3] J. Bernstein and L. Newhouse. Old optimizer, new norm: an anthology. arXiv:2409.20325v2, 2024.

[4] D.P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice Hall, 1989.

[5] Å. Björck and C. Bowie. An iterative algorithm for computing the best estimate of an orthogonal matrix. *SIAM Journal on Numerical Analysis*, 8(2):358–364, 1971.

[6] J. A. Blackard and D. J. Dean. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Comput. Electron. Agric.*, 24(3):131–151, 1999.

[7] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, England, 2004.

[8] L. Cannelli, F. Facchinei, V. Kungurstev, and G. Scutari. Asynchronous parallel algorithms for nonconvex optimization. *Mathematical Programming, Series A*, 184:121–154, 2020.

[9] S. C. Chaturapruek, J. C. Duchi, and Ch. Ré. Asynchronous stochastic convex optimization. In *NIPS'15: Proceedings of the 29th International Conference on Neural Information Processing Systems*, volume 1, pages 1531–1539, 2015.

[10] D. Chazan and W. Miranker. Chaotic relaxation. *Linear Algebra and its Applications*, 2(2):199–222, 1969.

[11] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. LANCELOT*: a Fortran package for large-scale nonlinear optimization (Release A)*. Number 17 in Springer Series in Computational Mathematics. Springer Verlag, Heidelberg, Berlin, New York, 1992.

[12] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12, July 2011.

[13] M. Faw, I. Tziotis, C. Caramanis, A. Mokhtari, S. Shakkottai, and R. Ward. The power of adaptivity in SGD: Self-tuning step sizes with unbounded gradients and affine variance. In *Proceedings of 35th Conference on Learning Theory*, volume 178, pages 313–355, 2022.

[14] H. R. Feyzmahdavian and M. Johansson. Asynchronous iterations in optimization: New sequence results and sharper algorithmic guarantees. *Journal of Machine Learning Research*, 24:1–75, 2023.

[15] A. Frommer and D. B. Szyld. On asynchronous iterations. *Journal of Computational and Applied Mathematics*, 123(1–2):201–216, 2000.

[16] B. Ginsburg, P. Castonguay, O. Hrinchuk, O. Kuchaiev, R. Leary, V. Lavrukhin, J. Li, H. Nguyen, Y. Zhang, and J. M. Cohen. Training deep networks with stochastic gradient normalized by layerwise adaptive second moments. arXiv:1905.11286v3, 2019.

[9] Among other possibilities, we think here of structured networks such as mix of experts [23], DeepONets [35] or multi-frequency networks [17]. The approach of delayed gradients suggested by [55], in some sense close to DADPREC, may also be worth investigating.

[17] S. Gratton, V. Mercier, E. Riccietti, and Ph. L. Toint. A block-coordinate approach of multi-level optimization with an application to physics-informed neural networks. *Computational Optimization and Applications*, 89(2):385–417, 2024.

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

[19] A. Griewank and Ph. L. Toint. Local convergence analysis for partitioned quasi-Newton updates. *Numerische Mathematik*, 39:429–448, 1982.

[20] A. Griewank and Ph. L. Toint. On the existence of convex decomposition of partially separable functions. *Mathematical Programming*, 28:25–49, 1984.

[21] Z. Guo, W. Yin, R. Jin, and T. Yang. A novel convergence analysis for algorithms of the Adam family. In *Proceedings of OPT2021: 13th Annual Workshop on Optimization for Machine Learning*, 2021.

[22] V. Gupta, T. Koren, and Y. Singer. Shampoo: Preconditioned stochastic tensor optimization. In *Proceedings of the International Conference on Machine Learning*, pages 1842–1850, 2018.

[23] J. B. Hampshire and A. Waibel. The Meta-Pi network: building distributed knowledge representations for robust multisource pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(7):751—769, 1992.

[24] F. M. Harper and J. A. Konstan. The MovieLens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4), 2025.

[25] Y. Hong and J. Lin. Revisting convergence of Adagrad with relaxed assumptions. arXiv:2403.13794v2, 2024.

[26] K. Jordan, Y. Jin, V. Boza, Y. Jiacheng, F. Cecista, L. Newhouse, and J. Bernstein. URL:https://kellerjordan.github.io.posts/muon, 2024.

[27] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings in the International Conference on Learning Representations (ICLR)*, 2015.

[28] Z. Kovarik. Some iterative methods for improving orthonormality. *SIAM Journal on Numerical Analysis*, 7(3):386–389, 1970.

[29] V. Kungurtsev, M. Egan, B. Chatterjee, and D Alistarh. Asynchronous optimization methods for efficient training of deep neural networks with guarantees. arXiv;1905.11845v2, 2019.

[30] V. Kungurtsev, M. Egan, B. Chatterjee, and D Alistarh. Asynchronous optimization methods for efficient training of deep neural networks with guarantees. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8209–8216, 2021.

[31] Criteo Labs. Display advertising challenge. Kaggle, 2014.

[32] R. Leblond, F. Pedregosa, and S. Lacoste-Julien. Improved asynchronous parallel optimization analysis for stochastic incremental methods. *Journal of Machine Learning Research*, 19(81):1–68, 2018.

[33] X. Lian, Y. Huang, Y. Li, and J. Liu. Asynchronous parallel stochastic gradient for nonconvex optimization. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[34] J. Liu, J. Lin, X.Yao, Z. Jiang, G. Lai, Y. Du, Y. Qin, W. Xu, E. Lu, J. Yan, Y. Chen and. H. Zheng, Y. Liu, S. Liu, B. Yin, W. He, H. Zhu, Y. Wang, J. Wang, M. Dong, Z. Zhang, Y. Kang, H. Zhang, X. Xu, Y. Zhang, Y. Wu, W. Zhou, and Z. Yang. MUON is scalable for LLM training. arXiv:2502.16982, 2025.

[35] L. Lu, P. Jin, and G. Karniadakis. DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators.

[36] H. Mania, X. Pan, D. Papailiopoulos, B. Recht, K. Ramchandran, and M. I. Jordan. Perturbed iterate analysis for asynchronous stochastic optimization. *SIAM Journal on Optimization*, 27(4):2202—2229, 2017.

[37] B. McMahan and M. Streeter. Adaptive bound optimization for online convex optimization. In *Conference on Learning Theory*, page 244sq, 2010.

[38] A. Nabli and E. Oyallon. DADA0: Decoupled accelerated decentralized asynchronous optimization. arXiv:2208.00779v3, 2022.

[39] G. Nadiradze, I. Markov, B. Chatterjee, and V. Kungurstev. Elastic consistency: a practical consistency model for distributed stochastic gradient descent. *ACM SIGACT News*, 53(2):64–82, 2022.

[40] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A.Y. Ng. Reading digits in natural images with unsupervised feature learning, 2011.

[41] L. M. Nguyen, P. H. Nguyen, P. Richtarik, K. Scheinberg, M. Takáč, and M. van Dijk. New convergence aspects of stochastic gradient algorithms. *Journal of Machine Learning Research*, 20(176):1–49, 2019.

[42] L. M. Nguyen, P. H. Nguyen, M. van Dijk, P. Richtarik, K. Scheinberg, and M. Takáč. SGD and Hogwild! convergence without the bounded gradients assumption. *Proceedings of Machine Learning Research*, 80:3750–3758, 2018.

[43] F. Niu, B. Recht, Ch. Ré, and S. J. Wright. HOGWILD!: A lock-free approach to parallelizing stochastic gradient descent. *Advances in Neural Information Processing Systems*, 4(1):693—-701, 2011.

[44] Z. Peng, Y. Xu, M. Yan, and W. Yin. ARock: An algorithmic framework for asynchronous parallel coordinate updates. *SIAM Journal on Scientific Computing*, 38(5):2851–2879, 2016.

[45] Th. Pethick, W. Xie, K. Antonakopoulos, Z. Zhu, A. Silveti-Falls, and V. Cevher. Training deep learning models with norm-constrained LMOs. arXiv:2502.07529v2, 2025.

[46] C. Si, D. Zhang, and W. Shen. AdaMuon: Adaptive Muon optimizer. arXiv2507.11005, 2025.

[47] S. Ubaru, J. Chen, and Y. Saad. Fast estimation of $tr(f(a))$ via stochastic Lanczos quadrature. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1075–1099, 2017.

[48] N. Vyas, D. Morwani, R. Zhao, M. Kwun, I. Shapira, B. Brandfonbrener, L. Janson, and S. Kakade. SOAP: Improving and stabilizing Shampoo using Adam. arXiv:2409.11321, 2024.

[49] B. Wang, H. Zhang, Z. Ma, and W. Chen. Convergence of Adagrad for non-convex objectives: simple proofs and relaxed assumptions. In *Proceedings of the Thirty-Sixth Conference on Learning Theory*, pages 161–190, 2023.

[50] R. Ward, X. Wu, and L. Bottou. Adagrad stepsizes: sharp convergence over nonconvex landscapes. In *Proceedings in the International Conference on Machine Learning (ICML2019)*, 2019.

[51] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST, a novel image dataset for benchmarking machine learning algorithms. arXiv:1708.07747, 2017.

[52] N. Xiao, X. Hu, X. Lin, and K.-C. Toh. Adam family methods for nonsmooth optimization with convergence guarantees. *Journal of Machine Learning Research*, 25, 2024.

[53] A. W. Yu, L. Huang, Q. Lin, R. Salakhutdinov, and J. Carbonell. Block-normalized gradient method; an empirical study for training deep neural network. arXiv:1707.04822v2, 2017.

[54] M. Zhang, Y. Liu, and H. Schaeffer. AdaGrad meets Muon: Adaptive stepsizes for orthogonal updates. arXiv:2509.02981v2, 2025.

[55] H. Zhuang, Y. Wang, Q. Liu, S. Zhang, and Z. Lin. Fully decoupled neural network learning using delayed gradients. arXiv:1906.09108v3, 2019.

Appendix 1: Detailed analysis for momentum-less DADPREC

The proof of Theorem 3.1 follows the broad lines of [18]. In order to keep our argument as compact as possible, we refer, in what follows, to that reference for the proofs of results directly extracted from it[10].

Our first step is to analyze the (expected) descent at iteration t.

Lemma A.1 Suppose that Assumption 2 holds. Then

$$\mathbb{E}_k[f(X_{t+1})] \leq f(X_t) - \eta \sum_{\ell \in \mathcal{A}_t} \mathbb{E}_t\left[\|Z_{t,\ell}\|_{*,\ell} \left\langle \widetilde{G}_{t,\ell}, \mathcal{N}_{t,\ell}(Z_{t,\ell}) \right\rangle_F\right] + \eta \sum_{\ell \in \mathcal{A}_t} \mathbb{E}_t\left[\|Z_{t,\ell}\|_{*,\ell} \left| \left\langle \widetilde{G}_{t,\ell} - G_{t,\ell}, \mathcal{N}_{t,\ell}(Z_{t,\ell}) \right\rangle_F \right|\right] + \frac{L_G \eta^2}{2} \sum_{\ell \in \mathcal{A}_t} \mathbb{E}_t\left[\|Z_{t,\ell}\|_{*,\ell}^2\right]. \tag{A.1}$$

Proof. See [18, Lemma 3.1]. □

The following trace inequalities may then be proved.

[10] Notation has changed: in [18], the iteration index was k, the normalization operator $S_\ell(\cdot)$ and the preconditioner update operator $\mathcal{L}_{k,\ell}(\cdot)$.

Lemma A.2 Suppose that a DADPREC.MG algorithm is applied to problem (1.1). Consider the DADPREC algorithm. We have that, for all $k \geq 0$,

$$\sum_{\ell=1}^{L} \mathrm{tr}(\Pi_{t,\ell}^{1/2}) - \sum_{\ell=1}^{L} \mathrm{tr}(\Pi_{-1,\ell}^{1/2}) \leq \sum_{\ell=1}^{L} \sum_{j \in \mathcal{J}_{k,\ell}} \mathrm{tr}\Big(\Pi_{j,\ell}^{-1/2} \mathcal{U}_{j,\ell}(\widetilde{G}_{j,\ell})^2\Big) \tag{A.2}$$

and

$$\sum_{\ell=1}^{L} \sum_{j \in \mathcal{J}_{t,\ell}} \mathrm{tr}\Big(\Pi_{j,\ell}^{-1} \mathcal{U}_{j,\ell}(\widetilde{G}_{j,\ell})^2\Big) \leq \sum_{\ell=1}^{L} \mathrm{tr}\Big(\log(\Pi_{t,\ell})\Big) - \sum_{\ell=1}^{L} \mathrm{tr}\Big(\log(\Pi_{-1,\ell})\Big). \tag{A.3}$$

Proof. See [18, Lemmas 3.2, 3.3 and 3.4]. In the last of these lemmas, the double sums $\sum_{j=0}^{k} \sum_{\ell=1}^{L}$ are replaced by $\sum_{\ell=1}^{L} \sum_{j \in \mathcal{J}_{t,\ell}}$. □

Because not every block is necessarily updated at iteration t in the DADPREC algorithm, we now need to re-prove the suitable variant of the classical telescoping argument.

Theorem A.3 Suppose that Assumption 1, 2, 3 and 6 hold. Define

$$\Delta_t \stackrel{\mathrm{def}}{=} \mathbb{E}\left[\sum_{\ell=1}^{L} \mathrm{tr}(\log \Pi_{k,\ell}) - \sum_{\ell=1}^{L} \mathrm{tr}(\log \Pi_{-1,\ell})\right]. \tag{A.4}$$

Then, for every $k \geq 0$,

$$\eta\, \mathbb{E}\left[\sum_{\ell=1}^{L} \mathrm{tr}(\Pi_{t,\ell}^{1/2})\right] \leq \kappa_{\mathrm{gap}} + \eta\, \nu_t \sqrt{\Delta_t} + \left(\eta\omega + \frac{L_G \eta^2}{2}\right) \Delta_t, \tag{A.5}$$

where $\kappa_{\mathrm{gap}} \stackrel{\mathrm{def}}{=} \mathbb{E}[f(X_0)] - f_{\mathrm{low}} + \eta\, \varsigma\, N$.

Proof. Taking the full expectation in the conditional descent inequality (A.1) and summing for $j = 0$ to t gives

$$\begin{aligned}\eta \sum_{\ell=1}^{L} \sum_{j \in \mathcal{J}_{t,\ell}} \mathbb{E}\Big[\|Z_{j,\ell}\|_{*,\ell} \left\langle \widetilde{G}_{j,\ell}, \mathcal{N}_{j,\ell}(Z_{j,\ell}) \right\rangle_F\Big] &\leq \mathbb{E}[f(X_0)] - f_{\mathrm{low}} \\ &+ \eta \sum_{\ell=1}^{L} \sum_{j \in \mathcal{J}_{t,\ell}} \mathbb{E}\Big[\|Z_{j,\ell}\|_{*,\ell} \,\Big|\, \left\langle \widetilde{G}_{j,\ell} - G_{j,\ell}, \mathcal{N}_{j,\ell}\ell(Z_{j,\ell}) \right\rangle_F \,\Big|\Big] + \frac{L_G \eta^2}{2} \sum_{\ell=1}^{L} \sum_{j \in \mathcal{J}_{t,\ell}} \mathbb{E}\big[\|Z_{j,\ell}\|_{*,\ell}^2 \|\mathcal{N}_{j,\ell}(Z_{j,\ell})\|_\ell^2\big].\end{aligned} \tag{A.6}$$

Using successively (3.1) and (A.2), we obtain that

$$\begin{aligned}\eta \sum_{\ell=1}^{L} \sum_{j \in \mathcal{J}_{t,\ell}} \mathbb{E}\Big[\|Z_{j,\ell}\|_{*,\ell} \left\langle \widetilde{G}_{j,\ell}, \mathcal{N}_{j,\ell}(Z_{j,\ell}) \right\rangle_F\Big] &= \eta \sum_{\ell=1}^{L} \sum_{j \in \mathcal{J}_{t,\ell}} \mathbb{E}\Big[\mathrm{tr}\big(\Pi_{j,\ell}^{-1/2} \mathcal{U}_{j,\ell}(\widetilde{G}_{j,\ell})^2\big)\Big] \\ &\geq \eta\, \mathbb{E}\left[\sum_{l=1}^{L} \mathrm{tr}(\Pi_{t,\ell}^{1/2}) - \sum_{\ell=1}^{L} \mathrm{tr}(\Pi_{-1,\ell}^{1/2})\right].\end{aligned} \tag{A.7}$$

Now observe that the Cauchy-Schwarz inequality and the definition of the normalization op-

erator $\mathcal{N}_{j,\ell}$ give that

$$|\left\langle \widetilde{G}_{j,\ell} - G_{j,\ell}, \mathcal{N}_\ell(z_{j,\ell})\right\rangle_F| \leq \|\widetilde{G}_{j,\ell} - G_{j,\ell}\|_{*,\ell}\, \|\mathcal{N}_\ell(Z_{j,\ell})\|_\ell = \|\widetilde{G}_{j,\ell} - G_{j,\ell}\|_{*,\ell}.$$

Therefore, for $j \in \mathcal{J}_{t,\ell}$,

$$\mathbb{E}\Big[\|Z_{j,\ell}\|_{*,\ell}\, |\left\langle \widetilde{G}_{j,\ell} - G_{j,\ell}, \mathcal{N}_{j,\ell}(Z_{j,\ell})\right\rangle_F|\Big] \leq \sqrt{\mathbb{E}\Big[\|\widetilde{G}_{j,\ell} - G_{j,\ell}\|_{*,\ell}^2\Big]}\, \sqrt{\mathbb{E}\Big[\|Z_{j,\ell}\|_{*,\ell}^2\Big]}.$$

The Cauchy-Schwarz inequality also implies that, for any vectors a and b with nonnegative components,

$$\sum_j \sqrt{a_j}\sqrt{b_j} \leq \|\sqrt{a}\|_E\, \|\sqrt{b}\|_E = \sqrt{\sum_j a_j}\sqrt{\sum_j\ b_j}. \tag{A.8}$$

Hence, summing over $\ell \in \mathcal{A}_j$ and using (2.2) and (2.7), we obtain that

$$\sum_{\ell\in\mathcal{A}_j} \mathbb{E}\Big[\|Z_{j,\ell}\|_{*,\ell}\, |\left\langle \widetilde{G}_{j,\ell} - G_{j,\ell}, \mathcal{N}_{j,\ell}(Z_{j,\ell})\right\rangle_F|\Big] \leq \sqrt{\sum_{\ell\in\mathcal{A}_j} \mathbb{E}\Big[\|\widetilde{G}_{j,\ell} - G_{j,\ell}\|_*^2\Big]}\, \sqrt{\sum_{\ell\in\mathcal{A}_j} \mathbb{E}\Big[\|Z_{j,\ell}\|_{*,\ell}^2\Big]}.$$

Summing now over $j \in \{0,\ldots,t\}$, noting that $\sum_{j=0}^t \sum_{\ell\in\mathcal{A}_j} = \sum_{\ell=1}^L \sum_{j\in\mathcal{J}_{t,\ell}}$ and using (A.8) again, we deduce that

$$\begin{aligned}
&\eta \sum_{\ell=1}^L \sum_{j\in\mathcal{J}_{t,\ell}} \mathbb{E}\Big[\|Z_{j,\ell}\|_{*,\ell}\, |\left\langle \widetilde{G}_{j,\ell} - G_{j,\ell}, \mathcal{N}_{j,\ell}(z_{j,\ell})\right\rangle_F|\Big] \\
&\qquad \leq \eta \sqrt{\sum_{\ell=1}^L \sum_{j\in\mathcal{J}_{t,\ell}} \mathbb{E}\Big[\|\widetilde{G}_{j,\ell} - G_{j,\ell}\|_*^2\Big]}\, \sqrt{\sum_{\ell=1}^L \sum_{j=0}^t \mathbb{E}\Big[\|Z_{j,\ell}\|_{*,\ell}^2\Big]} \\
&\qquad \leq \eta\nu_t \sqrt{\sum_{j=0}^t \sum_{\ell=1}^L \mathbb{E}\Big[\|Z_{j,\ell}\|_{*,\ell}^2\Big]} + \eta\omega \sum_{j=0}^t \sum_{\ell=1}^L \mathbb{E}\big[\|Z_{j,\ell}\|_{*,\ell}^2\big]
\end{aligned}$$

where we used (3.5) to obtain the last inequality. But, by (3.2), (A.3) and (A.4),

$$\sum_{j=0}^t \sum_{\ell=1}^L \mathbb{E}\big[\|Z_{j,\ell}\|_{*,\ell}^2\big] = \sum_{\ell=1}^L \sum_{j\in\mathcal{J}_{t,\ell}} \mathbb{E}\big[\|Z_{j,\ell}\|_{*,\ell}^2\big] = \sum_{j=0}^t \sum_{\ell=1}^L \mathbb{E}\Big[\mathrm{tr}\big(\Pi_{j,\ell}^{-1}\mathcal{U}_{j,\ell}(\widetilde{G}_{j,\ell})^2\big)\Big] \leq \Delta_t, \tag{A.9}$$

so that

$$\eta \sum_{j=0}^t \sum_{\ell=1}^L \mathbb{E}\Big[\|Z_{j,\ell}\|_{*,\ell}\, |\left\langle \widetilde{G}_{j,\ell} - G_{j,\ell}, \mathcal{N}_{j,\ell}(Z_{j,\ell})\right\rangle_F|\Big] \leq \eta\, \nu_t\, \sqrt{\Delta_t} + \eta\omega\Delta_t. \tag{A.10}$$

Similarly, using the definition of the normalization operator $\mathcal{N}_{j,\ell}$ and (A.9), the quadratic term satisfies

$$\frac{L_G\eta^2}{2} \sum_{\ell=1}^L \sum_{j\in\mathcal{J}_{t,\ell}} \mathbb{E}\big[\|Z_{j,\ell}\|_{*,\ell}^2 \|\mathcal{N}_{j,\ell}(Z_{j,\ell})\|_\ell^2\big] = \frac{L_G\eta^2}{2} \sum_{\ell=1}^L \sum_{j\in\mathcal{J}_{t,\ell}} \mathbb{E}\big[\|Z_{j,\ell}\|_{*,\ell}^2\big] \leq \frac{L_G\eta^2}{2}\Delta_t. \tag{A.11}$$

Substituting (A.7), (A.10) and (A.11) into (A.6) then yields that

$$\eta\, \mathbb{E}\left[\sum_{\ell=1}^L \mathrm{tr}(\Pi_{t,\ell}^{1/2}) - \sum_{\ell=1}^L \mathrm{tr}(\Pi_{-1,\ell}^{1/2})\right] \leq f(X_0) - f_{\rm low} + \eta\, \nu_t\, \sqrt{\Delta_t} + \left(\eta\omega + \frac{L_G\eta^2}{2}\right)\Delta_t, \tag{A.12}$$

which is exactly (A.5) after taking into account that $\Pi_{-1,\ell} = \varsigma I_\ell$ for all $\ell \in \{1,\ldots,L\}$. □

We now recall from [18] a crucial consequence of the previous theorem, which provides a bound Θ_t on the expected trace of all preconditioners at iteration t.

Theorem A.4 Suppose that Assumptions 1, 2, 3 and 6 hold. Then, for all $t \geq 0$,

$$\mathbb{E}\left[\sum_{\ell=1}^{L} \mathrm{tr}(\Pi_{t,\ell}^{1/2})\right] \leq \Theta_t \stackrel{\rm def}{=} \max\left[\, \kappa_\Theta,\, T_t \,\right], \tag{A.13}$$

where

$$\kappa_\Theta = \max\left[e^{\max\left[1,\, \frac{\kappa_0}{2N}\right]},\, \frac{3\kappa_{\rm gap}}{\eta},\, 24N\left(\omega + \frac{L_G}{\eta}\right)\log\left(24N\left(\omega + \frac{L_G}{\eta}\right)\right)\right]$$

with $\kappa_{\rm gap}$ defined in Theorem A.3,

$$T_t = 12\sqrt{N}\,\nu_t \sqrt{\max\left[1, \log\left(12\sqrt{N}\,\nu_t\right)\right]}, \tag{A.14}$$

ν_t being defined in Assumption 6 and $\kappa_0 = -\sum_{\ell=1}^{L} d_\ell \log(d_\ell) - N\log(\varsigma)$.

Proof. See [18, Lemmas 3.6, 3.7 and 3.8, and Theorem 3.9]. In particular, Lemma 3.6 proves that

$$\Delta_t \leq \kappa_0 + 2N \log\left(\mathbb{E}\left[\sum_{\ell=1}^{L} \mathrm{tr}(\Pi_{t,\ell}^{1/2})\right]\right). \tag{A.15}$$

□

This last theorem finally gives us all the ingredients to derive the convergence and complexity results of Theorem 3.1. The bound (A.13), together with Assumption 4, indeed implies convergence of the criticality measure on each block with a *uniform rate-of-convergence bound*. As shown by the next theorem, the expected global rate of convergence is (in order) proportional to the value of Θ_t. The proof of this statement is substantially more involved than [18, Theorem 3.10] because of the introduction of delays.

Proof of Theorem 3.1

Assumption 4 gives that, for each $\ell \in \{1, \ldots, L\}$,

$$\sum_{j \in \mathcal{J}_{t,\ell}} \|\widetilde{G}_{j,\ell}\|_{*,\ell}^2 \leq \kappa_\circ^2 \sum_{j \in \mathcal{J}_{t,\ell}} \mathrm{tr}(\mathcal{U}_{j,\ell}(\widetilde{G}_{j,\ell})^2) = \kappa_\circ^2\, \mathrm{tr}\Big(\sum_{j \in \mathcal{J}_{t,\ell}} \mathcal{U}_{j,\ell}(\widetilde{G}_{j,\ell})^2\Big) \leq \kappa_\circ^2\, \mathrm{tr}(\Pi_{t,\ell})$$

But

$$\mathrm{tr}(\Pi_{t,\ell}) = \sum_{i=1}^{d_\ell} \left(\sqrt{\lambda_i[\Pi_{t,\ell}]}\right)^2 \leq \left(\sum_{i=1}^{d_\ell} \sqrt{\lambda_i[\Pi_{t,\ell}]}\right)^2 = \mathrm{tr}(\Pi_{t,\ell}^{1/2})^2,$$

and thus, summing over $\ell \in \{1, \ldots, L\}$ and using the fact that $\sum_i a_i^2 \leq (\sum_i a_i)^2$,

$$\sqrt{\sum_{\ell=1}^{L} \sum_{j \in \mathcal{J}_{t,\ell}} \|\widetilde{G}_{j,\ell}\|_{*,\ell}^2} \leq \kappa_\circ \sqrt{\sum_{\ell=1}^{L} \mathrm{tr}(\Pi_{t,\ell}^{1/2})^2} \leq \kappa_\circ \sqrt{\left(\sum_{\ell=1}^{L} \mathrm{tr}(\Pi_{t,\ell}^{1/2})\right)^2} = \kappa_\circ \sum_{\ell=1}^{L} \mathrm{tr}(\Pi_{t,\ell}^{1/2}).$$

Using now the Cauchy-Schwarz and Jensen's inequalities with this last bound, we deduce that

$$\begin{aligned}\sum_{\ell=1}^{L}\sum_{j\in\mathcal{J}_{t,\ell}}\mathbb{E}\Big[\|\widetilde{G}_{j,\ell}\|_{*,\ell}\Big] &= \mathbb{E}\left[\sum_{\ell=1}^{L}\sum_{j\in\mathcal{J}_{t,\ell}}\|\widetilde{G}_{j,\ell}\|_{*,\ell}\right]\\ &\le \sqrt{L\max_{\ell\in\{1,\ldots,L\}}|\mathcal{J}_{t,\ell}|}\,\mathbb{E}\left[\sqrt{\sum_{\ell=1}^{L}\sum_{j\in\mathcal{J}_{t,\ell}}\|\widetilde{G}_{j,\ell}\|_{*,\ell}^2}\right] \qquad (A.16)\\ &\le \kappa_\circ\sqrt{L(t+1)}\,\mathbb{E}\left[\sum_{\ell=1}^{L}\mathrm{tr}(\Pi_{t,\ell}^{1/2})\right]\\ &\le \kappa_\circ\sqrt{L(t+1)}\,\Theta_t\end{aligned}$$

where the last inequality results from (A.13). Now let $H_j = (G_{\pi(j,1),1},\ldots,G_{\pi(j,L),L})$. Then

$$\sum_{j=0}^{t}\mathbb{E}[\|G_j\|_*] \le \sum_{j=0}^{t}\mathbb{E}[\|G_j - H_j\|_*] + \sum_{j=0}^{t}\mathbb{E}[\|H_j\|_*]\,. \qquad (A.17)$$

Consider the two terms of the right-hand side separately. To bound the first, we apply Assumption 2 and obtain that

$$\|G_j - G_{\pi(j,\ell)}\|_*^2 \le L_G^2\|X_j - X_{\pi(j,\ell)}\|^2 \le L_G^2\eta^2\sum_{i=\pi(j,\ell)}^{j-1}\|Z_i\|_*^2.$$

Assumption 7 then guarantees that each $\|Z_i\|_*^2$ can only appear τ times when summing over j, and thus that

$$\sum_{j=0}^{t}\sum_{\ell=1}^{L}\mathbb{E}\big[\|G_j - G_{\pi(j,\ell)}\|_*^2\big] \le \tau L_G^2\eta^2\sum_{j=0}^{t}\sum_{\ell=1}^{L}\mathbb{E}\big[\|Z_j\|_*^2\big] = \tau L_G^2\eta^2 L\sum_{j=0}^{t}\mathbb{E}\big[\|Z_j\|_*^2\big]\,.$$

But (A.9), (A.15) and (A.13) imply that

$$\sum_{j=0}^{t}\sum_{\ell=1}^{L}\mathbb{E}\big[\|Z_{j,\ell}\|_{*,\ell}^2\big] \le \Delta_t \le \kappa_0 + 2N\log\left(\mathbb{E}\left[\sum_{\ell=1}^{L}\mathrm{tr}(\Pi_{t,\ell}^{1/2})\right]\right) \le \kappa_0 + 2N\log(\Theta_t). \qquad (A.18)$$

and thus, using (2.2), that

$$\sum_{j=0}^{t}\mathbb{E}\big[\|G_j - H_j\|_*^2\big] = \sum_{j=0}^{t}\sum_{\ell=1}^{L}\mathbb{E}\big[\|G_{j,\ell} - G_{\pi(j,\ell),\ell}\|_{*,\ell}^2\big] \le \tau L_G^2\eta^2 L\left[\kappa_0 + 2N\log(\Theta_t)\right].$$

The Cauchy-Schwarz inequality then gives that

$$\sum_{j=0}^{t}\mathbb{E}[\|G_j - H_j\|_*] \le \sqrt{t+1}\sqrt{\sum_{j=0}^{t}\mathbb{E}[\|G_j - H_j\|_*^2]} \le \sqrt{t+1}\,\sqrt{\tau L_G^2\eta^2 L\left[\kappa_0 + 2N\log(\Theta_t)\right]}. \qquad (A.19)$$

Consider now the second term in the right-hand side of (A.17). We have, again using (2.2), that

$$\sum_{j=0}^{t}\|H_j\|_* = \sum_{j=0}^{t}\sqrt{\sum_{\ell=1}^{L}\|G_{\pi(j,\ell),\ell}\|_{*,\ell}^2} \le \sum_{j=0}^{t}\sum_{\ell=1}^{L}\|G_{\pi(j,\ell),\ell}\|_{*,\ell}$$

Now Assumption 7 ensures each $\|G_{\pi(j,\ell),\ell}\|_{*,\ell}^2$ appears at most τ times in the sum over j. Thus,

$$\sum_{j=0}^{t}\|H_j\|_* \le \sum_{j=0}^{t}\sum_{\ell=1}^{L}\|G_{\pi(j,\ell),\ell}\|_{*,\ell} \le \tau\sum_{\ell=1}^{L}\sum_{j\in\mathcal{J}_{t,\ell}}\|G_{j,\ell}\|_{*,\ell}. \qquad (A.20)$$

But Assumption 5, Jensen's inequality and the law of total expectation ensure that

$$\mathbb{E}[\|G_{j,\ell}\|_{*,\ell}] = \mathbb{E}\Big[\|\mathbb{E}_j\Big[\widetilde{G}_{j,\ell}\Big]\|_{*,\ell}\Big] \leq \mathbb{E}\Big[\mathbb{E}_j\Big[\|\widetilde{G}_{j,\ell}\|_{*,\ell}\Big]\Big] = \mathbb{E}\Big[\|\widetilde{G}_{j,\ell}\|_{*,\ell}\Big].$$

Substituting this identity in (A.20) and using (A.16), we then obtain that

$$\sum_{j=0}^{t} \mathbb{E}[\|H_j\|_*] \leq \tau \sum_{\ell=1}^{L} \sum_{j\in\mathcal{J}_{t,\ell}} \mathbb{E}\Big[\|\widetilde{G}_{j,\ell}\|_{*,\ell}\Big] \leq \tau\kappa_\circ \sqrt{L(t+1)}\,\Theta_t. \tag{A.21}$$

Combining (A.19) and (A.21) and dividing by $t+1$ finally yields (3.6).

Proof of Corollary 3.2

Observe that, when Θ_t (as defined in Theorem A.4) grows with t, the bound (3.6) is dominated by the term $\kappa_c\Theta_t/\sqrt{t+1}$. Since Corollary 3.2 solely aims at describing dominating terms for growing t, we may ignore the other terms and merely consider that (3.6) reduces to

$$\frac{1}{t+1}\sum_{j=0}^{t} \mathbb{E}[\|G_j\|_*] \leq \frac{\kappa_c\Theta_t}{\sqrt{t+1}}, \tag{A.22}$$

which is identical to the result (equation (3.42)) of Theorem 3.10 in [18]. The rest of the proof then follows that of [18, Corollary 3.11] with ν_k^2 replaced by

$$\mu_{\max} \max_{\ell\in\{1,\ldots,L\}} \left[\sum_{j\in\mathcal{J}_{t,\ell}} \frac{1}{\min[|\mathcal{J}_{j,\ell}|]}\right] \leq \begin{cases} \dfrac{\mu_{\max}\sigma_{\rm tot}^2}{1-\alpha}\psi_t^{1-\alpha} & \text{if} \quad \alpha < 1, \\ \mu_{\max}\sigma_{\rm tot}^2 \log(\psi_t) & \text{if} \quad \alpha = 1, \\ \mu_{\max}\sigma_{\rm tot}^2\zeta(\alpha) & \text{if} \quad \alpha > 1, \end{cases}$$

yielding (3.9).

Appendix 2: Detailed analysis for DADPREC with momentum

We start by establishing a bound on the norm of the difference between the block-wise momentum $M_{t,\ell}$ and the approximate gradient $\widetilde{G}_{t,\ell}$.

Lemma A.5 Consider Algorithms DADPREC.MM and DADPREC.MG. Suppose that Assumption 2 holds. Define

$$E_{t,\ell} = M_{t,\ell} - \widetilde{G}_{t,\ell} \quad \text{for} \quad t \geq 0 \ \text{ and } \ \ell \in \mathcal{A}_t. \tag{A.23}$$

Then

$$\sum_{\ell=1}^{L}\sum_{j\in\mathcal{J}_{t,\ell}} \mathbb{E}\big[\|E_{j,\ell}\|_{*,\ell}^2\big] \leq \frac{3}{(1-\mu_{\max})^2}\left[2\sum_{\ell=1}^{L}\sum_{j\in\mathcal{J}_{t,\ell}} \bar{\mu}_{j,\ell}^2\mathbb{E}\Big[\|\widetilde{G}_{j,\ell} - G_{j,\ell}\|_{*,\ell}^2\Big] + \tau L L_G^2\eta^2 \sum_{j=0}^{t}\mathbb{E}\big[\|Z_j\|_*^2\big]\right] \tag{A.24}$$

where $\bar{\mu}_{j,\ell} = \max[\,\mu_{\pi(j,\ell),\ell},\, \mu_{j,\ell}\,]$.

Proof. Suppose that $t \geq 1$ and note that the definition of the momentum and (A.23) give that, for $\ell \in \mathcal{A}_t$,

$$\begin{aligned} E_{t,\ell} &= \mu_{t,\ell}M_{\pi(t-1,\ell),\ell} + (1-\mu_{t,\ell})\widetilde{G}_{t,\ell} - \widetilde{G}_{t,\ell} \\ &= \mu_{t,\ell}(M_{\pi(t-1,\ell),\ell} - \widetilde{G}_{\pi(t-1,\ell),\ell}) + \mu_{t,\ell}(\widetilde{G}_{\pi(t-1,\ell),\ell} - G_{\pi(t-1,\ell),\ell}) + \mu_{t,\ell}(G_{\pi(t-1,\ell),\ell} - \widetilde{G}_{t,\ell}) \\ &= \mu_{t,\ell}E_{\pi(t-1,\ell),\ell} + \mu_{t,\ell}(\widetilde{G}_{\pi(t-1,\ell),\ell} - G_{\pi(t-1,\ell),\ell}) + \mu_{t,\ell}(G_{\pi(t-1,\ell),\ell} - \widetilde{G}_{t,\ell}) \end{aligned}$$

Thus, using the facts that $\mu_{t,\ell} \leq \mu_{\max} < 1$, that $(a+b)^2 \leq (1+\rho)a^2 + (1+1/\rho)b^2$ for any $\rho > 0$ and that $(a+b+c)^2 \leq 3a^2 + 3b^2 + 3c^2$, we obtain that

$$\begin{aligned}\|E_{t,\ell}\|_{*,\ell}^2 \leq & (1+\rho)\mu_{\max}^2 \|E_{\pi(t-1,\ell),\ell}\|_{*,\ell}^2 \\ & + 3\left(1+\frac{1}{\rho}\right)\mu_{t,\ell}^2 \left[\|G_{t,\ell} - G_{\pi(t-1,\ell),\ell}\|_{*,\ell}^2 + \|\widetilde{G}_{\pi(t-1,\ell),\ell} - G_{\pi(t-1,\ell),\ell}\|_{*,\ell}^2 + \|\widetilde{G}_{t,\ell} - G_{t,\ell}\|_{*,\ell}^2\right].\end{aligned}$$

Now let $\rho = (1-\mu_{\max})/\mu_{\max}$. Then $(1+\rho)\mu_{\max}^2 = \mu_{\max} < 1$ and $(1+1/\rho) = 1/(1-\mu_{\max})$. Hence, summing over $j \in \mathcal{J}_{t,\ell}$, we obtain that

$$\begin{aligned}\sum_{\substack{j\in\mathcal{J}_{t,\ell}\\ j\geq 1}} \|E_{j,\ell}\|_{*,\ell}^2 \leq & \mu_{\max} \sum_{j\in\mathcal{J}_{k,\ell}} \|E_{\pi(j-1,\ell),\ell}\|_{*,\ell}^2 \\ & + \frac{3}{1-\mu_{\max}} \sum_{j\in\mathcal{J}_{t,\ell}} \mu_{j,\ell}^2 \left[\|G_{j,\ell} - G_{\pi(j-1,\ell),\ell}\|_{*,\ell}^2 + \|\widetilde{G}_{\pi(j-1,\ell),\ell} - G_{\pi(j-1,\ell),\ell}\|_{*,\ell}^2 + \|\widetilde{G}_{j,\ell} - G_{j,\ell}\|_{*,\ell}^2\right]\end{aligned}$$

Observe now that (2.2), Assumption 2 with (2.5) and the definition of the normalization operator imply that

$$\begin{aligned}\|G_{j,\ell} - G_{\pi(j-1,\ell),\ell}\|_{*,\ell}^2 &\leq \|G_j - G_{\pi(j-1,\ell)}\|_*^2 \\ &\leq L_G^2 \sum_{q=1}^{L} \|X_{j,q} - X_{\pi(j-1,\ell),q}\|_q^2 \\ &\leq L_G^2 \sum_{q=1}^{L} \sum_{s=\pi(j-1,\ell)}^{j-1} \|X_{s+1,q} - X_{s,q}\|_q^2 \\ &= L_G^2 \eta^2 \sum_{q=1}^{L} \sum_{s=\pi(j-1,\ell)}^{j-1} \|Z_{s,q}\|_{*,q}^2 \|\mathcal{N}_{s,q}(Z_{s,q})\|_q^2 \\ &= L_G^2 \eta^2 \sum_{q=1}^{L} \sum_{s=\pi(j-1,\ell)}^{j-1} \|Z_{s,q}\|_{*,q}^2.\end{aligned} \tag{A.25}$$

But Assumption 7 implies that the sum over s in the last right-hand side contains at most τ terms. Thus, using also (2.2) and the fact that $\mu_{j,\ell} < 1$,

$$\sum_{\substack{j\in\mathcal{J}_{t,\ell}\\ j\geq 1}} \mu_{j,\ell}^2 \|G_{j,\ell} - G_{\pi(j-1,\ell),\ell}\|_{*,\ell}^2 \leq \tau L_G^2 \eta^2 \sum_{q=1}^{L} \sum_{s=0}^{t-1} \|Z_{s,q}\|_{*,q}^2 = \tau L_G^2 \eta^2 \sum_{s=0}^{t-1} \|Z_s\|_*^2.$$

Summing now over $\ell \in \{1,\ldots,L\}$, taking full expectation and defining $\vartheta_{t,\ell}^2 = \mathbb{E}\left[\|\widetilde{G}_{t,\ell} - G_{t,\ell}\|_{*,\ell}^2\right]$, we deduce that

$$\begin{aligned}\sum_{\ell=1}^{L} \sum_{\substack{j\in\mathcal{J}_{t,\ell}\\ j\geq 1}} \mathbb{E}\big[\|E_{j,\ell}\|_{*,\ell}^2\big] &\leq \mu_{\max} \sum_{\ell=1}^{L} \sum_{\substack{j\in\mathcal{J}_{t,\ell}\\ j\geq 1}} \mathbb{E}\big[\|E_{\pi(j-1,\ell),\ell}\|_{*,\ell}^2\big] \\ &\qquad + \frac{3}{1-\mu_{\max}} \left[\tau L_G^2 \eta^2 \sum_{\ell=1}^{L} \sum_{s=0}^{t-1} \mathbb{E}\big[\|Z_s\|_*^2\big] + \sum_{\ell=1}^{L} \sum_{\substack{j\in\mathcal{J}_{t,\ell}\\ j\geq 1}} \mu_{j,\ell}^2 (\vartheta_{\pi(j-1,\ell),\ell}^2 + \vartheta_{j,\ell}^2)\right] \\ &\leq \mu_{\max} \sum_{\ell=1}^{L} \sum_{\substack{j\in\mathcal{J}_{t,\ell}\\ j\geq 1}} \mathbb{E}\big[\|E_{j,\ell}\|_{*,\ell}^2\big] + \frac{3}{1-\mu_{\max}} \left[\tau L L_G^2 \eta^2 \sum_{s=0}^{t-1} \mathbb{E}\big[\|Z_s\|_*^2\big] + 2 \sum_{\ell=1}^{L} \sum_{\substack{j\in\mathcal{J}_{t,\ell}\\ j\geq 1}} \bar{\mu}_{j,\ell}^2 \vartheta_{j,\ell}^2\right],\end{aligned}$$

so that

$$\sum_{\ell=1}^{L}\sum_{\substack{j\in\mathcal{J}_{t,\ell}\\ j\geq 1}}\mathbb{E}\big[\|E_{j,\ell}\|_{*,\ell}^2\big]\leq\frac{3}{(1-\mu_{\max})^2}\left[\tau LL_G^2\eta^2\sum_{s=0}^{t-1}\mathbb{E}\big[\|Z_s\|_*^2\big]+2\sum_{\ell=1}^{L}\sum_{\substack{j\in\mathcal{J}_{t,\ell}\\ j\geq 1}}\bar{\mu}_{j,\ell}^2\vartheta_{j,\ell}^2\right]$$

which, with the fact that $E_0=\widetilde{G}_0-G_0$ and (2.2), concludes the proof of (A.24). □

Proof of Theorem 4.1

Observe now that the DADPREC.MM algorithm is nothing but Algorithm DADPREC where the momentum $M_{t,\ell}$ plays the role of the approximate gradient $\widetilde{G}_{t,\ell}$. Moreover Assumption 8 exactly reflects this change of perspective. Also note that, for all $\ell\in\{1,\ldots,L\}$ and $j\in\mathcal{J}_{t,\ell}$,

$$\|M_{j,\ell}-G_{j,l}\|_{*,\ell}^2\leq 2\|E_{j,\ell}\|_{*,\ell}^2+2\|\widetilde{G}_{j,\ell}-G_{j,l}\|_{*,\ell}^2.$$

Hence, using Lemma A.5 and given that $\bar{\mu}_{j,\ell}<1$, we have that

$$\sum_{\ell=1}^{L}\sum_{j\in\mathcal{J}_{t,\ell}}\mathbb{E}\big[\|M_{j,\ell}-G_{j,l}\|_{*,\ell}^2\big]<\left(\frac{3}{(1-\mu_{\max})^2}+2\right)\sum_{j=0}^{t}\mathbb{E}\Big[\|\widetilde{G}_j-G_j\|_*^2\Big]+\frac{2[\tau LL_G^2\eta^2}{(1-\mu_{\max})^2}\sum_{j=0}^{t}\mathbb{E}\big[\|Z_j\|_*^2\big].$$

This provides an adapted formulation to replace (3.5) in Assumption 6, which thus continues to hold with

$$\nu_t^2=\left(\frac{3}{(1-\mu_{\max})^2}+2\right)\sum_{j=0}^{t}\mathbb{E}\Big[\|\widetilde{G}_j-G_j\|_*^2\Big]\quad\text{and}\quad\omega^2=\frac{2\tau LL_G^2\eta^2}{(1-\mu_{\max})^2}.\tag{A.26}$$

As a consequence, the theory developed for the DADPREC algorithm also holds for its DADPREC.MM variant and this proves Theorem 4.1.

Proof of Theorem 4.2

This section considers the convergence of the DADPREC.MG algorithm. We first recall a result on the impact of using this algorithm on the structural identities of Assumption 8.

Lemma A.6 Suppose that Assumptions 2 and 8 and (4.3) hold. Then

$$\|Z_{t,\ell}\|_{*,\ell}\,\langle G_{t,\ell},\mathcal{N}_{t,\ell}(Z_{t,\ell})\rangle_F\geq\frac{1}{\kappa_\square}\,\mathrm{tr}\big(\Pi_{t,\ell}^{-1/2}\mathcal{U}_{t,\ell}(\widetilde{G}_{t,\ell})^2\big)-\mathrm{tr}\big(\Pi_{t,\ell}^{-1/2}\mathcal{U}_{t,\ell}(E_{t,\ell})^2\big)-\|Z_{t,\ell}\|_{*,\ell}\|E_{t,\ell}\|_{*,\ell}\tag{A.27}$$

and

$$\|Z_{t,\ell}\|_{*,\ell}^2\leq\kappa_\square\mathrm{tr}\big(\Pi_{t,\ell}^{-1}\mathcal{U}_{t,\ell}(G_{t,\ell})^2\big)+\kappa_\square\,\mathrm{tr}\big(\Pi_{t,\ell}^{-1}\mathcal{U}_{t,\ell}(E_{t,\ell})^2\big)\tag{A.28}$$

Proof. See [18, Lemmas A.1 and A.2]. □

Lemma A.7 Suppose that that Assumption 2 and 8 and (4.3) hold. Then

$$\sum_{\ell=1}^{L}\sum_{j\in\mathcal{J}_{t,\ell}}\mathbb{E}[\|Z_{j,\ell}\|_{*,\ell}\|E_{j,\ell}\|_{*\ell}] \leq \frac{12\theta_t^2}{(1-\mu_{\max})^2} + \left(\frac{6\tau LL_G^2\eta^2}{(1-\mu_{\max})^2}+2\right)\sum_{j=0}^{t}\mathbb{E}\big[\|Z_j\|_*^2\big]\,, \quad \text{(A.29)}$$

$$\sum_{j=0}^{t}\sum_{\ell=1}^{L}\mathbb{E}\Big[\mathrm{tr}\big(\Pi_{j,\ell}^{-1/2}\mathcal{U}_{t,\ell}(E_{j,\ell})^2\big)\Big] \leq \frac{6\kappa_\diamond\theta_t^2}{(1-\mu_{\max})^2\sqrt{\varsigma}} + \frac{3\kappa_\diamond\tau LL_G^2\eta^2}{(1-\mu_{\max})^2\sqrt{\varsigma}}\sum_{j=0}^{t}\mathbb{E}\big[\|Z_j\|_*^2\big] \quad \text{(A.30)}$$

and

$$\sum_{j=0}^{t}\sum_{\ell=1}^{L}\mathbb{E}\Big[\mathrm{tr}\big(\Pi_{t,\ell}^{-1}\mathcal{U}_{t,\ell}(E_{t,\ell})^2\big)\Big] \leq \frac{6\kappa_\diamond\theta_t^2}{(1-\mu_{\max})^2\varsigma} + \frac{3\kappa_\diamond\tau LL_G^2\eta^2}{(1-\mu_{\max})^2\varsigma}\sum_{j=0}^{t}\mathbb{E}\big[\|Z_j\|_*^2\big]\,. \quad \text{(A.31)}$$

Proof. See [18, Lemma A.3]. □

Lemma A.8 Suppose that Assumption 2 and 8 and (4.3) hold. Suppose in addition that (4.4) is satisfied. Then

$$\begin{aligned}\sum_{j=0}^{t}\sum_{\ell=1}^{L}\mathbb{E}\big[\|Z_{j,\ell}\|_{*,\ell}\,\langle G_{j,\ell},\mathcal{N}_{j,\ell}(Z_{j,\ell})\rangle_F\big] \geq & \frac{1}{\kappa_\square}\sum_{j=0}^{t}\sum_{\ell=1}^{L}\mathbb{E}\Big[\mathrm{tr}\big(\Pi_{t,\ell}^{-1/2}\mathcal{U}_{t,\ell}(\widetilde{G}_{t,\ell})^2\big)\Big] \\ & - \kappa_{1,\nu}\,\theta_t^2 - \kappa_{1,z}\sum_{j=0}^{t}\mathbb{E}\big[\|Z_j\|_*^2\big]\end{aligned} \quad \text{(A.32)}$$

with

$$\kappa_{1,\nu} = \frac{6\kappa_\diamond}{(1-\mu_{\max})^2\sqrt{\varsigma}} + \frac{12}{(1-\mu_{\max})^2} \quad \text{and} \quad \kappa_{1,z} = \frac{3\kappa_\diamond\tau LL_G^2\eta^2}{(1-\mu_{\max})^2\sqrt{\varsigma}} + \frac{6\tau LL_G^2\eta^2}{(1-\mu_{\max})^2} + 2, \quad \text{(A.33)}$$

and

$$\sum_{j=0}^{t}\mathbb{E}\big[\|Z_j\|_*^2\big] \leq 2\sum_{j=0}^{t}\sum_{\ell=1}^{L}\mathbb{E}\Big[\mathrm{tr}\big(\Pi_{j,\ell}^{-1}\mathcal{U}_{j,\ell}(G_{j,\ell})^2\big)\Big] + \kappa_{2,\nu}\,\theta_t^2 + \kappa_{2,z}, \quad \text{(A.34)}$$

where

$$\kappa_{2,\nu} = \frac{6\kappa_\square\kappa_\diamond}{(1-\mu_{\max})^2\varsigma} \quad \text{and} \quad \kappa_{2,z} = \begin{cases} 0 & \text{if the first part of (4.4) holds,} \\ \frac{3\kappa_\square\kappa_\diamond\tau LL_G^2\eta^2}{(1-\mu_{\max})^2\varsigma}\,\kappa_Z & \text{if the second part of (4.4) holds.}\end{cases} \quad \text{(A.35)}$$

Proof. The inequality (A.32) is obtained by substituting (A.29) and (A.30) into (A.27), Moreover, taking the expectation in (A.28), summing for $j \in \{0,\ldots,t\}$ and $\ell \in \{1,\ldots,L\}$ and substituting (A.31) gives that

$$\begin{aligned}\sum_{j=0}^{t}\sum_{\ell=1}^{L}\mathbb{E}\big[\|Z_{j,\ell}\|_{*,\ell}^2\big] \leq & \sum_{j=0}^{t}\sum_{\ell=1}^{L}\mathbb{E}\Big[\mathrm{tr}\big(\Pi_{j,\ell}^{-1}\mathcal{U}_{j,\ell}(G_{j,\ell})^2\big)\Big] + \frac{6\kappa_\square\kappa_\diamond}{(1-\mu_{\max})^2\varsigma}\,\theta_t^2 \\ & + \frac{3\kappa_\square\kappa_\diamond\tau LL_G^2\eta^2}{(1-\mu_{\max})^2\varsigma}\sum_{j=0}^{t}\mathbb{E}\big[\|Z_j\|_*^2\big]\,.\end{aligned}$$

Suppose first that the first part of (4.4) holds. Then

$$\frac{3\kappa_{\square}\kappa_{\diamond}\tau L L_G^2\eta^2}{(1-\mu_{\max})^2\varsigma} \leq \frac{1}{2}$$

and (A.34) follows with $\kappa_{2,z} = 0$. Alternatively, if the second part of (4.4) holds, then (A.34) follows with

$$\kappa_{2,z} = \frac{3\kappa_{\square}\kappa_{\diamond}\tau L L_G^2\eta^2}{(1-\mu_{\max})^2\varsigma}\,\kappa_{\mu Z}.$$

□

From this point on, the analysis follows the lines of the argument of Appendix 1 with Lemma A.8 providing an alternate set of structural inequalities. Lemma A.2 is unchanged. The modifications to Theorem A.3 are minor. It is restated as follows.

Theorem A.9 Suppose that Assumption 1 and 8 and (4.4) hold. Define Δ_j as in (A.4). Then, for every $k \geq 0$,

$$\eta\,\mathbb{E}\left[\sum_{\ell=1}^{L}\mathrm{tr}(\Pi_{t,\ell}^{1/2})\right] \leq \kappa_{\rm gap} + \kappa_{\nu\nu}\theta_t^2 + \eta\,\kappa_{\nu\Delta}\,\theta_t\,\sqrt{\Delta_t} + \eta\,\kappa_{\Delta}\,\Delta_t, \tag{A.36}$$

where

$$\kappa_{\rm gap} \stackrel{\rm def}{=} \mathbb{E}[f(X_0)] - f_{\rm low} + \eta\,\varsigma\,N + \eta\sqrt{\kappa_{2,z}} + \left(\eta\kappa_{1,z} + \frac{L_G\eta^2}{2}\right)\kappa_{2,z} \tag{A.37}$$

$$\kappa_{\nu\nu} = \eta\left(\kappa_{1,\nu} + \sqrt{\kappa_{2,\nu}} + \kappa_{1,z}\kappa_{2,\nu} + \frac{L_G\eta}{2}\right), \quad \kappa_{\nu\Delta} = \sqrt{2} \quad \text{and} \quad \kappa_{\Delta} = 2\kappa_{1,z} + L_G\eta. \tag{A.38}$$

Proof. In the proof of Theorem A.3, a perturbation $\eta\kappa_{1,\nu}\,\theta_t^2 + \eta\kappa_{1,z}\,\sum_{j=0}^{t}\sum_{\ell=1}^{L}\mathbb{E}\Big[\|Z_{t,\ell}\|_{*,\ell}^2\Big]$ is subtracted from the right-hand side of (A.7) in order to reflect (A.32). The proof then goes on unmodified, up to the substitution leading to (A.12), in which the perturbed (A.7) then gives that

$$\eta\,\mathbb{E}\left[\sum_{\ell=1}^{L}\mathrm{tr}(\Pi_{t,\ell}^{1/2}) - \sum_{\ell=1}^{L}\mathrm{tr}(\Pi_{-1,\ell}^{1/2})\right] \leq f(X_0) - f_{\rm low} + \eta\kappa_{1,\nu}\theta_t^2 + \eta\theta_t\sqrt{\zeta_t} + \left(\eta\kappa_{1,z} + \frac{L_G\eta^2}{2}\right)\zeta_t.$$

where $\zeta_t = \sum_{j=0}^{t}\sum_{\ell=1}^{L}\mathbb{E}\Big[\|Z_{j,\ell}\|_{*,\ell}^2\Big]$. But this definition, (A.34) and (A.3) give that

$$\zeta_t \leq \kappa_{2,z} + \kappa_{2,\nu}\,\theta_t^2 + 2\sum_{j=0}^{t}\sum_{\ell=1}^{L}\mathbb{E}\Big[\mathrm{tr}\big(\Pi_{j,\ell}^{-1}\mathcal{U}_{j,\ell}(G_{j,\ell})^2\big)\Big] \leq \kappa_{2,z} + \kappa_{2,\nu}\,\theta_t^2 + 2\Delta_t,$$

and thus, using $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$,

$$\begin{aligned}\eta\,\mathbb{E}\left[\sum_{\ell=1}^{L}\mathrm{tr}(\Pi_{t,\ell}^{1/2}) - \sum_{\ell=1}^{L}\mathrm{tr}(\Pi_{-1,\ell}^{1/2})\right] &= f(X_0) - f_{\rm low} + \eta\sqrt{\kappa_{2,z}} + \eta\left(\kappa_{1,\nu} + \sqrt{\kappa_{2,\nu}} + \kappa_{1,z}\kappa_{2,\nu} + \frac{L_G\eta}{2}\right)\theta_t^2 \\ &\quad + \sqrt{2}\eta\theta_t\sqrt{\Delta_t} + 2\left(\eta\kappa_{1,z} + \frac{L_G\eta^2}{2}\right)\Delta_t + \left(\eta\kappa_{1,z} + \frac{L_G\eta^2}{2}\right)\kappa_{2,z}\end{aligned}$$

yielding (A.36)-(A.38) after taking into account that $\Pi_{-1,\ell} = \varsigma I_\ell$ for all $\ell \in \{1,\ldots,L\}$. □

Note that the form of bound (A.36) differs very little from that of (A.5): besides using different constants, (A.36) now involves a term in $\kappa_{\nu\nu}\theta_t^2$. This term then percolates through the proof of Theorem A.4, so that (A.13) now becomes

$$\mathbb{E}\left[\sum_{\ell=1}^{L} \mathrm{tr}(\Pi_{t,\ell}^{1/2})\right] \leq \Theta_t \stackrel{\rm def}{=} \max\left[\kappa_\Theta, \frac{3\kappa_{\nu\nu}\theta_t^2}{\eta}, T_t\right]. \tag{A.39}$$

The proof of Theorem 3.1 then stands without modification other than using this updated value of Θ_t, finally yielding Theorem 4.2.

Proof of Corollary 4.3

Finally, the proof of Corollary 4.3 is similar in spirit to that of Corollary 3.11 in [18], with two twists. The first is that the formula for the block-wise variance is now given by (4.8), which, together with (4.9), implies[11] that

$$\begin{aligned}\sum_{\ell=1}^{L}\sum_{j\in\mathcal{J}_{t,\ell}} \|M_{j,\ell} - G_{j,\ell}\|_{*,\ell}^2 &\leq \mu_{\max}\sum_{\ell=1}^{L}\sigma_\ell^2 \sum_{j\in\mathcal{J}_{t,\ell}} \frac{1}{\min[|\mathcal{J}_{j,\ell}|, |\mathcal{J}_{j-1,\ell}|]^{\alpha+2\beta}} + \omega^2\sum_{j=0}^{t}\mathbb{E}\big[\|Z_{j,\ell}\|_{*,\ell}^2\big] \\ &\leq \mu_{\max}\sigma_{\rm tot}^2\sum_{j=1}^{t}\frac{1}{|\mathcal{J}_{j-1,\ell}|^{\alpha+2\beta}} + \omega^2\sum_{j=0}^{t}\mathbb{E}\big[\|Z_{j,\ell}\|_{*,\ell}^2\big].\end{aligned}$$

Therefore, using $\psi_{t-1} \leq \psi_t$, we may set the bound

$$\theta_t^2 = \begin{cases} \frac{\sigma_{\rm tot}^2}{1-\alpha-2\beta}\psi_t^{1-\alpha-2\beta} & \text{if} \quad \alpha+2\beta<1, \\ \sigma_{\rm tot}^2\log(\psi_t) & \text{if} \quad \alpha+2\beta=1, \\ \sigma_{\rm tot}^2\zeta(\alpha+2\beta) & \text{if} \quad \alpha+2\beta>1. \end{cases} \tag{A.40}$$

The second is the presence of term $3\kappa_{\nu\nu}\theta_t^2$ in (4.6). Since the dominant term in the convergence bound (4.5) is the last, the final order of convergence is determined by the maximal power of θ_t occuring in the expression of Θ_t, which is now θ_t^2 because of this new term. Thus substituting (A.40) in (4.5) and only keeping the dominant term in t gives (4.10). □

Appendix 3: Implementation and additional results for asynchronous momentum variants

We first describe our warm-started power iteration to compute an approximation of the nuclear norm of G. It is based on the proxy

$$\|G\|_* \approx \frac{\|G\|_F^2}{\sigma_{\max}[G]}$$

where $\sigma_{\max}[G]$ is the largest singular value of G. Note that $\|G\|_F^2$ is easily computable, but we estimate $\sigma_{\max}[G]$ using two iterations of the power iterations on G. This does not ensure a convergent estimator, but gives a deterministic lower bound

$$\frac{\|G\|_F^2}{\sigma_{\max}[G]} = \frac{\sum_{i=1}^n \sigma_i[G]^2}{\sigma_{\max}[G]} \leq \sum_{i=1}^n \sigma_i[G] = \|G\|_*,$$

where equality holds if and only if all singular values are identical. Our algorithm is warm-started: we initialize the power iteration with the result obtained at the previous optimisation iteration. As the gradient G changes progressively, so does its dominant singular vector, and two iterations are typically enough to obtain an accurate estimation est($\|G\|_*$). The detailed algorithm is given on the next page.

Algorithm A.1: WSPOWER

Given: a starting vector v resulting from the previous optimization iteration (a random vector at the first).

1. Compute $u = Gv$, $u = u/\|u\|$, $v = G^T u$, $\sigma_{\max} = \|v\|$, $v = v/\|v\|$ (repeat twice)
2. Set $\text{est}(\|G\|_*) = \frac{1}{\sigma_{\max}} \sum_{i=1}^{m} \sum_{j=1}^{m} G_{i,j}^2$,
3. Save v for the next call.

The cost of using WSPOWER for an $m \times n$ matrix amounts to $\mathcal{O}(4mn)$ flops for the four matrix-vector products, plus $\mathcal{O}(mn)$ flops for $\|G\|_F^2$, giving a total of $\mathcal{O}(5mn)$ flops, compared to $\mathcal{O}(mn \min(m,n))$ flops for the SVD. In our experiments with 256×2048 matrices, using WSPOWER is $\min(m,n)/5 \approx 50$ times faster.

Although a mere proxy of $\|G\|_*$, WSPOWER works remarkably well, but its warm-start feature is crucial. Without it, accuracy results diverge on larger models (SVHN: 56%, MoE-FMNIST: 77%, to compare with SVHN 70.44% ± 0.43, MoE 88.63% ± 0.34 with warm start, this last result being the best achieved for this benchmark in all our runs) The throughput of our asynchronous MG optimization variants using WSPOWER then become comparable that of MM variants, as shown in Table A.10 below.

We now complete the results mentioned in Section 6.5. Table A.8 gives the final loss values obtained when using different asynchronous momentum variants.

Dataset	No Mom.	MMconst	MGconst	MGconstA	MMdecay	MGdecay	MGdecayA
Fashion-MNIST	0.308	**0.239**	0.336	0.259	0.267	0.316	0.273
MoE-FMNIST	0.279	0.240	0.245	**0.186**	0.290	0.296	0.288
CovType	0.240	**0.149**	0.226	0.173	0.225	0.256	0.229
MovieLens	0.353	**0.146**	0.347	0.209	0.344	0.397	0.305
Criteo	0.088	**0.008**	0.107	0.027	0.081	0.126	0.041
SVHN	0.819	0.856	0.954	0.798	**0.762**	0.932	0.809

Table A.8: Final loss value for asynchronous momentum variants

We see that the MMconst variant dominates on CovType (0.149), Criteo (0.008!), Fashion-MNIST (0.239)and MovieLens (0.146), while MGconstA is best for MoE-FMNIST and MGdecay for SVHN.

The generalization metrics for the different variants with momentum are given in Table A.9.

Dataset	Metric	No Mom.	MMconst	MGconst	MGconstA	MMdecay	MGdecay	MGdecayA
Fashion-MNIST	acc.(%)	87.34	88.16	86.31	87.86	**88.19**	87.00	87.92
MoE-FMNIST	acc.(%)	87.39	87.97	88.11	**88.63**	86.93	86.99	87.11
CovType	acc.(%)	89.56	**92.71**	90.01	91.92	90.13	88.88	90.07
MovieLens	RMSE	0.916	0.992	**0.914**	0.956	0.915	0.923	0.926
Criteo	AUC	0.717	0.720	**0.728**	0.716	0.716	0.723	0.708
SVHN	acc.(%)	69.39	68.69	65.10	70.44	**71.55**	66.19	69.45

Table A.9: Generalization metrics for asynchronous momentum variants

Care should however be exercised: compared to the momentum-less option, the loss for MovieLens improves from 0.353 to 0.008, but the accuracy worsens from 0.916 to 0.992, illustrating a possible overfitting in that case.

The throughputs per problem are given in Table A.10, which shows that the introduction of the MM momentum has a limited effect on throughput, but that the MG approach has a clear negative impact, due to the double decomposition that we pinpointed in Section 6.5. Fortunately, the use of the warm-started power iteration to approximate the nuclear norm of $\widetilde{G}_\ell$ essentially corrects this deficiency.

[11] With the convention that $|\mathcal{J}_{-1,\ell}| = 1$ for all $\ell \in \{1, \ldots, L\}$.

Dataset	No Mom.	MMconst	MGconst	MGconstA	MMdecay	MGdecay	MGdecayA
Fashion-MNIST	10	9	5	9	10	6	8
MoE-FMNIST	42	34	18	34	28	21	27
CovType	105	95	56	99	105	67	101
MovieLens	89	90	57	84	79	50	87
Criteo	89	84	72	93	82	58	96
SVHN	1	1	1	1	1	1	1

Table A.10: Throughput for asynchronous momentum variants

Finally focussing on the classification problems, we report, in Table A.11, the variance of the accuracy obtained for the 8 independent runs. This table suggests that the variance on accuracy is nearly always better for the MM momentum than for the MG approach. In particular, a comparison of MMconst with MGconst shows that the former improves the accuracy of all classification problems by as much as 3.58% (for SVHN).

Dataset	No Mom.	MMconst	MGconst	MGconstA	MMdecay	MGdecay	MGdecayA
Fashion-MNIST	±0.23	±0.23	±0.26	±0.30	±**0.09**	±0.18	±0.22
MoE-FMNIST	±**0.30**	±0.33	±0.38	±0.34	±0.79	±0.38	±0.53
CovType	±**0.11**	±0.14	±0.28	±0.17	±0.18	±0.24	±0.20
SVHN	±2.78	±1.15	±1.54	±**0.43**	±1.27	±2.10	±1.22

Table A.11: Accuracy variance for asynchronous momentum variants on classification problems