On the Convergence Analysis of Asynchronous SGD for Solving Consistent Linear Systems

Atal Narayan Sahu[a], Aritra Dutta[a,b,c,d,*], Aashutosh Tiwari[e], Peter Richtárik[b]

[a] *Extreme Computing Research Center (ECRC), King Abdullah University of Science and Technology (KAUST), Saudi Arabia*
[b] *Visual Computing Center (VCC), KAUST, Saudi Arabia*
[c] *Department of Mathematics and Computer Science (IMADA), University of Southern Denmark (SDU), Denmark*
[d] *Pioneer Centre for AI (P1), Denmark*
[e] *Indian Institute of Technology, Kanpur*

Abstract

Parallel and distributed stochastic algorithms have drawn significant attention in the realm of big data and machine learning. While the synchronous versions of these algorithms are well understood in terms of their convergence, the convergence analyses of their asynchronous counterparts are not widely studied. In this paper, we propose and analyze a *distributed, asynchronous* parallel algorithm to solve an arbitrary, consistent linear system by reformulating the system into a stochastic optimization problem as Richtárik and Takáč in [1]. We compare the convergence rates of our asynchronous algorithm with the synchronous parallel algorithm proposed by Richtárik and Takáč in [1] under different choices of the hyperparameters—the stepsize, the damping factor, the number of processors, and the delay factor. We show that our asynchronous parallel algorithm enjoys a global linear convergence rate, similar to the synchronous parallel algorithm in [1] under the same setup. We also show that our asynchronous algorithm improves upon the synchronous algorithm in [1] with a better convergence rate when the number of processors is larger than four. Furthermore, our asynchronous parallel algorithm performs asymptotically better than its synchronous counterpart for certain

*Corresponding author
Email addresses: atal.sahu@kaust.edu.sa (Atal Narayan Sahu), aritra.dutta@kaust.edu.sa (Aritra Dutta), ashut669@gmail.com (Aashutosh Tiwari), peter.richtarik@kaust.edu.sa (Peter Richtárik)
URL: https://www.aritradutta.com/ (Aritra Dutta), https://richtarik.org/ (Peter Richtárik)

linear systems. Finally, we compute the minimum number of processors required for those systems for which our asynchronous algorithm is better and find that this number can be as low as two for some ill-conditioned problems.

Keywords: Linear systems, distributed optimization, stochastic optimization, asynchronous communication, parallel algorithms, iterative methods
2020 MSC: 15A06, 15B52, 65F10, 65Y20, 68Q25, 68W20, 68W40, 90C20

1. Introduction

Optimization problems have become increasingly complex in big data and machine learning (ML). Although computers are now more powerful and inexpensive, these complex problems evaluated on large datasets are challenging to maneuver by a single processor [2]. In contrast to the traditional optimization algorithms designed to run on a single processor, parallel and distributed algorithms can handle large data more efficiently. To deal with large datasets, parallel algorithms take advantage of multiple processors. Each processor accesses and processes its data partition in small batches, called *mini-batches*, and synchronizes their updates; this process continues until convergence. Shalev-Shwartz et al. [3] in 2007, and Gimpel et al. [4] in 2010 explored the mini-batch stochastic algorithms in both the serial and parallel settings. In 2011, Dekel et al. [5] proposed a distributed mini-batch algorithm (for online predictions) — a method that converts many serial gradient-based online prediction algorithms into distributed algorithms with an asymptotically optimal regret bound. However, the synchronous parallel algorithms tend to slow down in a distributed setting due to unpredictable communication faults, network latency [6], and processors with different processing speeds [7].

Asynchronous algorithms were first introduced by Chazan and Miranker on chaotic relaxation in 1969 [8] (also, see Frommer and Szyld [9], and [10]), and recent research shows they can remedy the slowdown of synchronous parallel algorithms. Processors with different processing speeds and storage capacities perform updates without synchronizing with others in asynchronous algorithms. However, the inherent dynamics of asynchronous algorithms are challenging compared to their synchronous counterparts,

and their convergence analyses are much more mathematically involved. In the literature, the comparisons between the convergence rates of the asynchronous algorithms and their synchronous counterparts are less understood. This paper aims to understand the convergence rates of the asynchronous and synchronous algorithms in a relatively simple setup to solve an arbitrary, consistent linear system. Before discussing the problem formulation, setup, and contribution, we start with a brief overview of stochastic optimization.

Stochastic optimization. In modern ML and data-fitting applications, stochastic optimization is a broadly studied field. Consider the *stochastic optimization problem*:

$$\min_{x\in\mathbb{R}^n} f(x) = \min_{x\in\mathbb{R}^n} \mathbb{E}_{\mathbf{S}\sim\mathcal{D}}[f_{\mathbf{S}}(x)], \tag{1}$$

where $\mathcal{D}$ is a distribution and $\mathbf{S}$ is a random sample drawn from the distribution. In supervised ML, (1) is refer to *empirical risk minimization* (ERM) problem:

$$\min_{x\in\mathbb{R}^n}[f(x) = \sum_{i=1}^{n} \frac{1}{n} \underbrace{\mathbb{E}_{\mathbf{S}_i\sim\mathcal{D}_i}[f_{\mathbf{S}_i}(x)]}_{:=f_i(x)}], \tag{2}$$

where $f_i(x)$'s are instantiated by different distributions $\mathcal{D}_i$ and $\mathbf{S}_i$ is sampled from $\mathcal{D}_i$. In the distributed setting, n in the ERM problem denotes number of processors/workers. Therefore, (2) is important from the distributed deep learning perspective as it captures data-parallelism (distributed over n processors, e.g., GPUs/CPUs/TPUs, etc.). One of the most popular algorithms for solving (1) is the stochastic gradient descent (SGD) [11]. For a given sequence of stepsize parameters, $\{\omega_k\}$ with $\omega_k > 0$, and the sequence of iterates, $\{x_k\}$, the updates of SGD take the form

$$x_{k+1} = x_k - \omega_k \nabla f_{\mathbf{S}_k}(x_k), \tag{3}$$

where $\nabla f_{\mathbf{S}_k}(x_k)$ is the stochastic gradient arising from the sample, $\mathbf{S}_k \sim \mathcal{D}$ drawn in each iteration and is an unbiased estimator of the gradient of f. A natural direction for solving (1) is to design a synchronized parallel update by using SGD [12]. If more than one processors are available, then they can work simultaneously and each one of them can calculate a stochastic gradient independent of the other processors. At the end, the

user can average the stochastic gradients from all processors to obtain the update as

$$x_{k+1} = x_k - \frac{\omega_k}{\tau} \sum_{i=1}^{\tau} \nabla f_{\mathbf{S}_{ki}}(x_k), \tag{4}$$

where $\nabla f_{\mathbf{S}_{ki}}(x_k)$ is the stochastic gradient arising from the independent sample, $\mathbf{S}_{ki} \sim \mathcal{D}$ from each of the τ processors and $\omega_k > 0$ is a uniform stepsize across k^{th} iteration. Note that, if $\tau = 1$, then the update scheme in (4) is (3).

In this paper, we compare the convergence rates of the asynchronous and synchronous algorithms in solving any arbitrary consistent linear systems by reformulating them into stochastic optimization problems. In the following section, we introduce the *stochastic reformulation of a linear system.*

1.1. Stochastic reformulation of a linear system

In [1], Richtárik and Takáč reformulated any arbitrary consistent linear system into a stochastic optimization problem (see details in [1, 13]; also [14]). Consider a linear system:

$$\mathbf{A}x = b, \tag{5}$$

where $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{A} \neq 0$. Let the set of solutions $\mathcal{L} := \{x : \mathbf{A}x = b\}$ be non-empty. That is, we consider a consistent linear system that has a solution which is not necessarily unique. Richtárik and Takáč, reformulated (5) into different stochastic problems and showed that for any arbitrary consistent linear system, the stochastic reformulations of (5) are exact. In other words, the set of the solutions of any of those equivalent stochastic formulations is exactly the same as $\mathcal{L}$ — which is formally defined as the *exactness* assumption; see Assumption 1 in Section 2. To motivate further, we will now introduce some technicalities. For a symmetric positive definite matrix $\mathbf{B}$, denote $\langle \cdot\ ,\ \cdot \rangle_{\mathbf{B}}$ as the $\mathbf{B}$-inner product and let $\|x\|_{\mathbf{B}} = \sqrt{x^\top \mathbf{B} x}$ be the norm induced by it. Therefore, by using the idea proposed in [1], one can define a stochastic function, $f_{\mathbf{S}}(x) := \frac{1}{2}\|\mathbf{A}x - b\|^2_{\mathbf{H}}$, where $\mathbf{H} = \mathbf{S}(\mathbf{S}^\top \mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S})^\dagger \mathbf{S}^\top$ is a random, symmetric, and positive definite matrix. Minimizing (1) with $f_{\mathbf{S}}(x) = \frac{1}{2}\|\mathbf{A}x - b\|^2_{\mathbf{H}}$

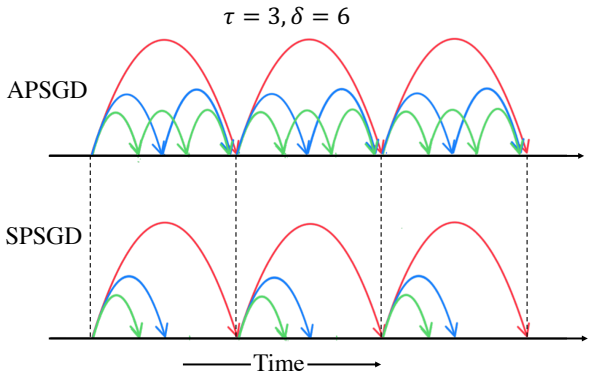


Figure 1: The above example demonstrates the setting to compare between our asynchronous parallel SGD (APSGD) and synchronous parallel SGD (SPSGD) proposed by Richtárik and Takáč in [1]. The depicted system consists of three processors ($\tau = 3$) with varying processing power, each denoted with a different color—slowest processor is in red. For APSGD, when the slowest processor can perform one update, the faster blue and green processors are able to perform three additional updates. With a maximum delay of $\delta = 6$, the faster processors wait for the red processor to perform its update before proceeding further. In contrast, SPSGD, at the same time, performs one update. After computing the gradient, the faster blue and green processors wait for gradient aggregation with the slower red processor.

solves (5) [1]. The *basic method* in [1] is the central algorithm whose iterates are SGD steps with a fixed stepsize parameter, $\omega > 0$ (see (3)) applied to solve (1) with $f_{\mathbf{S}}(x) = \frac{1}{2}\|\mathbf{A}x - b\|^2_{\mathbf{H}}$. Therefore, the iterates of the *basic method* are:

$$x_{k+1} = x_k - \omega \mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S}_k (\mathbf{S}_k^\top \mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S}_k)^\dagger \mathbf{S}_k^\top (\mathbf{A}x_k - b), \tag{6}$$

where $\mathbf{S}_k$ is sampled from the distribution, $\mathcal{D}$ in each iteration. The *synchronous parallel SGD* (SPSGD) algorithm in [1] is an extension of the *basic method* applied to a synchronous system with τ processors. The user averages the total yield from all processors at the end; see details in Section 2.1.

[1] In order to solve (5) via minimizing (1), one only needs local information of the stochastic function $f_{\mathbf{S}}(x)$, for example, the stochastic gradient $\nabla f_{\mathbf{S}}(x)$ without any explicit access to the function, its gradient, or the Hessian.

1.2. Contribution

In this paper, we solve (1) with $f_{\mathbf{S}}(x) = \frac{1}{2}\|\mathbf{A}x - b\|^2_{\mathbf{H}}$ [2] in a *distributed asynchronous* setup. We consider the abstraction with a central *master* server and τ independent *workers*, where the *master* obtains the gradients from the *workers* with a delay. We follow the framework of the *basic method* proposed by Richtárik and Takáč [1] to design our asynchronous SGD, and our algorithm is related to Hogwild! of Recht et al. [7] and the delayed proximal gradient algorithm of Feyzmahdavian et al. [15]. In a shared-memory model with τ independent *workers*, our algorithm updates the model, x that is accessible to all workers. Each *worker* can contribute an update to the model, although they can be of different processing speeds. Therefore, whenever a *worker* computes a stochastic gradient at x, it performs an SGD update, y at that point and communicates it to the *master*. The *master* eventually experiences a delay and updates the final model state, x by using a convex combination of its current update of the model, x available, and the SGD update, y that was communicated with a delay, and assigns the final model state to the available worker. We explain the algorithm formally in Section 3. The following are our main contributions in this paper:

- Inspired by SPSGD in [1], we design an asynchronous parallel SGD (APSGD) to solve a consistent linear system; see Algorithm 1 in Section 3.
- We propose a detailed convergence analysis of our APSGD algorithm in Section 4. The convergence analysis of our proposed APSGD is a standalone result that can handle any variable delay, δ_t bounded by a maximum delay, δ. See Theorem 4 and Theorem 5. Moreover, our convergence analysis does not require any stronger assumption such as sparsity as in Mania et al. [16] and Leblond et al. [17]. We refer to Table 1 for a quick overview of these results. For a detailed analysis, see Section 4.
- We compare the convergence rates and the iteration complexities of APSGD and SPSGD in Section 5. At this end, we assume time equivalence between one iter-

[2] stochastic reformulation of the linear system (5)

Algorithm	Quantity	Case	ω	θ	τ	Complexity	Reference
APSGD	$\mathbb{E}[\|x_t - x_\star\|^2_{\mathbf{B}}]$	$\omega^\star \leq 2$	1	θ_1	τ	$\frac{\xi_a(1,\delta)}{\lambda^+_{\min}}$	This paper
			1	θ_1	∞	$\frac{3}{4\lambda^+_{\min}}$	Theorem 4
			$\omega^\star$	$\theta_{\omega^\star}$	τ	$\frac{\xi_a(\omega^\star,\delta)}{\lambda^+_{\min}}$	
			$\omega^\star$	$\theta_{\omega^\star}$	∞	$\frac{3\lambda^+_{\min}+\lambda_{\max}}{4\lambda^+_{\min}}$	
APSGD	$\mathbb{E}[\|x_t - x_\star\|^2_{\mathbf{B}}]$	$\omega^\star \geq 2$	2	θ_2	τ	$\frac{\xi_a(2,\delta)}{\lambda^+_{\min}}$	This paper
			2	θ_2	∞	$\frac{1+2\lambda^+_{\min}}{4\lambda^+_{\min}}$	Theorem 5
SPSGD	$\mathbb{E}[\|x_t - x_\star\|^2_{\mathbf{B}}]$	$\omega \in (0, 2/\xi_s(\tau))$	1	-	τ	$\frac{1}{(2-\xi_s(\tau))\lambda^+_{\min}}$	[1]
			$1/\xi_\tau$	-	τ	$\frac{\xi_s(\tau)}{\lambda^+_{\min}}$	[1]
			$1/\lambda_{\max}$	-	∞	$\frac{\lambda_{\max}}{\lambda^+_{\min}}$	[1]

Table 1: Iteration complexities of APSGD and SPSGD or parallel basic method. Here *"Complexity"* denotes the number of iterations required to make *"Quantity"* smaller than an error tolerance $\epsilon > 0$, where we suppress a $\log(1/\epsilon)$ factor in all expressions in the *"Complexity"* column. All the results are for *maximum delay,* $\delta = c\tau$ for some $c \geq 1$. Additionally, the complexity results of APSGD are normalized by δ—inline with our setting in Section 5 (see Figure 1). For APSGD, we have, $\xi_a(1,\delta) := \frac{3}{4} + \frac{1+\sqrt{1+2\delta(1-\lambda^+_{\min})}}{4\delta}$, $\xi_a(\omega^\star,\delta) := \frac{3\lambda^+_{\min}+\lambda_{\max}}{4} + \frac{\sqrt{1+\delta(2-k)}+2(\delta+1+\delta(1-k)\lambda^+_{\min})}{2\delta\sqrt{1+\delta(2-k)}}$ and $\xi_a(2,\delta) := \frac{1}{4} + \frac{\lambda^+_{\min}}{2} + \frac{1+\sqrt{\delta+1}}{2\delta}$, where $k = \lambda^+_{\min} + \lambda_{\max}$ and $c \geq 1$. We also have, $\theta_1 := \frac{\sqrt{1+2\delta(1-\lambda^+_{\min})}-1}{\delta(1-\lambda^+_{\min})}$, $\theta_2 := \frac{\sqrt{\delta+1}-1}{\delta}$, and $\theta_{\omega^\star} := \frac{k(\sqrt{1+\delta(2-k)}-1)}{\delta(2-k)}$. For parallel SGD, we have $\xi_s(\tau) := \frac{1}{\tau} + \left(1 - \frac{1}{\tau}\right)\lambda_{\max}$.

ation of APSGD and δ iterations of SPSGD; see Figure 1. We find that asymptotically as the number of processors, τ approaches to ∞, APSGD has a better iteration complexity than SPSGD for certain linear systems—These are consequences of Theorem 4 and Theorem 5. Moreover, for such linear systems, we also compute the minimum number of processors such that APSGD has a better iteration complexity than the synchronous method, and find that this number can be as low as 2 for some (highly ill-conditioned) problems; see Table 2 and 3.

1.3. Distributed optimization—Related work

Parallel and distributed stochastic optimization algorithms broadly fall into two classes: (*i*) centralized (that follows a *master-worker architecture*) and (*ii*) decentralized (e.g., `Allreduce` communication collective in the message passing interface (MPI) [18]), based on their communication protocol. This paper follows the centralized setup where a central *master* node coordinates with all the *worker* nodes. Depending on the update rule, centralized algorithms can be further categorized into (*i*) synchronous and (*ii*) asynchronous algorithms. We will quote a few representatives of each category for completeness.

Among the first works, Zinkevich et al. [12] proposed and analyzed synchronous parallel SGD. Richtárik and Takáč in [19] showed by parallelizing, randomized block coordinate descent methods can be accelerated. Yang [20] proposed a distributed stochastic dual coordinate ascent algorithm in a *star-shaped distributed network* and analyzed the trade-off between computation and communication. Similarly, Jaggi et al. [21] proposed Communication-efficient distributed dual Coordinate Ascent or COCOA (we refer to the references in [21, 22] for distributed primal-dual methods; additionally, see [23] for application to distributed model predictive control). Fercoq and Richtárik in [24] proposed Accelerated Parallel PROXimal method (APPROX)—a unison of three ideas—acceleration, parallelization, and proximal method. In [25], Richtárik and Takáč proposed and analyzed a *hybrid* coordinate descent method known as HYDRA that partitions the coordinates over the nodes, independently from the other nodes, and applies updates to the selected coordinates in parallel (see [26, 27] and HYDRA-2 [28] for more insights). Shamir et al. [29] proposed a distributed approximate Newton-type method or DANE in a similar line of work.

Synchronous data-parallelism [30] is widely adopted in practice, e.g., training large deep neural networks (DNNs) using mainstream deep learning toolkits. In a constant speed network, reduced data volume translates to a faster training [6, 31]—based on this, gradient compression techniques (see [32, 6, 33, 34], and references therein) are used to train large DNN models in a distributed, synchronous setup. However, these techniques are orthogonal to our discussion, and we refer interested readers to [6] for a comprehensive survey and quantitative evaluation of them.

Although synchronous stochastic algorithms are well explored regarding their convergence rates, acceleration, and parallelization [35, 36], they may suffer from *memory locking*. The computing nodes need to wait for the update from the slowest node. Hogwild! by Recht et al. [7] is the first asynchronous SGD that does not use the *memory locking*, and as a result, the computing nodes can modify the parameters at the same time. De Sa et al. [37] proposed Buckwild! which is a low-precision asynchronous SGD. Additionally, [37] analyzed Hogwild! type algorithms with relaxed assumptions; also see [38]. Noel et al. [39] proposed Dogwild!—a distributed Hogwild! algorithm for CPU and GPU. Chaturapruek et al. [40] showed that for convex problems, under similar conditions as regular SGD, asynchronous SGD achieves a similar asymptotic convergence rate. The perturbed iterate analysis of Mania et al. [16] and Leblond et al. [17] has emerged as a promising technique for analyzing asynchronous algorithms, with applications to distributed optimization with compressed gradients and local SGD [41]. While the initial perturbed iterate analyses depend on a gradient sparsity assumption [16, 17], the latter overcomes this [41].

Asynchronous parallel SGD algorithms are also highly deployed in practice. Recently, Lian et al. [42] studied two asynchronous parallel algorithms—one is over a computer network, and another is on a shared memory system. They claimed that in their setting, proposed asynchronous parallel algorithms could achieve a linear speedup if the number of workers is bounded by the square root of the total number of iterations. In 2017, Zheng et al. [43] proposed an algorithm called delay compensated asynchronous SGD (DC-ASGD) for training deep neural networks, which compensates for the *delayed* gradient updates by using approximate Hessian information. With experimental validity on deep neural networks, [43] claimed that DC-ASGD outperforms both synchronous SGD and asynchronous SGD, and nearly approaches the performance of sequential SGD.

Among the others, asynchronous algorithms by Aytekin et al. [44], distributed SDCA by Ma et al. [45], distributed SVRG by Lee et al., [46] and Zhao and Li [47], asynchronous parallel SAGA by Leblond et al. [17], proximal asynchronous SAGA by Pedregosa et al. [48], decentralized asynchronous parallel SGD by Lian et al. [49] are to name a few. We refer to [50, 51] for an in-depth understanding.

Notation. We provide a table of the most frequently used notation in this paper; see 6. Here we include some basic notations. We write the matrices in bold uppercase letters and denote vectors and scalars by lowercase letters. We define the range space and null space of a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ as $\mathrm{Im}(\mathbf{A}) := \{y \in \mathbb{R}^m : y = \mathbf{A}x\}$, and $N(\mathbf{A}) := \{x \in \mathbb{R}^n : \mathbf{A}x = 0\}$, respectively. We further define the Euclidean inner product as $\langle\cdot,\cdot\rangle$, and for a symmetric positive (semi)-definite matrix $\mathbf{B}$, we denote $\langle\cdot,\cdot\rangle_{\mathbf{B}}$ as the $\mathbf{B}$-inner product, and define $\|x\|_{\mathbf{B}} = \sqrt{x^\top \mathbf{B}x}$ as the (semi)-norm induced by it. By $\|x\|$ and $\|x\|_\infty$, we denote the ℓ_2 and ℓ_∞ norm of a vector, x, respectively.

Organization. In Section 2, we review some key results related to the stochastic reformulation of linear systems and describe SPSGD by Richtárik and Takáč, in [1]. Next in Section 3, we propose APSGD and present its convergence analysis in Section 4. Finally, we compare the convergence rates of APSGD with SPSGD in Section 5.

2. Stochastic reformulation of a linear system: A few key results

In addition to Section 1.1, we quote some results from [1, 13] without their proofs. These results are used to establish our main results. Let

$$\mathbf{Z} = \mathbf{Z}_{\mathbf{S}} := \mathbf{A}^\top \mathbf{S}(\mathbf{S}^\top \mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S})^\dagger \mathbf{S}^\top \mathbf{A},$$

and $\mathbb{E}[\mathbf{Z}] := \mathbb{E}_{\mathbf{S}\sim\mathcal{D}}[\mathbf{Z}]$ be such that $\mathcal{D}$ is a user-defined distribution and $\mathbf{S}$ be a random matrix drawn from $\mathcal{D}$. Let $\mathbf{B}$ be a $n \times n$ symmetric positive definite matrix. Define $\mathbf{W} := \mathbf{B}^{-\frac{1}{2}}\mathbb{E}[\mathbf{Z}]\mathbf{B}^{-\frac{1}{2}}$, and let $\mathbf{W} = \mathbf{U}\Lambda\mathbf{U}^\top$ be a eigenvalue decomposition of $\mathbf{W}$, where $\mathbf{U}^\top\mathbf{U} = \mathbf{U}\mathbf{U}^\top = \mathbf{I}$, and $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ is a diagonal matrix with eigenvalues, $0 \le \lambda_i \le 1$ arranged in a non-increasing order. We denote $\lambda_{\max} = \lambda_1$ as the largest, and $\lambda_{\min}^+$ as the smallest non-zero eigenvalue of $\mathbf{W}$. We start by defining the *exactness* assumption.

Assumption 1 (Assumption 2 in [1])**.** *(Exactness) Let* $\mathcal{X} = \arg\min_{x\in\mathbb{R}^n} f(x) = \{x : f(x) = 0\} = \{x : \nabla f(x) = 0\}$. *We assume* $\mathcal{X} = \mathcal{L}$.

By $x_\star = \Pi_{\mathcal{L}}^{\mathbf{B}}(x_0)$ we denote $x_\star$ to be the projection of the initial iterate x_0 onto the set $\mathcal{L}$ in $\mathbf{B}$-norm and quote the following results.

Remark 1. *For $x_\star \in \mathcal{L}$, the gradient and Hessian of f are $\nabla f(x) = \mathbf{B}^{-1}\mathbb{E}[\mathbf{Z}](x-x_\star)$ and $\nabla^2 f(x) = \mathbf{B}^{-1}\mathbb{E}[\mathbf{Z}]$, respectively.*

Lemma 1 (Lemma 4.7 in [1])**.** *For all $x \in \mathbb{R}^n$, $x_\star \in \mathcal{L}$ and for a given $\mathbf{S}$ we have*

$$\|x-x_\star-\omega\nabla f_{\mathbf{S}}(x)\|_{\mathbf{B}}^2 = \|(\mathbf{I}-\omega\mathbf{B}^{-1}\mathbf{Z})(x-x_\star)\|_{\mathbf{B}}^2 = \|x-x_\star\|_{\mathbf{B}}^2 - 2\omega(2-\omega)f_{\mathbf{S}}(x). \tag{7}$$

Lemma 2 (Lemma 4.2 in [1])**.** *For all $x \in \mathbb{R}^n$ and $x_\star = \Pi_{\mathcal{L}}^{\mathbf{B}}(x)$ we have $\frac{\lambda_{\min}^+}{2}\|x - x_\star\|_{\mathbf{B}}^2 \le f(x) \le \frac{\lambda_{\max}}{2}\|x - x_\star\|_{\mathbf{B}}^2$.*

Lemma 3 (Lemma 4.5 in [1])**.** *Consider any $x \in \mathbb{R}^n$ and $x_\star = \Pi_{\mathcal{L}}^{\mathbf{B}}(x)$. If $\lambda_i = 0$ then we have $u_i^\top \mathbf{B}^{1/2}(x - x_\star) = 0$.*

2.1. Synchronous parallel SGD (SPSGD) [1]

In this section, we describe the *parallel basic method*, also known as synchronous parallel SGD (SPSGD) by Richtárik and Takáč. Let there be τ processors working independently. Starting from a given iterate, x_k, SPSGD performs one step of the *basic method* independently on each of the τ processors, and finally averages the results. This leads to the update rule of SPSGD as follows:

$$x_{k+1} = \frac{1}{\tau}\sum_{i=1}^{\tau} z_{k+1,i}, \tag{8}$$

where $z_{k+1,i} = x_k - \omega\mathbf{B}^{-1}\mathbf{A}^\top\mathbf{S}_{ki}(\mathbf{S}_{ki}^\top\mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top\mathbf{S}_{ki})^\dagger\mathbf{S}_{ki}^\top(\mathbf{A}x_k - b)$. Note that, in each iterate an independent sample, $\mathbf{S}_{ki} \sim \mathcal{D}$ drawn for each of the τ processors and $\omega > 0$ is a fixed stepsize parameter. SPSGD enjoys a linear convergence rate under the *exactness* assumption; see Theorem 1.

Theorem 1 (Convergence of SPSGD [1])**.** *Let the exactness assumption hold and $x_\star = \Pi_{\mathcal{L}}^{\mathbf{B}}(x_0)$. Let $\{x_k\}_{k\ge 0}$ be the sequence of random iterates produced by the parallel method (see* (8)*) where the stepsize $\omega \in (0, 2/\xi_s(\tau))$, such that $\xi_s(\tau) = \frac{1}{\tau} + (1 - \frac{1}{\tau})\lambda_{\max}$. Then*

$$\mathbb{E}[\|x_{k+1} - x_\star\|_{\mathbf{B}}^2] \le \beta_s(\omega,\tau)^k \frac{\lambda_{\max}}{2}\|x_0 - x_\star\|_{\mathbf{B}}^2, \tag{9}$$

where

$$\beta_s(\omega,\tau) = 1 - \omega(2 - \omega\xi_s(\tau))\lambda_{\min}^+. \tag{10}$$

Richtárik and Takáč [1] further showed that the convergence rate, $\beta_s(\omega,\tau)$ is minimized for $\omega(\tau) = 1/\xi_s(\tau)$ and the optimal $\beta_{s_{\text{opt}}}(\omega(\tau),\tau)$ is

$$\beta_{s_{\text{opt}}}(\omega(\tau),\tau) = 1 - \frac{\lambda^+_{\min}}{\frac{1}{\tau} + \left(1-\frac{1}{\tau}\right)\lambda_{\max}}. \tag{11}$$

With this optimal rate, $\beta_{s_{\text{opt}}}(\omega(\tau),\tau)$, let the corresponding optimal iteration complexity of SPSGD is $\chi_{s_{opt}}(\tau)$. If

$$k \geq \chi_{s_{opt}}(\tau)\log\frac{1}{\epsilon}, \quad \text{then} \quad \mathbb{E}\left[\|x_k - x_\star\|^2_{\mathbf{B}}\right] \leq \epsilon\|x_0 - x_\star\|^2_{\mathbf{B}}, \tag{12}$$

where $\chi_{s_{opt}}(\tau) = \frac{\frac{1}{\tau}+\left(1-\frac{1}{\tau}\right)\lambda_{\max}}{\lambda^+_{\min}}$.

Remark 2. *The best strong convergence rate for the basic method (SPSGD with $\tau = 1$) is achieved when the stepsize parameter, $\omega = 1$. We define the strong convergence in Section 4.2.*

3. Asynchronous parallel SGD (APSGD)

Let there be τ independent processors or *workers* and a central server or the *master* (see Section 1) and we perform the iterative updates in an *asynchronous* manner. Whenever a *worker* computes a stochastic gradient at a given point, it performs an SGD step at that point and communicates the update to the *master*. After that, the *master* generates a new update by using a convex combination of its current update, and the update reported to it by the *worker*, and communicates back the resulting update to the *worker* to perform the next SGD step, and this process continues. Regardless of their processing speeds, whenever a *worker* communicates its latest update to the *master*, the master makes a convex combination of the latest iterate with it and assigns it to the worker to perform an SGD step. Therefore, no *worker* stays idle. Each worker performs and communicates the task they are assigned to the *master*, albeit in an asynchronous way. Feyzmahdavian et al. introduced this protocol in [15], and our proposed update rule is inspired by it.

Let t denote the iteration count with respect to the *master*'s frame. Let δ_t be a function of t such that $\delta_t \geq 0$ and represents the *delay* between the iterates of the *master* and the *workers*, which varies in each iteration. Let δ be the maximum delay

throughout the execution of the algorithm. Let $\theta \in [0,1]$ be a damping factor. Choose initial approximate, $x_0 \in \mathbb{R}^n$, and consider an update rule defined for $t \geq 1$ as:

$$y_t = x_{t-\delta_t} - \omega \nabla f_{\mathbf{S}_{t-\delta_t}}(x_{t-\delta_t}), \tag{13}$$

$$x_{t+1} = (1-\theta)x_t + \theta y_t. \tag{14}$$

Recall from Remark 1 that $\nabla f_{\mathbf{S}}(x) = \mathbf{B}^{-1}\mathbf{Z}(x - x_\star)$, where $x_\star$ is any solution of the system, $\mathbf{A}x = b$. Therefore, we can rewrite the update rule as:

$$\boxed{x_{t+1} - x_\star = (1-\theta)(x_t - x_\star) + \theta(\mathbf{I} - \omega \mathbf{B}^{-1}\mathbf{Z}_{t-\delta_t})(x_{t-\delta_t} - x_\star),} \tag{15}$$

where we denote $\mathbf{Z}_t := \mathbf{Z}_{\mathbf{S}_t}$, and $\mathbf{S}_t$ is sampled independently from $\mathcal{D}$. We refer to Algorithm 1 for the numerical procedure.

Algorithm 1: Asynchronous Parallel SGD (APSGD) in master-worker architecture (*time t in master's frame*)

System : 1 Master and τ workers, $k = 1, ..., \tau$;

repeat

  Master :

  Receive y_t from a worker k;

  Update current iterate x_t by using equation (14);

  Send updated iterate x_{t+1} to worker k;

  Workers:

  Receive iterate x_t from master;

  Compute $y_{t+\delta_t}$ via equation (13);

  Send $y_{t+\delta_t}$ to the master;

until *convergence*;

4. Convergence analysis

For APSGD, we study the convergence of the quantity, $\mathbb{E}\left[\|x_k - x_\star\|_{\mathbf{B}}^2\right]$. Richtárik and Takáč [1] refer this as *strong convergence*. We omit convergence of the quantity, $\|\mathbb{E}\left[x_k - x_\star\right]\|_{\mathbf{B}}^2$, defined as *weak convergence* in [1], as this is directly implied from the strong convergence result. Nevertheless, in practice, only strong convergence matters.

4.1. Overview of analysis

Let $p_t = \mathbb{E}\left[\|x_t - x_\star\|_{\mathbf{B}}^2\right]$. In the analysis, we handle the following recurrence relation

$$p_{t+1} \leq a\, p_t + b\, p_{t-\delta_t}, \quad \text{for all } t \geq 0, \tag{16}$$

where $a \geq 0$, $b \geq 0$, and $a + b < 1$; see Theorem 2 for this result. We present the recurrence relation (16) in the following form:

$$u_{t+1} \leq \mathbf{A}_{\delta_t} u_t, \tag{17}$$

where $u_t = (p_t \;\; p_{t-1} \cdots p_{t-\delta})^\top$, and $\mathbf{A}_{\delta_t} \in \mathbb{R}^{(\delta+1)\times(\delta+1)}$ is a state transition matrix, such that $\mathbf{A}_{\delta_t} = \begin{pmatrix} [a \;\; \mathbf{0}_{\delta_t-1}^\top] & b & [0_{\delta-\delta_t}^\top] \\ I_\delta & & \mathbf{0}_\delta \end{pmatrix}$. Note that, $\mathbf{0}_n$ is a vector in $\mathbb{R}^n$ with all zeros, and $I_n \in \mathbb{R}^{n\times n}$ is the $n \times n$ identity matrix.

The characteristic equation of the matrix, $\mathbf{A}_{\delta_t}$ is:

$$\gamma^{\delta_t+1} - a\gamma^{\delta_t} - b = 0. \tag{18}$$

This is true for any time-varying delay, δ_t. Moreover, for the special case, when $\delta_t = \delta$, it will reduce to,

$$\gamma^{\delta+1} - a\gamma^{\delta} - b = 0. \tag{19}$$

The spectral radius of $\mathbf{A}_{\delta_t}$ is the magnitude of the largest root of the characteristic polynomial in (18). We first show that the largest magnitude root is the only positive root of (18); see our discussion in Section 4.3.1. We then show in Lemma 5 that the rate of convergence of (16) is given by the spectral radius, ρ, of the matrix, $\mathbf{A}_\delta$, that is,

$$p_{t+1} \leq \rho \max(p_t, \rho p_{t-1}, \ldots, \rho^\delta p_{t-\delta}). \tag{20}$$

Since ρ cannot be explicitly derived, we determine a $\beta \in [0, 1)$ as an algebraic upper bound of this positive root; see Lemma 6 in Section 4.4.1. Thus, if the maximum delay for APSGD is δ, then by unrolling the recurrence in (20), one can have

$$p_{t+1} \leq \beta^{t-\delta+1} \max(p_\delta, \beta p_{\delta-1}, \ldots, \beta^\delta p_0). \tag{21}$$

Finally, in Section 4.5 we analyze the rate of convergence and the optimal iteration complexity of APSGD under varying stepsize, ω, damping parameter, θ, and maximum delay, δ and record our findings in Theorem 4 and Theorem 5.

4.2. Preliminaries

We start with the following Lemma.

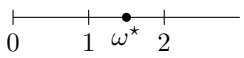
Lemma 4. *Assume that* $x_0 \in \mathrm{Im}(\mathbf{B}^{-1}\mathbf{A}^\top)$. *Then* $x_t \in \mathrm{Im}(\mathbf{B}^{-1}\mathbf{A}^\top)$ *for all* t. *It follows that* $\Pi_{\mathcal{L}}^{\mathbf{B}}(x_t) = \Pi_{\mathcal{L}}^{\mathbf{B}}(x_0)$ *for all* t.

Proof. The first part follows from (15). The second part follows from the observation that $\mathrm{Im}(\mathbf{B}^{-1}\mathbf{A}^\top)$ is the $\mathbf{B}$-orthogonal complement of the nullspace of $\mathbf{A}$. □

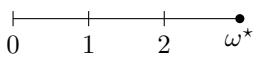
4.2.1. Two scenarios based on the relative position of $\omega^\star$ and 2

Let $\omega^\star = \frac{2}{(\lambda_{\min}^+ + \lambda_{\max})}$. According to the problem, there are two scenarios based on the relative position of $\omega^\star$ with respect to 2 as follows:

Case 1: $\omega^\star \leq 2$ that is $\lambda_{\min}^+ + \lambda_{\max} \geq 1$.



Case 2: $\omega^\star \geq 2$ that is $\lambda_{\min}^+ + \lambda_{\max} \leq 1$.



Definition 1. *Define* $\alpha(\omega) := \max_{i:\lambda_i>0} |1 - \omega\lambda_i|$. *It is easy to see that*

$$\alpha(\omega) = \begin{cases} 1 - \omega\lambda_{\min}^+, & \text{if } 0 \leq \omega \leq \omega_\star \\ \omega\lambda_{\max} - 1, & \text{if } \omega \geq \omega_\star. \end{cases}$$

4.3. Recurrence relation

The following theorem establishes a recurrence relation required to analyze the strong convergence of APSGD Algorithm (see Algorithm 1).

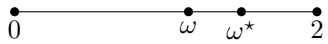
Theorem 2 (Recurrence)**.** *Assume exactness. Assume that* $x_0, \ldots, x_\delta \in \mathrm{Im}(\mathbf{B}^{-1}\mathbf{A}^\top)$ *and let* $x_\star = \Pi_{\mathcal{L}}^{\mathbf{B}}(x_0)$. *Let* $\{x_t\}$ *be the sequence of random iterates produced by the asynchronous method (via* (15)*) with delay* $\delta_t \geq 0$, *stepsize* $\omega \geq 0$, *and damping parameter* $\theta \in [0, 1]$. *Let* $r_t = \mathbf{B}^{1/2}(x_t - x_\star)$ *and* $\alpha(\omega) := \max_{i:\lambda_i>0} |1 - \omega\lambda_i|$. *Then*

$$\boxed{\mathbb{E}[\|r_{t+1}\|^2] \leq K_1(\theta,\omega)\mathbb{E}[\|r_t\|^2] + K_2(\theta,\omega)\mathbb{E}[\|r_{t-\delta_t}\|^2],} \tag{22}$$

with the following estimates of $K_1(\theta,\omega)$ and $K_2(\theta,\omega)$:

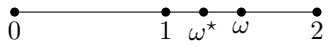
Case 1:

(i) For $\omega \le \omega^\star \le 2$ and $\alpha(\omega) = 1 - \omega\lambda^+_{\min}$:



$$K_1(\theta,\omega) := (1-\theta)(1-\theta+\theta\alpha(\omega)), \qquad K_2(\theta,\omega) := \theta(\theta(1-\omega(2-\omega)\lambda^+_{\min})+(1-\theta)\alpha(\omega)). \tag{23}$$

(ii) For $\omega^\star \le \omega \le 2$ and $\alpha(\omega) = \omega\lambda_{\max} - 1$:



$$K_1(\theta,\omega) := (1-\theta)(1-\theta+\theta\alpha(\omega)), \qquad K_2(\theta,\omega) := \theta(\theta(1-\omega(2-\omega)\lambda^+_{\min})+(1-\theta)\alpha(\omega)).$$

(iii) For $\omega \ge 2$ and $\alpha(\omega) = \omega\lambda_{\max} - 1$:

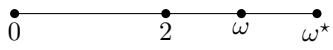
$$K_1(\theta,\omega) := (1-\theta)(1-\theta+\theta\alpha(\omega)), \qquad K_2(\theta,\omega) := \theta(\theta(1-\omega(2-\omega)\lambda_{\max})+(1-\theta)\alpha(\omega)).$$

Case 2:

(i) For $\omega \le 2 \le \omega^\star$ and $\alpha(\omega) = 1 - \omega\lambda^+_{\min}$:

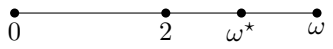
$$K_1(\theta,\omega) := (1-\theta)(1-\theta+\theta\alpha(\omega)), \qquad K_2(\theta,\omega) := \theta(\theta(1-\omega(2-\omega)\lambda^+_{\min})+(1-\theta)\alpha(\omega)). \tag{24}$$

(ii) For $2 \leq \omega \leq \omega^\star$ *and* $\alpha(\omega) = 1 - \omega\lambda^+_{\min}$*:*



$$K_1(\theta,\omega) := (1-\theta)(1-\theta+\theta\alpha(\omega)), \qquad K_2(\theta,\omega) := \theta(\theta(1-\omega(2-\omega)\lambda_{\max})+(1-\theta)\alpha(\omega)). \tag{25}$$

(iii) For $\omega \geq \omega^\star$ *and* $\alpha(\omega) = \omega\lambda_{\max} - 1$*:*



$$K_1(\theta,\omega) := (1-\theta)(1-\theta+\theta\alpha(\omega)), \qquad K_2(\theta,\omega) := \theta(\theta(1-\omega(2-\omega)\lambda_{\max})+(1-\theta)\alpha(\omega)).$$

Proof. We hereby provide the proof for $\omega \in [0,2]$. The proof for $\omega \geq 2$ follows similarly by using Lemma 2 in (26).

By taking norms on both sides of (15) and then applying Lemma 1, we get

$$\begin{aligned}
\|x_{t+1} - x_\star\|^2_{\mathbf{B}} \overset{(15)}{=} \quad & (1-\theta)^2\|x_t - x_\star\|^2_{\mathbf{B}} + \theta^2\|(\mathbf{I} - \omega\mathbf{B}^{-1}\mathbf{Z}_{t-\delta_t})(x_{t-\delta_t} - x_\star)\|^2_{\mathbf{B}} \\
& +2(1-\theta)\theta\langle x_t - x_\star, (\mathbf{I} - \omega\mathbf{B}^{-1}\mathbf{Z}_{t-\delta_t})(x_{t-\delta_t} - x_\star)\rangle_{\mathbf{B}} \\
\overset{\text{Lemma 1}}{=} \quad & (1-\theta)^2\|x_t - x_\star\|^2_{\mathbf{B}} + \theta^2\|x_{t-\delta_t} - x_\star\|^2_{\mathbf{B}} - 2\omega(2-\omega)\theta^2 f_{\mathbf{S}_{t-\delta_t}}(x_{t-\delta_t}) \\
& +2(1-\theta)\theta\langle x_t - x_\star, (\mathbf{I} - \omega\mathbf{B}^{-1}\mathbf{Z}_{t-\delta_t})(x_{t-\delta_t} - x_\star)\rangle_{\mathbf{B}}.
\end{aligned}$$

The above identity can be written as

$$\begin{aligned}
\|r_{t+1}\|^2 = (1-\theta)^2\|r_t\|^2 + \theta^2\|r_{t-\delta_t}\|^2 - 2\omega(2-\omega)\theta^2 f_{\mathbf{S}_{t-\delta_t}}(x_{t-\delta_t}) \\
+2(1-\theta)\theta\langle r_t, (\mathbf{I} - \omega\mathbf{B}^{-1/2}\mathbf{Z}_{t-\delta_t}\mathbf{B}^{-1/2})r_{t-\delta_t}\rangle.
\end{aligned}$$

Conditioning on $x_t, \ldots, x_0$, the only free random variable is $\mathbf{S}_{t-\delta_t}$. Therefore, in view of Lemma 2 and using the eigenvalue decomposition $\mathbf{B}^{-1/2}\mathbb{E}[\mathbf{Z}]\mathbf{B}^{-1/2} = \mathbf{U}\Lambda\mathbf{U}^\top$, we get the following bound on $C := \mathbb{E}\left[\|r_{t+1}\|^2 \mid x_t, \ldots, x_0\right]$:

$$\begin{aligned}
C \quad = \quad & (1-\theta)^2\|r_t\|^2 + \theta^2\|r_{t-\delta_t}\|^2 - 2\omega(2-\omega)\theta^2 f(x_{t-\delta_t}) + 2(1-\theta)\theta r_t^\top(\mathbf{I} - \omega\mathbf{B}^{-1/2}\mathbb{E}[\mathbf{Z}]\mathbf{B}^{-1/2})r_{t-\delta_t} \\
\overset{\text{Lemma 2}}{\leq} \quad & (1-\theta)^2\|r_t\|^2 + \theta^2(1-\omega(2-\omega)\lambda^+_{\min})\|r_{t-\delta_t}\|^2 + 2(1-\theta)\theta \underbrace{r_t^\top(\mathbf{I} - \omega\mathbf{U}\Lambda\mathbf{U}^\top)r_{t-\delta_t}}_{D}.
\end{aligned} \tag{26}$$

Further, we have

$$
\begin{aligned}
D &= r_t^\top (\mathbf{I} - \omega \mathbf{U}\Lambda\mathbf{U}^\top) r_{t-\delta_t} \\
&= (\mathbf{U}^\top r_t)^\top (\mathbf{I} - \omega\Lambda)\mathbf{U}^\top r_{t-\delta_t} \\
&= \sum_i (1-\omega\lambda_i) u_i^\top r_t u_i^\top r_{t-\delta_t} \\
&\overset{\text{Lemmas 3 and 4}}{=} \sum_{i:\lambda_i \neq 0} (1-\omega\lambda_i) u_i^\top r_t u_i^\top r_{t-\delta_t} \\
&\leq \sum_{i:\lambda_i \neq 0} |1-\omega\lambda_i| |u_i^\top r_t u_i^\top r_{t-\delta_t}| \\
&\leq \alpha(\omega) \sum_{i:\lambda_i \neq 0} |u_i^\top r_t u_i^\top r_{t-\delta_t}| \\
&\overset{\text{(Cauchy-Schwarz)}}{\leq} \alpha(\omega) \|r_t\| \|r_{t-\delta_t}\| \\
&\overset{\text{(AM-GM)}}{\leq} \frac{\alpha(\omega)}{2} \left(\|r_t\|^2 + \|r_{t-\delta_t}\|^2 \right).
\end{aligned}
$$

Combining the bounds on C and D, we get

$$
\begin{aligned}
C &\leq (1-\theta)^2 \|r_t\|^2 + \theta^2 (1 - \omega(2-\omega)\lambda_{\min}^+) \|r_{t-\delta_t}\|^2 + (1-\theta)\theta\alpha(\omega) \left(\|r_t\|^2 + \|r_{t-\delta_t}\|^2 \right) \\
&= \left[(1-\theta)^2 + (1-\theta)\theta\alpha(\omega) \right] \|r_t\|^2 \\
&\quad + \left[\theta^2 (1 - \omega(2-\omega)\lambda_{\min}^+) + (1-\theta)\theta\alpha(\omega) \right] \|r_{t-\delta_t}\|^2. \qquad (27)
\end{aligned}
$$

The final result is obtained after we take full expectation and apply the tower property of expectation. □

Remark 3. *To analyze the convergence of the recurrence relation in* (22)*, we can replace the inequality in* (22) *with equality and write*

$$
q_{t+1} = K_1(\theta,\omega) q_t + K_2(\theta,\omega) q_{t-\delta_t}, \qquad (28)
$$

where we initialize the process by setting $q_t = \mathbb{E}\left[\|x_t - x_\star\|_{\mathbf{B}}^2 \right]$ *for* $t \in \{0, \ldots, \tau\}$*. It can be proved by using induction that* $\mathbb{E}\left[\|x_t - x_\star\|_{\mathbf{B}}^2 \right] \leq q_t$ *for all* t*, and hence the convergence rate of* $\mathbb{E}\left[\|x_t - x_\star\|_{\mathbf{B}}^2 \right]$ *will not be slower than* q_t*.*

Note that, (28) follows recurrence (16) with $a = K_1(\theta,\omega)$ and $b = K_2(\theta,\omega)$. Therefore, we represent (28) in the form of (17). In the following, we examine the characteristic polynomial of the state transition matrix.

4.3.1. Characteristic polynomial

From (18), the characteristic polynomial of the state transition matrix, $\mathbf{A}_{\delta_t}$, of recurrence (28) is

$$p_\delta(\gamma) := \gamma^{\delta_t+1} - K_1(\theta,\omega)\gamma^{\delta_t} - K_2(\theta,\omega). \tag{29}$$

For convenience, we denote $K_1 = K_1(\theta,\omega)$ and $K_2 = K_2(\theta,\omega)$, from now on. The spectral radius is the largest magnitude root of (29). We quote the following result on the bounds of the root of a polynomial

Theorem 3. *(Cauchy [52]) Let $f(x) = x^n - \sum_{i=1}^{n} b_i x^{n-i}$ be a polynomial, where all the coefficients, b_i's are non-negative and at least one of them is nonzero. The polynomial $f(x)$ has a unique (simple) positive root p and the absolute values of the other roots do not exceed p.*

By using Theorem 3 we can conclude that the unique positive root of (29) is the spectral radius of (29).

Remark 4. *The spectral radius is smaller than 1 only when $K_1 + K_2 < 1$, as $p_\delta(1)$ is positive for the root to be smaller than 1. If $K_1 + K_2 \geq 1$, then either the spectral radius is not a good bound for the one-step rate of convergence or the algorithm does not converge.*

4.4. Convergence of the recurrence relation

The following lemma states that the rate of convergence is determined by calculating the spectral radius of this state transition matrix.

Lemma 5. *The rate of convergence of the recurrence in* (16) *is given by the spectral radius, ρ, of the state transition matrix, $\mathbf{A}_\delta$.*

Proof. Let $D = \text{diag}(1, \rho, \rho^2, \ldots, \rho^\delta) \in \mathbb{R}^{(\delta+1)\times(\delta+1)}$ be a diagonal matrix. Then,

$$D\mathbf{A}_{\delta_t}D^{-1} = \begin{pmatrix} [a \;\; \mathbf{0}_{\delta_t-1}^\top] & b\rho^{-\delta_t} & [0_{\delta-\delta_t}^\top] \\ \rho I_\delta & & \mathbf{0}_\delta \end{pmatrix}.$$

Notice the first row of $D\mathbf{A}_{\delta_t}D^{-1}$ has entries a and $b\rho^{-\delta_t}$, and the characteristic equation of $\mathbf{A}_\delta$, that is, equation (19) gives, $a + b\rho^{-\delta} = \rho$. Since $0 < \rho < 1$ (Remark 4) and $a, b \geq 0$, this implies, $\rho \geq a + b\rho^{-\delta_t}$ for any $0 < \delta_t < \delta$. The remaining rows of $D\mathbf{A}_{\delta_t}D^{-1}$ have one ρ, and all the other entries zero. Hence, $\left\|D\mathbf{A}_{\delta_t}D^{-1}\right\|_\infty \leq \rho$ for all $0 \leq \delta_t \leq \delta$. So if $u_{t+1} = \mathbf{A}_{\delta_t}u_t$, we have, $\|Du_{t+1}\|_\infty = \left\|D\mathbf{A}_{\delta_t}D^{-1}Du_t\right\|_\infty \leq \rho\|Du_t\|_\infty$. This gives

$$p_{t+1} \leq \max(p_{t+1}, \rho p_t, \ldots, \rho^\delta p_{t-\delta+1}) \leq \rho \max(p_t, \rho p_{t-1}, \ldots, \rho^\delta p_{t-\delta}). \tag{30}$$

Hence the result. □

4.4.1. Bounding polynomial for the characteristic polynomial

In light of Lemma 5, we need to focus only on $\delta_t = \delta$. Finding a closed form analytic expression for the unique positive root of (29) is hard. Therefore, we find a polynomial that bounds $p_\delta(\gamma)$ on the interval $[0, 1]$. This provides us with an upper bound on the spectral radius and we use this upper bound as the convergence rate of APSGD.

Lemma 6. *The polynomial,*

$$g_\delta(\gamma) := \left(1 + \frac{1}{\delta} - K_1\right)\gamma^\delta - \left(K_2 + \frac{1}{\delta}\right), \tag{31}$$

bounds the characteristic polynomial $p_\delta(\gamma)$ from below on $[0, 1]$ and its root

$$\beta(\theta, \omega, \delta) = \left(\frac{K_2 + \frac{1}{\delta}}{1 - K_1 + \frac{1}{\delta}}\right)^{\frac{1}{\delta}} \tag{32}$$

is an upper bound to the unique positive root of $p_\delta(\gamma)$.

Proof. We have

$$p_\delta(0) \geq g_\delta(0) \text{ and } p_\delta(1) = g_\delta(1).$$

Let $q_\delta(\gamma) = p_\delta(\gamma) - g_\delta(\gamma)$ then $q_\delta(\gamma) = \gamma^{\delta+1} - \left(1 + \frac{1}{\delta}\right)\gamma^\delta + \frac{1}{\delta}$, and $q'_\delta(\gamma) = (\delta + 1)(\gamma - 1)\gamma^{\delta-1} \leq 0$ for all $\gamma \in [0, 1]$. Therefore, for all $\gamma \in [0, 1]$, we have $q_\delta(\gamma) \geq q_\delta(1)$, which implies $q_\delta(\gamma) \geq 0$. Thus, $p_\delta(\gamma) - g_\delta(\gamma) \geq 0$ for all $\gamma \in [0, 1]$. Therefore, the root of $g_\delta(\gamma)$, $\beta(\theta, \omega, \delta) = \left(\frac{K_2 + \frac{1}{\delta}}{1 - K_1 + \frac{1}{\delta}}\right)^{\frac{1}{\delta}}$ is an upper bound to the unique positive root of the characteristic polynomial $p_\delta(\gamma)$. □

4.5. Convergence results

From now on, we denote $\beta(\theta, \omega, \delta)$ as the rate of convergence of APSGD with stepsize ω, damping parameter θ, and maximum delay δ. From (32) we have

$$\beta(\theta, \omega, \delta) = \left(\frac{K_2 + \frac{1}{\delta}}{1 - K_1 + \frac{1}{\delta}} \right)^{\frac{1}{\delta}}. \tag{33}$$

Let $\beta_{a_{\text{opt}}}$ denote the rate of convergence of APSGD with optimally tuned θ and ω. That is,

$$\beta_{a_{opt}}(\delta) = \min_{\theta,\omega} \left(\frac{K_2 + \frac{1}{\delta}}{1 - K_1 + \frac{1}{\delta}} \right)^{\frac{1}{\delta}}. \tag{34}$$

Let

$$P_\delta = \max \left(\|x_\delta - x_\star\|^2_{\mathbf{B}}, \beta_{a_{opt}}(\delta)\|x_{\delta-1} - x_\star\|^2_{\mathbf{B}}, \ldots, \beta_{a_{opt}}(\delta)^\delta \|x_0 - x_\star\|^2_{\mathbf{B}} \right). \tag{35}$$

Then, we obtain the following convergence guarantee

$$\mathbb{E}\left[\|x_T - x_\star\|^2_{\mathbf{B}} \right] \leq \beta_{a_{opt}}(\delta)^{T-\delta} P_\delta.$$

We define,

$$\chi_{a_{opt}}(\delta) := \frac{\delta}{1 - (\beta_{a_{opt}})^\delta} \geq \frac{\delta}{\log\left(\beta_{a_{opt}}^{-\delta}\right)} = \frac{1}{\log\left(\beta_{a_{opt}}^{-1}\right)}. \tag{36}$$

such that if

$$T - \delta \geq \chi_{a_{opt}}(\delta) \log\left(\frac{1}{\epsilon}\right), \quad \text{then this implies } \mathbb{E}\left[\|x_T - x_\star\|^2_{\mathbf{B}} \right] \leq \epsilon P_\delta. \tag{37}$$

Plugging (34) in (36), we have

$$\chi_{a_{opt}}(\delta) = \delta \cdot \min_{\theta,\omega} \left(\frac{1 - K_1 + \frac{1}{\delta}}{1 - K_1 - K_2} \right). \tag{38}$$

Denote

$$U(\theta, \omega) = \frac{1 - K_1 + \frac{1}{\delta}}{1 - K_1 - K_2}. \tag{39}$$

Therefore,

$$\chi_{a_{opt}}(\delta) = \delta \cdot \min_{\theta,\omega} U(\theta, \omega). \tag{40}$$

Note that, the above expression for $U(\theta, \omega)$ is feasible only when $K_1 + K_2 < 1$. We now only remain to optimize for $U(\theta, \omega)$ to derive our convergence result. In the next Section we show that we can find optimal (θ, ω) for **Case 1**, and a good combination of (θ, ω) for **Case 2**.

4.5.1. The optimal ω

Lemma 7. *Assume that we have two choices of ω, say, ω_1 and ω_2, such that for all $\theta \in [0,1]$, both K_1 and K_2 are smaller for ω_1 than ω_2. Then ω_1 is a better choice than ω_2.*

Proof. For all $\theta \in [0,1]$, we have

$$\begin{aligned} K_1(\theta,\omega_1) &\leq K_1(\theta,\omega_2) \\ \text{and} \quad K_2(\theta,\omega_1) &\leq K_2(\theta,\omega_2). \end{aligned}$$

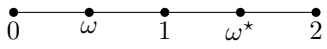
By the definition of $U(\theta,\omega)$ in (39) (provided $K_1 + K_2 < 1$ in both choices of parameters) $U(\theta,\omega_1) \leq U(\theta,\omega_2)$ for all $\theta \in [0,1]$. Hence the result. □

Lemma 8. *The optimal ω for all θ in $[0,1]$ in both the cases (see Theorem 2) lies in the range $[1,\omega^\star]$.*

Proof. First we show how K_1 and K_2 behave for both the cases in the range $\omega \in [0,1]$ and $\omega \in [\omega^\star,\infty)$. Define $s(\omega) := (1-\omega(2-\omega)\lambda^+_{\min})$ and $t(\omega) := (1-\omega(2-\omega)\lambda_{\max})$.

Case 1:

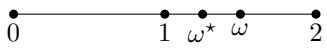
(i) For $0 \leq \omega \leq 1$ and $\alpha(\omega) = 1 - \omega\lambda^+_{\min}$:



$$K_1(\theta,\omega) := (1-\theta)(1-\theta+\theta\alpha(\omega)),\ K_2(\theta,\omega) := \theta(\theta\underbrace{(1-\omega(2-\omega)\lambda^+_{\min})}_{s(\omega)}+(1-\theta)\alpha(\omega)). \tag{41}$$

We see that both $\alpha(\omega)$ and $s(\omega)$ are monotonically decreasing in the interval $\omega \in [0,1]$. Hence, both K_1 and K_2 are monotonically decreasing in $\omega \in [0,1]$.

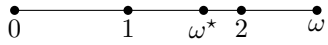
(ii) For $\omega^\star \leq \omega \leq 2$ and $\alpha(\omega) = \omega\lambda_{\max} - 1$:



$$K_1(\theta,\omega) := (1-\theta)(1-\theta+\theta\alpha(\omega)),\ K_2(\theta,\omega) := \theta(\theta\underbrace{(1-\omega(2-\omega)\lambda^+_{\min})}_{s(\omega)}+(1-\theta)\alpha(\omega)).$$

Again, both $\alpha(\omega)$ and $s(\omega)$ are monotonically increasing in the interval $\omega \in [\omega^\star, 2]$. Hence, both K_1 and K_2 are monotonically increasing in $\omega \in [\omega^\star, 2]$.

(iii) For $\omega \geq 2$ and $\alpha(\omega) = \omega\lambda_{\max} - 1$:

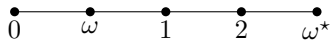


$$K_1(\theta,\omega) := (1-\theta)(1-\theta+\theta\alpha(\omega)),\ K_2(\theta,\omega) := \theta(\theta\underbrace{(1-\omega(2-\omega)\lambda_{\max})}_{t(\omega)}+(1-\theta)\alpha(\omega)).$$

Again, both $\alpha(\omega)$ and $t(\omega)$ are monotonically increasing in the interval $\omega \in [2, \infty)$. Hence, both K_1 and K_2 are monotonically increasing in $\omega \in [2, \infty)$.

Case 2:

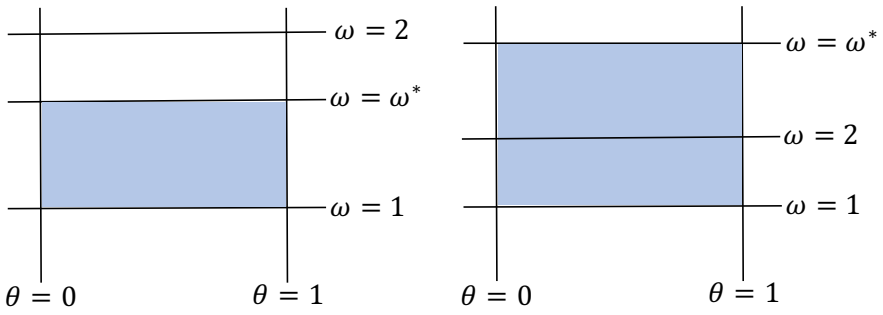
(i) For $0 \leq \omega \leq 1$ and $\alpha(\omega) = 1 - \omega\lambda^+_{\min}$:



$$K_1(\theta,\omega) := (1-\theta)(1-\theta+\theta\alpha(\omega)),\ K_2(\theta,\omega) := \theta(\theta\underbrace{(1-\omega(2-\omega)\lambda^+_{\min})}_{s(\omega)}+(1-\theta)\alpha(\omega)). \tag{42}$$

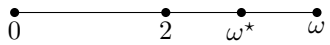
Similar to case 1, both $\alpha(\omega)$ and $s(\omega)$ are monotonically decreasing in the interval $\omega \in [0, 1]$. Hence, both K_1 and K_2 are monotonically decreasing in $\omega \in [0, 1]$.

(ii) For $\omega \geq \omega^\star$ and $\alpha(\omega) = \omega\lambda_{\max} - 1$:



(a) (b)

Figure 2: The region of interest $(\theta, \omega) = [0, 1] \times [1, \omega^\star]$ is shaded– (a) Case 1: $\omega^\star \leq 2$, that is, $\lambda^+_{\min} + \lambda_{\max} \geq 1$, (b) Case 2: $\omega^\star \geq 2$, that is, $\lambda^+_{\min} + \lambda_{\max} \leq 1$.



$$K_1(\theta,\omega) := (1-\theta)(1-\theta+\theta\alpha(\omega)),\ K_2(\theta,\omega) := \theta(\theta\underbrace{(1-\omega(2-\omega)\lambda_{\max})}_{t(\omega)}+(1-\theta)\alpha(\omega)).$$

Again, both $\alpha(\omega)$ and $t(\omega)$ are monotonically increasing in the interval $\omega \in [\omega^\star, \infty)$. Hence, both K_1 and K_2 are monotonically increasing in $\omega \in [\omega^\star, \infty)$.

Now, for both the cases, for all $\theta \in [0, 1]$ we have the following:

$\omega \leq 1$: Both K_1 and K_2 monotonically decrease till $\omega = 1$, then from Lemma 7, 1 is the optimal ω in $[0, 1]$. (Note that $\omega^\star$ is always greater than or equal to 1.)

$\geq \omega^\star$: Both K_1 and K_2 monotonically increase after $\omega = \omega^\star$, then from remark 7, $\omega^\star$ is the optimal ω in $[\omega^\star, \infty)$.

Thus, the optimal ω for all $\theta \in [0, 1]$ in both the cases lies in the range $[1, \omega_\star]$. □

Thus Lemma 8 shows us that we only need to constrain ourselves in $\theta \in [0, 1]$ and $\omega \in [1, \omega^\star]$ for finding the minimum value of $U(\theta, \omega)$.

4.5.2. Upper bound on $U(\theta, \omega)$ for Case 1

As we proved in Lemma 8, we only need to focus in the region $\omega \in [1, \omega^\star]$. See Figure 2a. From (23) we have:

$$K_1(\theta,\omega) := (1-\theta)^2+\theta(1-\theta)(1-\omega\lambda^+_{\min}), \qquad K_2(\theta,\omega) := \theta^2\left(1-\omega(2-\omega)\lambda^+_{\min}\right)+\theta(1-\theta)(1-\omega\lambda^+_{\min}). \tag{43}$$

Substituting the values of K_1 and K_2 in (39), we have

$$U(\theta,\omega) = \frac{\theta\left(1+(1-\theta)\,\omega\lambda^+_{\min}\right)+\frac{1}{\delta}}{\theta\omega\,(2-\theta\omega)\,\lambda^+_{\min}}. \tag{44}$$

Now we compute ∇U and find:

$$\frac{\partial U}{\partial\theta} = \frac{\theta^2\omega\delta\left(1+(\omega-2)\,\lambda^+_{\min}\right)+2\,(\theta\omega-1)}{\theta^2\omega\delta\,(2-\theta\omega)^2\,\lambda^+_{\min}}. \tag{45}$$

Setting $\frac{\partial U}{\partial\theta}=0$ we have

$$\theta^2\omega\delta\left(1+(\omega-2)\,\lambda^+_{\min}\right)+2\,(\theta\omega-1)=0, \tag{46}$$

which implies

$$\theta^2\left(\omega\delta+\omega^2\delta\lambda^+_{\min}-2\omega\delta\lambda^+_{\min}\right)+\theta\,(2\omega)-2=0. \tag{47}$$

Similarly,

$$\frac{\partial U}{\partial\omega} = \frac{2\delta\theta\,(\theta\omega-1)+\delta\theta^2\omega^2\,(1-\theta)\,\lambda^+_{\min}+2\,(\theta\omega-1)}{\theta\omega^2\delta\,(2-\theta\omega)^2\,\lambda^+_{\min}}, \tag{48}$$

and setting $\frac{\partial U}{\partial\omega}=0$ gives

$$2\delta\theta\,(\theta\omega-1)+\delta\theta^2\omega^2\,(1-\theta)\,\lambda^+_{\min}+2\,(\theta\omega-1)=0, \tag{49}$$

which implies

$$\theta^3\left(-\omega^2\delta\lambda^+_{\min}\right)+\theta^2\left(\omega^2\delta\lambda^+_{\min}+2\omega\delta\right)+\theta\,(2\omega-2\delta)-2=0. \tag{50}$$

For the gradient ∇U to vanish inside the region of interest ($\omega\in(1,\omega^\star)$ and $\theta\in(0,1)$), we want both equations (47) as well as (50) to hold. We try to find out the solutions of: $(47)-(50)=0$, as any solution satisfying both equations (47) and (50) also satisfies $(47)-(50)=0$. Therefore, we have

$$(47)-(50) = \theta^3\left(\omega^2\delta\lambda^+_{\min}\right)+\theta^2\left(-2\omega\delta\lambda^+_{\min}-\omega\delta\right)+2\theta\delta \tag{51}$$

$$= \delta\theta\underbrace{\left(\lambda^+_{\min}\theta^2\omega^2-\left(2\lambda^+_{\min}+1\right)\theta\omega+2\right)}_{\text{A quadratic in }\theta\omega}. \tag{52}$$

One solution of $(47) - (50) = 0$ from above is $\theta = 0$, which does not lie inside the region of interest. Next we focus on the quadratic:

$$\lambda_{\min}^{+}\theta^2\omega^2 - \left(2\lambda_{\min}^{+} + 1\right)\theta\omega + 2 \ = 0, \tag{53}$$

that is,

$$\lambda_{\min}^{+}\left(\theta\omega - \frac{1}{\lambda_{\min}^{+}}\right)(\theta\omega - 2) \ = 0, \tag{54}$$

which gives the solution $\theta\omega = \frac{1}{\lambda_{\min}^{+}}$ and $\theta\omega = 2$. The maximum value of $\theta\omega$ attainable in the region of interest is $\omega^\star$, for $\theta = 1$ and $\omega = \omega^\star$. Hence $\theta\omega = \frac{1}{\lambda_{\min}^{+}}$ is not attainable inside the region of interest. Also, as we are dealing with Case 1, $\omega^\star \leq 2$, hence $\theta\omega = 2$ is also not attainable inside the region of interest. Thus,∇U does not vanish inside the region of interest. Therefore, the minima lies on the boundary of the region. We now discuss following four boundary cases:

$\theta = 0$. U is not defined.

$\theta = 1$**.** We have

$$U(1,\omega) = \underbrace{\frac{1 + \frac{1}{\delta}}{\omega(2-\omega)\lambda_{\min}^{+}}}_{\text{minimized at } \omega=1} \implies U(1,1) = \frac{1 + \frac{1}{\delta}}{\lambda_{\min}^{+}}.$$

We note that $U(1,\omega) > \frac{1}{\lambda_{\min}^{+}}$ for all δ.

$\omega = 1$**.** Let $\theta_1 = \underset{\theta}{\operatorname{argmin}} U(\theta, 1)$. Then θ_1 is the solution of equation (47) at $\omega = 1$. Substituting $\omega = 1$ in (47) we get the following quadratic in θ

$$\theta^2\delta(1 - \lambda_{\min}^{+}) + 2\theta - 2 = 0. \tag{55}$$

As we want θ in $[0, 1]$ and find

$$\theta_1 = \frac{\sqrt{1 + 2\delta(1 - \lambda_{\min}^{+})} - 1}{\delta(1 - \lambda_{\min}^{+})}. \tag{56}$$

Therefore,

$$U(\theta_1, 1) = \frac{\frac{3}{4} + \frac{1+\sqrt{1+2\delta(1-\lambda_{\min}^{+})}}{4\delta}}{\lambda_{\min}^{+}}. \tag{57}$$

Note that for $\delta \geq 4$, $U(\theta_1, 1) \leq \frac{1}{\lambda^+_{\min}}$. Thus, when $c\tau \geq 4$, the asynchronous parallel method performs better than the basic method for Case 1.

$\omega = \omega^\star$**.** Let $\theta_{\omega^\star} = \underset{\theta}{\operatorname{argmin}} U(\theta, \omega^\star)$. Then we get $\theta_{\omega^\star}$ as:

$$\theta_{\omega^\star} = \frac{k(\sqrt{1+\delta(2-k)}-1)}{\delta(2-k)} \qquad (k = \lambda^+_{\min} + \lambda_{\max}). \tag{58}$$

Therefore,

$$U(\theta_{\omega^\star}, \omega^\star) = \frac{\frac{k}{4} + \frac{\lambda^+_{\min}}{2} + \frac{\sqrt{1+\delta(2-k)}+2(\delta+1+\delta(1-k)\lambda^+_{\min})}{2\delta\sqrt{1+\delta(2-k)}}}{\lambda^+_{\min}} \tag{59}$$

$$= \frac{\frac{3\lambda^+_{\min}+\lambda_{\max}}{4} + \frac{\sqrt{1+\delta(2-k)}+2(\delta+1+\delta(1-k)\lambda^+_{\min})}{2\delta\sqrt{1+\delta(2-k)}}}{\lambda^+_{\min}}. \tag{60}$$

Therefore, in Case 1, the optimal parameter combination is $(\theta_1, 1)$ or $(\theta_{\omega^\star}, \omega^\star)$. This leads us to the below result.

Theorem 4. *If* $\lambda^+_{\min}+\lambda_{\max} \geq 1$. *The quantities,* $\chi_{a_{opt}}(\delta), P_\delta, U(\theta_1, 1)$ *and* $U(\theta_{\omega^\star}, \omega^\star)$ *are given in equations* (38)*,* (35)*,* (57) *and* (60)*, respectively. If APSGD (Algorithm 1) runs for*

$$T \geq \delta \cdot \min\left(U(\theta_1, 1), U(\theta_{\omega^\star}, \omega^\star)\right) \log\left(\frac{1}{\epsilon}\right) + \delta \geq \chi_{a_{opt}}(\delta) \log\left(\frac{1}{\epsilon}\right) + \delta$$

iterations, then we have $\mathbb{E}\left[\|x_T - x_\star\|^2_{\mathbf{B}}\right] \leq \epsilon P_\delta$,

4.5.3. Upper bound on $U(\theta, \omega)$ *for Case 2*

We split the interval into two parts: $\omega \in [1, 2]$ and $\omega \in [2, \omega^\star]$. See Figure 2b.

Part 1 : $\omega \in [1, 2]$**.** $K_1(\theta, \omega)$ and $K_1(\theta, \omega)$ are the same as described in equation (43). Similar to the Case 1, we find that ∇U does not vanish inside the region $\theta \in (0, 1)$ and $\omega \in (1, 2)$. Therefore, the minima lies on the boundary of the region. We now discuss four boundary cases:

$\theta = 0$**.** U is not defined.

$\theta = 1$**.** We have

$$U(1, \omega) = \underbrace{\frac{1+\frac{1}{\delta}}{\omega(2-\omega)\lambda^+_{\min}}}_{\text{minimized at } \omega=1} \implies U(1, 1) = \frac{1+\frac{1}{\delta}}{\lambda^+_{\min}}.$$

We note that $U(1,\omega) > \frac{1}{\lambda'_{\min}}$ for all δ.

$\omega = 1$**.** Let $\theta_1 = \underset{\theta}{\operatorname{argmin}} U(\theta, 1)$. Then we get θ_1 as:

$$\theta_1 = \frac{\sqrt{1 + 2\delta(1-\lambda'_{\min})} - 1}{\delta(1-\lambda'_{\min})}. \tag{61}$$

Therefore,

$$U(\theta_1, 1) = \frac{\frac{3}{4} + \frac{1+\sqrt{1+2\delta(1-\lambda'_{\min})}}{4\delta}}{\lambda'_{\min}}. \tag{62}$$

We note that for $\delta \geq 4$, $U(\theta_1, 1) \leq \frac{1}{\lambda'_{\min}}$.

$\omega = 2$**.** Let $\theta_2 = \underset{\theta}{\operatorname{argmin}} U(\theta, 2)$. Then we get θ_2 as:

$$\theta_2 = \frac{\sqrt{\delta + 1} - 1}{\delta}. \tag{63}$$

Therefore,

$$U(\theta_2, 2) = \frac{\frac{1}{4} + \frac{\lambda'_{\min}}{2} + \frac{1+\sqrt{\delta+1}}{2\delta}}{\lambda'_{\min}}. \tag{64}$$

We note that for $\delta \geq 3$, $U(\theta_2, 2) \leq \frac{1}{\lambda'_{\min}}$. Thus, for $c\tau \geq 3$, the asynchronous parallel method performs better than the basic method for Case 2.

Lemma 9. *For Case 2,*

$$U(\theta_2, 2) \leq U(\theta_1, 1) \leq U(1, 1), \quad \text{for all } \delta \geq 4. \tag{65}$$

Proof. The first inequality follows from the fact that $\lambda'_{\min} \leq \frac{1}{2}$ for Case 2 (as $\lambda'_{\min} + \lambda_{\max} \leq 1$). The second follows from the fact that $U(1,\omega)$ is always greater than $\frac{1}{\lambda'_{\min}}$ for all δ, whereas $U(\theta_1, 1)$ is smaller than $\frac{1}{\lambda'_{\min}}$ for $\delta \geq 4$. □

Thus, in Case 2, the optimal stepsize is $\omega = 2$ for $\omega \in [1, 2]$. This leads us to the following result:

Theorem 5. *Let $\lambda'_{\min} + \lambda_{\max} \leq 1$. The quantities, $\chi_{a_{opt}}(\delta)$ and P_δ are defined in* (38) *and* (35)*, respectively. If APSGD (Algorithm 1) runs for*

$$T \geq \frac{\frac{1}{4} + \frac{\lambda'_{\min}}{2} + \frac{1+\sqrt{\delta+1}}{2\delta}}{\lambda'_{\min}} \delta \log\left(\frac{1}{\epsilon}\right) + \delta \geq \chi_{a_{opt}}(\delta) \log\left(\frac{1}{\epsilon}\right) + \delta$$

iterations, then we have $\mathbb{E}\left[\|x_T - x_\star\|^2_{\mathbf{B}}\right] \leq \epsilon P_\delta$.

Part 2: $\omega \in [2, \omega^\star]$. We were not able to find the optimal stepsize θ for Case 2 in $\omega \in [2, \omega^\star]$. However, the iteration complexity for $\omega = 2$ is a non-trivial and significant result.

5. Comparison between APSGD and SPSGD

We now compare APSGD and SPSGD. Note that APSGD has two benefits over SPSGD:

- **Intra-iteration gain:** The fast workers do not stay idle waiting for the slow workers in APSGD. In contrast, all workers except the slowest stay idle in SPSGD. The *slowness* can be attributed to less processing power, or/and due to processing a "tougher" job in terms of time complexity, or slower communication between this worker and the master.

- **Inter-iteration gain:** There is no *synchronization barrier* in APSGD. Even when all the workers are equally fast, workers stay idle in SPSGD during the gradient aggregation phase. In contrast, no such phase exists in APSGD – all workers except the one synchronizing with the master are busy in APSGD.

SPSGD takes one step after averaging τ gradient computations, one by each worker. On the other hand, APSGD takes one step after every gradient computation. To compare SPSGD and APSGD, we assume time equivalence between one step of SPSGD and δ steps of APSGD; see Figure 1. We note this setting only considers the intra-iteration gain of APSGD due to the synchronization barrier every δ steps. Thus, APSGD is expected to perform even better without the synchronization barrier, but we do not consider that in our setting.

With the above time equivalence, we compare between $\chi_{s_{opt}}(\tau)$ and $\frac{\chi_{a_{opt}}(\delta)}{\delta}$. We note that we ignore the δ iteration offset in Theorem 4 and Theorem 5 for the ease of exposition. We denote the maximum delay observed by APSGD as $\delta = c\tau$, where $c \geq 1$. We perform two types of comparisons: (*i*) asymptotic comparison, where we compare if APSGD has faster convergence than SPSGD as the number of processors $\tau \to \infty$; (*ii*) non-asymptotic comparison, where we determine the exact number of

processors after which APSGD performs better than SPSGD. Surprisingly, for many ill-conditioned problems for SPSGD, this number is in the order of ten or less.

5.1. Asymptotic comparison

We know from (12) that the best iteration complexity $\chi_{s_{opt}}(\tau)$ of the synchronous parallel method is

$$\chi_{s_{opt}}(\tau) = \frac{\frac{1}{\tau} + \left(1 - \frac{1}{\tau}\right)\lambda_{\max}}{\lambda^+_{\min}}.$$

Rearranging the above equation we get

$$\chi_{s_{opt}}(\tau) = \frac{\lambda_{\max} + \frac{1-\lambda_{\max}}{\tau}}{\lambda^+_{\min}}. \tag{66}$$

Therefore,

$$\lim_{\tau\to\infty} \chi_{s_{opt}}(\tau) = \frac{\lambda_{\max}}{\lambda^+_{\min}}. \tag{67}$$

We now derive the normalized asymptotic iteration complexity of APSGD and compare with the iteration complexity of SPSGD. Since the iteration complexity of APSGD depends on whether $(\lambda^+_{\min} + \lambda_{\max})$ is less than or greater than 1, we consider both cases separately.

5.1.1. Case 1

Substituting $\delta = c\tau$ in equation (60) we get

$$\frac{\chi_a(\theta_{\omega^\star}, \omega^\star, \tau, c)}{c\tau} = \frac{\frac{3\lambda^+_{\min}+\lambda_{\max}}{4} + \frac{\sqrt{1+c\tau(2-k)}+2(c\tau+1+c\tau(1-k)\lambda^+_{\min})}{2c\tau\sqrt{1+c\tau(2-k)}}}{\lambda^+_{\min}}. \qquad (k = \lambda^+_{\min} + \lambda_{\max}). \tag{68}$$

Therefore,

$$\lim_{\tau\to\infty} \frac{\chi_a(\theta_{\omega^\star}, \omega^\star, \tau, c)}{c\tau} = \frac{\frac{3}{4}\lambda^+_{\min} + \frac{\lambda_{\max}}{4}}{\lambda^+_{\min}}. \tag{69}$$

Thus, comparing equations (67) and (69), we find out that APSGD has faster asymptotic convergence than SPSGD in Case 1 (when $(\lambda^+_{\min} + \lambda_{\max}) \in [1, 2]$).

5.1.2. Case 2

Similarly for Case 2, from (64) we get

$$\frac{\chi_a(\theta_2, 2, \tau, c)}{c\tau} = \frac{\frac{1}{4} + \frac{\lambda^+_{\min}}{2} + \frac{1+\sqrt{c\tau+1}}{2c\tau}}{\lambda^+_{\min}}. \tag{70}$$

$\lambda_{\min}^{+}$	$\lambda_{\max}$	$\lambda_{\min}^{+}+\lambda_{\max}$	κ	$\tau_{\min}$		
				$\delta=\tau$ $(c=1)$	$\delta=1.5\tau$ $(c=1.5)$	$\delta=2\tau$ $(c=2)$
10^{-1}	0.9	1	9	5	3	3
10^{-2}	0.99	1	99	4	3	2
10^{-3}	0.999	1	999	4	3	2
10^{-4}	0.9999	1	9999	4	3	2
2×10^{-1}	0.8	1	4	7	5	4

Table 2: **Case 1:** $\lambda_{\min}^{+}+\lambda_{\max}\geq 1$. Minimum number of processors, $\tau_{\min}$ required for APSGD to have faster convergence than SPSGD for different problem instances and maximum delay δ. We assume APSGD has δ updates in the time SPSGD has one update (See Figure 1).

Therefore,

$$\lim_{\tau\to\infty}\frac{\chi_a(\theta_2,2,\tau,c)}{c\tau}=\frac{\frac{1}{4}+\frac{\lambda_{\min}^{+}}{2}}{\lambda_{\min}^{+}}. \tag{71}$$

Finally, comparing equations (66) and (71), we find out that APSGD has faster asymptotic convergence than SPSGD in Case 2 (when $(\lambda_{\min}^{+}+\lambda_{\max})\in[0,1]$) and $\frac{1}{4}+\frac{\lambda_{\min}^{+}}{2}\leq\lambda_{\max}$ hold.

5.2. Non asymptotic comparison

As the above comparisons are asymptotic, we determine when exactly APSGD is better to use than SPSGD. For given combinations of $\lambda_{\min}^{+}$, $\lambda_{\max}$ and c, we compute the minimum number of processors, $\tau_{\min}$, for which the APSGD has better iteration complexity than SPSGD. We have considered $c\in\{1,1.5,2\}$, and various values of interest for $\lambda_{\min}^{+}$ and $\lambda_{\max}$. Denote $\kappa=\frac{\lambda_{\max}}{\lambda_{\min}^{+}}$. κ is the condition number for SPSGD, and (67) shows us that the larger κ is, the higher the iteration complexity of SPSGD. Throughout, $c=1$ corresponds to when the synchronous variant does not suffer from idle time due to load imbalance within phases.

$\lambda_{\min}^{+}$	$\lambda_{\max}$	$\lambda_{\min}^{+} + \lambda_{\max}$	κ	$\tau_{\min}$		
				$\delta = \tau$ $(c = 1)$	$\delta = 1.5\tau$ $(c = 1.5)$	$\delta = 2\tau$ $(c = 2)$
10^{-2}	0.4	0.41	40	12	5	2
10^{-2}	0.3	0.31	30	116	66	40
10^{-2}	0.27	0.28	27	1082	688	491
10^{-2}	0.26	0.27	26	9905	6504	4803
10^{-3}	0.4	0.401	400	11	5	2
10^{-3}	0.3	0.301	300	95	54	31
10^{-3}	0.27	0.271	270	635	398	278
10^{-3}	0.26	0.261	260	2721	1761	1281
10^{-4}	0.4	0.4001	4000	11	5	2
10^{-4}	0.3	0.3001	3000	94	52	31
10^{-4}	0.27	0.2701	2700	606	379	265
10^{-4}	0.26	0.2601	2600	2478	1602	1163
10^{-5}	0.4	0.40001	40000	11	5	2
10^{-5}	0.3	0.30001	30000	93	52	31
10^{-5}	0.27	0.27001	27000	604	377	264
10^{-5}	0.26	0.26001	26000	2456	1587	1152

Table 3: **Case 2:** $\lambda_{\min}^{+} + \lambda_{\max} \leq 1$**.** Minimum number of processors, $\tau_{\min}$ required for APSGD to have faster convergence than SPSGD for different problem instances and maximum delay δ. We assume APSGD has δ updates in the time SPSGD has one update (See Figure 1).

5.2.1. Case 1

Table 2 refers to the first case, when $\lambda^+_{\min} + \lambda_{\max} \geq 1$. We also add a column for the SPSGD condition number, κ. We notice that APSGD has a small $\tau_{\min}$ even for highly ill conditioned problems ($\kappa \sim 10^4$) in Case 1.

5.2.2. Case 2

Table 3 refers to the second case, when $\lambda^+_{\min} + \lambda_{\max} \leq 1$. We note that SPSGD is always better when $\frac{1}{4} + \frac{\lambda^+_{\min}}{2} > \lambda_{\max}$. Thus, in order to consider other cases, all the linear systems below satisfy $\frac{1}{4} + \frac{\lambda^+_{\min}}{2} \leq \lambda_{\max}$. We vary $\lambda^+_{\min}$ in log-scale, $\lambda_{\max} \in \{0.4, 0.3, 0.27, 0.26\}$, and $c \in \{1, 1.5, 2\}$.

- **Effect of varying $\lambda_{\max}$.** We observe that APSGD has small $\tau_{\min}$ even for highly ill-conditioned problems if $\lambda_{\max}$ is reasonably larger than $\frac{1}{4} + \frac{\lambda^+_{\min}}{2}$. For example, when $\lambda_{\max} = 0.4$, we have $\tau_{\min} = 2$ for $c = 2$ and $\tau = 11$ for $c = 1$ (for both cases, $\kappa = 40000$). Fixing $c = 2$, for each value of $\lambda^+_{\min}$, we observe that $\tau_{\min}$ increases from 2 for $\lambda_{\max} = 0.4$ to values in thousands for $\lambda_{\max} = 0.26$.

- **Effect of varying $\lambda^+_{\min}$.** We observe that for the same $\lambda_{\max}$, the smaller the $\lambda^+_{\min}$ is, the smaller $\tau_{\min}$ is. That is, for the same $\lambda_{\max}$, APSGD requires less processors to beat its synchronous counterpart for more ill-conditioned problems. For example, for $\lambda_{\max} = 0.27$ and $c = 2$, we have $\tau_{\min} = 491$ when $\lambda^+_{\min} = 10^{-2}$ (in this case, $\kappa = 27$); but $\tau_{\min} = 264$ when $\lambda^+_{\min} = 10^{-5}$ (in this case $\kappa = 27000$).

- **Effect of varying c.** We also observe that increasing c from 1 to 2 significantly brings down $\tau_{\min}$. This intuitively makes sense and follows our setup—as c increases, APSGD performs more steps while SPSGD performs one step.

Acknowledgement

Aritra Dutta acknowledges being an affiliated researcher at the Pioneer Centre for AI, Denmark.

References

[1] P. Richtárik, M. Takác, Stochastic reformulations of linear systems: algorithms and convergence theory, SIAM Journal on Matrix Analysis and Applications 41 (2) (2020) 487–524.

[2] Y. Huang, Y. Cheng, A. Bapna, O. Firat, D. Chen, M. Chen, H. Lee, J. Ngiam, Q. V. Le, Y. Wu, et al., Gpipe: Efficient training of giant neural networks using pipeline parallelism, Advances in neural information processing systems 32 (2019).

[3] S. Shalev-Shwartz, Y. Singer, N. Srebro, Pegasos: Primal estimated sub-gradient solver for SVM, in: International Conference on Machine Learning, 2007, pp. 807–814.

[4] K. Gimpel, D. Das, N. A. Smith, Distributed asynchronous online learning for natural language processing, in: Conference on Computational Natural Language Learning, 2010, pp. 213–222.

[5] O. Dekel, R. Gilad-Bachrach, O. Shamir, L. Xiao, Optimal distributed online prediction using mini-batches, Journal of Machine Learning Research 13 (1) (2012) 165–202.

[6] H. Xu, C.-Y. Ho, A. M. Abdelmoniem, A. Dutta, E. H. Bergou, K. Karatsenidis, M. Canini, P. Kalnis, Grace: A compressed communication framework for distributed machine learning, in: 2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS), IEEE, 2021, pp. 561–572.

[7] B. Recht, C. Re, S. Wright, F. Niu, Hogwild!: A lock-free approach to parallelizing stochastic gradient descent, in: Advances in Neural Information Processing Systems, 2011, pp. 693–701.

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

[9] A. Frommer, D. B. Szyld, On asynchronous iterations, Journal of Computational and Applied Mathematics 123 (2001) 201–216.

[10] D. P. Bertsekas, J. N. Tsitsiklis, Parallel and distributed computation: Numerical methods, 2003.

[11] H. Robbins, S. Monro, A stochastic approximation method, Annals of Mathematical Statistics 22 (1951) 400–407.

[12] M. Zinkevich, M. Weimer, L. Li, A. J. Smola, Parallelized stochastic gradient descent, in: Advances in Neural Information Processing Systems, 2010, pp. 2595–2603.

[13] R. M. Gower, P. Richtárik, Stochastic dual ascent for solving linear systems, arXiv preprint arXiv:1512.06890 (2015).

[14] R. M. Gower, P. Richtárik, Randomized iterative methods for linear systems, SIAM Journal on Matrix Analysis and Applications 36 (4) (2015) 1660–1690.

[15] H. R. Feyzmahdavian, A. Aytekin, M. Johansson, A delayed proximal gradient method with linear convergence rate, in: IEEE International Workshop on Machine Learning for Signal Processing (MLSP), 2014, pp. 1–6.

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

[17] R. Leblond, F. Pedregosa, S. Lacoste-Julien, ASAGA: Asynchronous Parallel SAGA, in: International Conference on Artificial Intelligence and Statistics, 2017, pp. 46–54.

[18] M. P. Forum, MPI: A message-passing interface standard (1994).

[19] P. Richtárik, M. Takáč, Parallel coordinate descent methods for big data optimization, Mathematical Programming 156 (1) (2016) 433–484.

[20] T. Yang, Trading computation for communication: Distributed stochastic dual coordinate ascent, in: Advances in Neural Information Processing Systems, 2013, pp. 629–637.

[21] M. Jaggi, V. Smith, M. Takáč, J. Terhorst, S. Krishnan, T. Hofmann, M. Jordan, Communication-efficient distributed dual coordinate ascent, in: Advances in Neural Information Processing Systems, 2014, pp. 3068–3076.

[22] M. Takác, A. S. Bijral, P. Richtárik, N. Srebro, Mini-batch primal and dual methods for SVMs., in: Advances in Neural Information Processing Systems, 2013, pp. 1022–1030.

[23] I. Necoara, D. Clipici, Efficient parallel coordinate descent algorithm for convex optimization problems with separable constraints: Application to distributed MPC, Journal of Process Control 23 (3) (2013) 243–253.

[24] O. Fercoq, P. Richtárik, Accelerated, parallel, and proximal coordinate descent, SIAM Journal on Optimization 25 (4) (2015) 1997–2023.

[25] P. Richtárik, M. Takáč, Distributed coordinate descent method for learning with big data, The Journal of Machine Learning Research 17 (1) (2016) 2657–2681.

[26] J. K. Bradley, A. Kyrola, D. Bickson, C. Guestrin, Parallel coordinate descent for l_1-regularized loss minimization, in: International Conference on Machine Learning, 2011.

[27] J. Mareček, P. Richtárik, M. Takáč, Distributed block coordinate descent for minimizing partially separable functions, in: Numerical Analysis and Optimization, Springer, 2015, pp. 261–288.

[28] O. Fercoq, Z. Qu, P. Richtárik, M. Takáč, Fast distributed coordinate descent for non-strongly convex losses, in: IEEE International Workshop on Machine Learning for Signal Processing, IEEE, 2014, pp. 1–6.

[29] O. Shamir, N. Srebro, T. Zhang, Communication-efficient distributed optimization using an approximate newton-type method, in: International Conference on Machine Learning, 2014, pp. 1000–1008.

[30] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, Q. Le, A. Ng, Large scale distributed deep networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1223–1231.

[31] R. R. Gajjala, S. Banchhor, A. M. Abdelmoniem, A. Dutta, M. Canini, P. Kalnis, Huffman coding based encoding techniques for fast distributed deep learning, in: Proceedings of ACM CoNEXT's 1st Workshop on Distributed Machine Learning, 2020, pp. 21–27.

[32] A. Dutta, E. H. Bergou, A. M. Abdelmoniem, C.-Y. Ho, A. N. Sahu, M. Canini, P. Kalnis, On the discrepancy between the theoretical analysis and practical implementations of compressed communication for distributed deep learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 3817–3824.

[33] H. Xu, K. Kostopoulou, A. Dutta, X. Li, A. Ntoulas, P. Kalnis, Deepreduce: A sparse-tensor communication framework for federated deep learning, in: Advances in Neural Information Processing Systems, Vol. 34, 2021.

[34] A. Sahu, A. Dutta, A. M Abdelmoniem, T. Banerjee, M. Canini, P. Kalnis, Rethinking gradient sparsification as total error minimization, in: Advances in Neural Information Processing Systems, Vol. 34, 2021.

[35] S. Shalev-Shwartz, T. Zhang, Accelerated mini-batch stochastic dual coordinate ascent, in: Advances in Neural Information Processing Systems, 2013, pp. 378–385.

[36] R. Johnson, T. Zhang, Accelerating stochastic gradient descent using predictive variance reduction, in: Advances in Neural Information Processing Systems, 2013, pp. 315–323.

[37] C. M. De Sa, C. Zhang, K. Olukotun, C. Ré, Taming the wild: A unified analysis of Hogwild!-style algorithms, in: Advances in Neural Information Processing Systems, 2015, pp. 2674–2682.

[38] L. Nguyen, P. H. Nguyen, M. Dijk, P. Richtárik, K. Scheinberg, M. Takáč, SGD and Hogwild! convergence without the bounded gradients Assumption, in: International Conference on Machine Learning, 2018, pp. 3747–3755.

[39] C. Noel, S. Osindero, Dogwild!-distributed hogwild for CPU & GPU, in: Proceedings of the NIPS Workshop on Distributed Machine Learning and Matrix Computations, 2014, pp. 693–701.

[40] S. Chaturapruek, J. C. Duchi, C. Ré, Asynchronous stochastic convex optimization: the noise is in the noise and SGD don't care, in: Advances in Neural Information Processing Systems, 2015, pp. 1531–1539.

[41] S. U. Stich, S. P. Karimireddy, The error-feedback framework: Better rates for SGD with delayed gradients and compressed communication, Journal of Machine Learning Research (2020).

[42] X. Lian, Y. Huang, Y. Li, J. Liu, Asynchronous parallel stochastic gradient for nonconvex optimization, in: Advances in Neural Information Processing Systems, 2015, pp. 2737–2745.

[43] S. Zheng, Q. Meng, T. Wang, W. Chen, N. Yu, Z.-M. Ma, T.-Y. Liu, Asynchronous stochastic gradient descent with delay compensation, in: International Conference on Machine Learning, 2017, pp. 4120–4129.

[44] A. Aytekin, H. R. Feyzmahdavian, M. Johansson, Analysis and implementation of an asynchronous optimization algorithm for the parameter server, arXiv:1610.05507 (2016).

[45] C. Ma, V. Smith, M. Jaggi, M.Jordan, P. Richtárik, M. Takáč, Adding vs. averaging in distributed primal dual optimization, in: International Conference on Machine Learning, 2015, pp. 1973–1982.

[46] J. D. Lee, Q. Lin, T. Ma, T. Yang, Distributed stochastic variance reduced gradient methods by sampling extra data with replacement, The Journal of Machine Learning Research 18 (1) (2017) 4404–4446.

[47] S. Y. Zhao, W. J. Li, Fast asynchronous parallel stochastic gradient descent: A lock-free approach with convergence guarantee, in: AAAI Conference on Artificial Intelligence, 2016, pp. 2379–2385.

[48] F. Pedregosa, R. Leblond, S. Lacoste-Julien, Breaking the nonsmooth barrier: A scalable parallel method for composite optimization, in: Advances in Neural Information Processing System, 2017, pp. 56–65.

[49] X. Lian, W. Zhang, C. Zhang, J. Liu, Asynchronous decentralized parallel stochastic gradient descent, in: International Conference on Machine Learning, 2018, pp. 3043–3052.

[50] C. M. De Sa, M. Feldman, C. Ré, K. Olukotun, Understanding and optimizing asynchronous low-precision stochastic gradient descent, in: ACM SIGARCH Computer Architecture News, Vol. 45, ACM, 2017, pp. 561–574.

[51] T. Ben-Nun, T. Hoefler, Demystifying parallel and distributed deep learning: An in-depth concurrency analysis, ACM Computing Surveys (CSUR) 52 (4) (2019) 65.

[52] V. V. Prasolov, Polynomials, Springer (2009).

6. Notation Glossary

The Basics		
$\mathbf{A}, b$	$m \times n$ matrix and $m \times 1$ vector defining the system $\mathbf{A}x = b$	Related to stochastic reformulation of the linear system (see (5))
$\mathcal{L}$	$\{x \ : \mathbf{A}x = b\}$ (solution set of the linear system)	
$\mathbf{B}$	$n \times n$ symmetric positive definite matrix	
$\langle x, y \rangle_{\mathbf{B}}$	$x^\top \mathbf{B} y$ ($\mathbf{B}$-inner product)	
$\|x\|_{\mathbf{B}}$	$\sqrt{\langle x, x \rangle_{\mathbf{B}}}$ (Norm induced by $\mathbf{B}$-inner product)	
$\mathbf{M}^\dagger$	Moore-Penrose pseudoinverse of matrix $\mathbf{M}$	

$\mathbf{S}$	a random real matrix with m rows	
$\mathcal{D}$	distribution from which matrix $\mathbf{S}$ is drawn	
$\mathbf{H}$	$\mathbf{H} = \mathbf{S}(\mathbf{S}^\top \mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S})^\dagger \mathbf{S}^\top$	
$\mathbf{Z}$	$\mathbf{A}^\top \mathbf{H}\mathbf{A}$	
$\mathrm{Im}(\mathbf{M})$	Image (range) space of matrix $\mathbf{M}$	
$\mathrm{Null}(\mathbf{M})$	Null space of matrix $\mathbf{M}$	
$\mathbb{E}[\cdot]$	expectation	
Projections		
$\Pi_{\mathcal{L}}^{\mathbf{B}}(x)$	projection of x onto the set $\mathcal{L}$ in the $\mathbf{B}$-norm	
$\mathbf{B}^{-1}\mathbf{Z}$	projection matrix in the $\mathbf{B}$-norm onto $\mathrm{Im}(\mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S})$	
Optimization		
$x_\star$	a solution of the linear system $\mathbf{A}x = b$	Assumption 1
$f_{\mathbf{S}}, \nabla f_{\mathbf{S}}, \nabla^2 f_{\mathbf{S}}$	stochastic function, its gradient and Hessian, respectively	
$\mathcal{L}_{\mathbf{S}}$	$\{x : \mathbf{S}^\top \mathbf{A}x = \mathbf{S}^\top b\}$ (set of minimizers of $f_{\mathbf{S}}$)	
f	$\mathbb{E}[f_{\mathbf{S}}]$	
∇f	gradient of f with respect to the $\mathbf{B}$-inner product	
$\nabla^2 f$	$\mathbf{B}^{-1}\mathbb{E}[\mathbf{Z}]$ (Hessian of f in the $\mathbf{B}$-inner product)	
Eigenvalues		
$\mathbf{W}$	$\mathbf{B}^{-\frac{1}{2}}\mathbb{E}[\mathbf{Z}]\mathbf{B}^{-\frac{1}{2}}$ (psd matrix with the same spectrum as $\nabla^2 f$)	
$\lambda_1, \ldots, \lambda_n$	eigenvalues of $\mathbf{W}$	
Λ	$\mathbf{Diag}(\lambda_1, \ldots, \lambda_n)$ (diagonal matrix of eigenvalues)	
$\mathbf{U}$	$[u_1, \ldots, u_n]$ (eigenvectors of $\mathbf{W}$)	
$\mathbf{U}\Lambda\mathbf{U}^\top$	eigenvalue decomposition of $\mathbf{W}$	
$\lambda_{\max}, \lambda_{\min}^+$	largest and smallest nonzero eigenvalues of $\mathbf{W}$	
Algorithms		
θ	damping parameter	
ω	stepsize / relaxation parameter	
$\omega^\star$	$\dfrac{2}{(\lambda_{\min}^+ + \lambda_{\max})}$	Theorem 2
τ	number of processors in the assembly	
δ_t	delay between the master and a particular worker processor	

$\delta(=c\tau)$	maximum delay for APSGD	
	For particular values of θ , ω *and* δ:	
$\alpha(\omega)$	$\max_{i:\lambda_i>0} \lvert 1-\omega\lambda_i \rvert$	Theorem 2
$K_1(\theta,\omega)$,	coefficient of $\mathbb{E}\left[\|r_t\|^2\right]$ and $\mathbb{E}\left[\|r_{t-\delta_t}\|^2\right]$ respectively in Theo-	Theorem 2
$K_2(\theta,\omega)$	rem 2	
$\beta_a(\theta,\omega,\delta)$	rate of convergence of the asynchronous parallel SGD	
$\beta_{a_{\rm opt}(\delta)}$	optimal rate of convergence of the asynchronous parallel SGD	
$\chi_a(\theta,\omega,\delta)$	iteration complexity of the asynchrous parallel SGD	
$\chi_{a_{opt}}(\delta)$	optimal iteration complexity of the asynchrous parallel SGD	
P_δ	$\max\left(\|x_\delta - x_\star\|^2_{\mathbf{B}}, \beta_{a_{opt}}(\delta)\|x_{\delta-1}-x_\star\|^2_{\mathbf{B}}, \ldots, \beta_{a_{opt}}(\delta)^\delta\|x_0-x_\star\|^2_{\mathbf{B}}\right)$	