A^{2}CiD^2: Accelerating Asynchronous Communication in Decentralized Deep Learning

Adel Nabli, Eugene Belilovsky, Edouard Oyallon

A^2CiD^2: Accelerating Asynchronous Communication in Decentralized Deep Learning

Adel Nabli
Concordia University, Mila
Sorbonne University, ISIR, CNRS
adel.nabli@sorbonne-universite.fr

Eugene Belilovsky
Concordia University, Mila

Edouard Oyallon
Sorbonne University, ISIR, CNRS

Abstract

Distributed training of Deep Learning models has been critical to many recent successes in the field. Current standard methods primarily rely on synchronous centralized algorithms which induce major communication bottlenecks and limit their usability to High-Performance Computing (HPC) environments with strong connectivity. Decentralized asynchronous algorithms are emerging as a potential alternative but their practical applicability still lags. In this work, we focus on peer-to-peer asynchronous methods due to their flexibility and parallelization potentials. In order to mitigate the increase in bandwidth they require at large scale and in poorly connected contexts, we introduce a principled asynchronous, randomized, gossip-based algorithm which works thanks to a continuous momentum named $\mathbf{A^2CiD^2}$. In addition to inducing a significant communication acceleration at no cost other than doubling the parameters, minimal adaptation is required to incorporate $\mathbf{A^2CiD^2}$ to other asynchronous approaches. We demonstrate its efficiency theoretically and numerically. Empirically on the ring graph, adding $\mathbf{A^2CiD^2}$ has the same effect as doubling the communication rate. In particular, we show consistent improvement on the ImageNet dataset using up to 64 asynchronous workers (A100 GPUs) and various communication network topologies.

1 Introduction

As Deep Neural Networks (DNNs) and their training datasets become larger and more complex, the computational demands and the need for efficient training schemes continues to escalate. Distributed training methods offer a solution by enabling the parallel optimization of model parameters across multiple workers. Yet, many of the current distributed methods in use are synchronous, and have significantly influenced the design of cluster computing environments. Thus, both the environments and algorithms rely heavily on high synchronicity in machine computations and near-instantaneous communication in high-bandwidth networks, favoring the adoption of centralized algorithms [7].

However, several studies [26, 41, 2, 27] are challenging this paradigm, proposing decentralized asynchronous algorithms that leverage minor time-delays fluctuations between workers to enhance the parallelization of computations and communications. Unlike centralized algorithms, decentralized approaches allow each node to contribute proportionally to its available resources, eliminating the necessity for a global central worker to aggregate results. Combined with asynchronous peer-to-peer communications, these methods can streamline the overall training process, mitigating common bottlenecks. This includes the Straggler Problem [39], the synchro-

Preprint. Under review.

nization between computations and communications [9], or bandwidth limitations [44], potentially due to particular network topologies like a ring graph [40]. However, due to the large number of parameters which are optimized, training DNNs with these methods still critically requires a considerable amount of communications [22], presenting an additional challenge [30]. This work aims to address these challenges by introducing a principled acceleration method for pair-wise communications in peer-to-peer training of DNNs, in particular for cluster computing. While conventional synchronous settings accelerate communications by integrating a Chebychev acceleration followed by Gradient Descent steps [35], the potential of accelerated asynchronous pair-wise gossips for Deep Learning (DL) remains largely unexplored. Notably, the sophisticated theory of Stochastic Differential Equations (SDEs) offers an analytical framework for the design and study of the convergence of these algorithms [12]. We introduce a novel algorithm **A^{2}CiD^2** (standing for **A**ccelerating **A**synchronous **C**ommunication **i**n **D**ecentralized **D**eep Learning) that requires minimal overhead and effectively decouples communications and computations, accelerating pair-wise communications via a provable, accelerated, randomized gossip procedure based on continuous momentum (i.e., a mixing ODE) and time [12, 32]. We emphasize that beyond the aforementioned hardware superiority, stochastic algorithms also allow to theoretically reach sublinear rates in convex settings [10], which opens the possibility to further principled accelerations. In practice, our method enables a virtual doubling of communication rate without any additional cost, simply by maintaining two copies of the model's parameters in each worker (see Fig. 1).

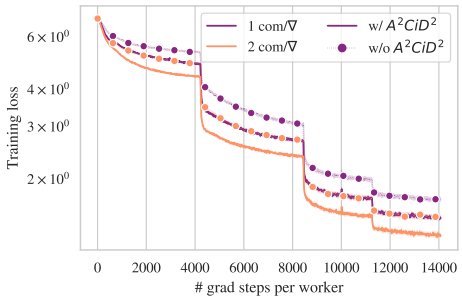


Figure 1: Adding **A^{2}CiD^2** has the same effect as doubling the communication rates on ImageNet on the ring graph with 64 workers. See Sec. 4.

Our key contributions are as follows: **(1)** We extend the continuized framework [12] to the non-convex setting, in order to obtain a neat framework to describe asynchronous decentralized DL training. **(2)** We propose a novel and simple continuized mechanism which allows to significantly improve communication efficiency in challenging settings, which we name **A^{2}CiD^2**. **(3)** We demonstrate that our method effectively minimizes the gap between centralized settings in environments hosting up to 64 asynchronous GPUs. **(4)** Our code is implemented in Pytorch [33], and can be found as an opensource implementation: https://github.com/AdelNabli/A2CiD2.

This paper is structured as follows: Sec. 3.1 outlines our model for asynchronous decentralized learning, while Sec. 3.2 discusses the training dynamic used to optimize our Deep models. Sec. 3.3 offers a comprehensive theoretical analysis of our method, which is validated empirically in Sec. 4.

2 Related Work

Large-scale distributed DL. Two paradigms allow to maintain high-parallelization. On one side, model-parallelism [9, 24], which splits a neural network on independant machines, allowing to use local learning methods [4, 3]. On the other hand data-parallelism, which accelerates learning by making use of larger mini-batch splitted accross multiple nodes [36] to maximally use GPU capacities. This parallelization entailing the use of larger batch-sizes, it requires an important process of adapting hyper-parameters [16], and in particular the learning rate scheduler. Developed for this setting, methods such as [16, 43] allow to stabilize training while maintaining good generalization performances. However, they have been introduced in the context of centralized synchronous training using All-Reduce schemes for communication, which still is the default setting of many approaches to data parallelism.

Decentralized DL. The pioneer work [26] is one of the first study to suggest the potential superiority of synchronous decentralized training strategies in practice. In terms of implementation in the cluster setting, decentralized frameworks have been shown to achieve higher throughput than optimized All-Reduce strategies [42, 36]. From the theoretical side, [21] propose a framework covering many settings of synchronous decentralized learning. However, as it consistently relies on using a global

discrete iterations count, the notion of time is more difficult to exploit, which reveals crucial in our setting. Furthermore, no communication acceleration is incorporated in these algorithms. [22] provides a comprehensive methodology to relate the consensus distance, *i.e.* the average distance of the local parameters to the global average, to a necessary communication rate to avoid degrading performance and could be easily applied to our method. [28] is a method focusing on improving performance via a discrete momentum modification, which indicates momentum variables are key to decentralized DL.

Asynchronous Decentralized DL. There exists many attempts to incorporate asynchrony in decentralized training [45, 5, 8, 27, 2], which typically aim at removing lock barriers of synchronous decentralized algorithms. To the best of our knowledge, none of them introduce communication acceleration, yet they could be simply combined with our approach. Although recent approaches such as [2, 27] perform peer-to-peer averaging of parameters instead of gradients, thus allowing to communicate *in parallel* of computing (as there is no need to wait for the gradients before communicating), they are still coupled: parameter updates resulting from computations and communications are scheduled in a specific order, limiting their speed. Furthermore, in practice, both those works implement a periodic averaging on an exponential graph (more favorable, see [40]), instead of the more general setting of a randomized gossip without restriction on the topology, as we do.

Communication reduction. Reducing communication overhead is an important topic for scalability [34]. For instance, [19, 20] allow to use strong compression factor in limited bandwidth setting, and the local SGD communication schedule of [29] is shown to be beneficial. Those methods could be independently and simply combined with ours to potentially benefit from an additional communication acceleration. By leveraging key properties of the resistance of the communication network [14], [12] showed that standard asynchronous gossip [6] can be accelerated, even to give efficient primal algorithms in the convex setting [32]. However, this acceleration has never been deployed in the DL context, until now. RelaySum [38] is an approach which allow to average exactly parameters produced by different time steps and thus potentially delayed. However, it requires either to use a tree graph topology, either to build ad-hoc spanning trees and has inherent synchronous locks as it averages neighbor messages in a specific order.

Notations: Let $n \in \mathbb{N}^*$ and $d \in \mathbb{N}^*$ an ambient dimension, for $x = (x^1, ..., x^n) \in \bigotimes_{i=1}^n \mathbb{R}^d$, we write $\bar{x} = \frac{1}{n}\sum_{i=1}^n x^i$ and $\mathbf{1}$ the tensor of ones such that $\bar{x} = \frac{1}{n}\mathbf{1}^\mathsf{T} x$. Ξ is a probability space with measure $\mathcal{P}$. $f(t) = \mathcal{O}(1)$ means there is a $C > 0$ such that for t large enough, $|f(t)| \leq C$, whereas $\tilde{\mathcal{O}}$-notation hides constants and polylogarithmic factors..

Table 1: Comparison of convergence rates for strongly convex and non-convex objectives against concurrent works in the fixed topology setting. We neglect logarithmic terms. Observe that thanks to the maximal resistance $\chi_2 \leq \chi_1$, our method obtains substantial acceleration for the bias term.

Method	**Strongly Convex**	**Non-Convex**
Koloskova et al. [21]	$\frac{\sigma^2}{n\mu^2\epsilon} + \sqrt{L}\frac{\chi_1\xi+\sqrt{\chi_1}\sigma}{\mu^{3/2}\sqrt{\epsilon}} + \frac{L}{\mu}\chi_1$	$\frac{L\sigma^2}{n\epsilon^2} + L\frac{\chi_1\xi+\sqrt{\chi_1}\sigma}{\epsilon^{3/2}} + \frac{L\chi_1}{\epsilon}$
A^{2}CiD^2 (Ours)	$\frac{\sigma^2+\sqrt{\chi_1\chi_2}\xi^2}{\mu^2\epsilon} + \frac{L}{\mu}\sqrt{\chi_1\chi_2}$	$L\frac{\sigma^2+\sqrt{\chi_1\chi_2}\xi^2}{\epsilon^2} + \frac{L\sqrt{\chi_1\chi_2}}{\epsilon}$
AD-PSGD [27]	-	$L\frac{\sigma^2+\xi^2}{\epsilon^2} + \frac{nL\chi_1}{\epsilon}$

3 Method

3.1 Model for a decentralized environment

We consider a network of n workers whose connectivity is given by edges $\mathcal{E}$. Local computations are modeled as (stochastic) point-wise processes N_t^i, and communications between nodes $(i, j) \in \mathcal{E}$ as M_t^{ij}. We assume that the communications are symmetric, meaning that if a message is sent from node i to j, then the reverse is true. In practice, such processes are potentially highly correlated and

could follow any specific law, and could involve delays. For the sake of simplicity, we do not model lags, though it is possible to obtain guarantees via dedicated Lyapunov functions [13]. In our setting, we assume that all nodes have similar buffer variables which correspond to a copy of a common model (e.g., a DNN). For a parameter x, we write x_t^i the model parameters at node i and time t and $x_t = (x_t^1, ..., x_t^n)$ their concatenation. In the following, we assume that each worker computes about 1 batch of gradient per unit of time (not necessarily simultaneously), which is a standard homogeneity assumption [18], and we denote by λ^{ij} the instantaneous expected frequency of edge (i, j), which we assume time homogeneous.

Definition 3.1 (Instantaneous expected Laplacian)**.** We define the Laplacian Λ as:

$$\Lambda \triangleq \sum_{(i,j)\in\mathcal{E}} \lambda^{ij}(e_i - e_j)(e_i - e_j)^\mathsf{T} . \tag{1}$$

In this context, a natural quantity is the algebraic connectivity [6] given by:

$$\chi_1 \triangleq \sup_{\|x\|=1, x\perp \mathbf{1}} \frac{1}{x^\mathsf{T}\Lambda x} . \tag{2}$$

For a connected graph (*i.e.* , $\chi_1 < +\infty$), we will also use the maximal resistance of the network:

$$\chi_2 \triangleq \frac{1}{2} \sup_{(i,j)\in\mathcal{E}} (e_i - e_j)^\mathsf{T}\Lambda^+(e_i - e_j) \leq \chi_1 . \tag{3}$$

Next sections will show that it is possible to accelerate the gossip algorithms from χ_1 to $\sqrt{\chi_1\chi_2} \leq \chi_1$. [12] or [32] emphasize the superiority of accelerated asynchronous gossips over accelerated synchronous ones. The complexity of the first is given by $\sqrt{\chi_1\chi_2}$ when the second relies on $\sqrt{\chi_1\|\Lambda\|}$. For instance, for a star graph with n nodes, one has: $\chi_1 = \chi_2 = 1, \|\Lambda\| = n$.

3.2 Training dynamic

The goal of a typical decentralized algorithm is to minimize the following quantity:

$$\inf_{x\in\mathbb{R}^d} f(x) \triangleq \inf_{x\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^n f_i(x) = \inf_{x_i = x_1} \frac{1}{n}\sum_{i=1}^n f_i(x_i) .$$

For this, we follow a first order optimization strategy consisting in using estimates of the gradient $\nabla f_i(x_i)$ via i.i.d unbiased Stochastic Gradient (SG) oracles given by $\nabla F_i(x_i, \xi_i)$ s.t. $\mathbb{E}_{\xi_i}[\nabla F_i(x_i, \xi_i)] = \nabla f_i(x_i)$. The dynamic of updates of our model evolves as the following SDE, for $\eta_t, \gamma_t, \alpha_t, \tilde{\alpha}_t$ some time-dependent scalar hyper-parameters and $dN_t^i(\xi_i)$ some point processes on $\mathbb{R}_+ \times \Xi$ with intensity $dt \otimes d\mathcal{P}$:

$$\begin{aligned} dx_t^i =& \eta(\tilde{x}_t^i - x_t^i)dt - \gamma \int_\Xi \nabla F_i(x_t^i, \xi_i)\, dN_t^i(\xi_i) - \alpha \sum_{j,(i,j)\in\mathcal{E}} (e_i - e_j)(e_i - e_j)^\mathsf{T} x_t dM_t^{ij} , \\ d\tilde{x}_t^i =& \eta(x_t^i - \tilde{x}_t^i)dt - \gamma \int_\Xi \nabla F_i(x_t^i, \xi_i)\, dN_t^i(\xi_i) - \tilde{\alpha} \sum_{j,(i,j)\in\mathcal{E}} (e_i - e_j)(e_i - e_j)^\mathsf{T} x_t dM_t^{ij} . \end{aligned} \tag{4}$$

We emphasize that while the dynamic Eq. 4 is formulated using SDEs [1], which brings the power of the continuous-time analysis toolbox, it is still *event-based* and thus discrete in nature. Hence, it can efficiently model practically implementable algorithms, as shown by Algo. 1. The coupling $\{x_t, \tilde{x}_t\}$ corresponds to a momentum term which will be useful to obtain communication acceleration as explained in the next section. Again, $\int_\Xi \nabla F_i(x_t^i, \xi_i)\, dN_t^i(\xi_i)$ will be estimated via i.i.d SGs sampled as N_t^i spikes. Furthermore, if $\bar{x}_0 = \overline{\tilde{x}_0}$, then, $\bar{x}_t = \overline{\tilde{x}_t}$ and we obtain a tracker of the average across workers which is similar to what is achieved through Gradient Tracking methods [19]. This is a key advantage of our method to obtain convergence guarantees, which writes as:

$$d\bar{x}_t = -\gamma \frac{1}{n}\sum_{i=1}^n \int_\Xi \nabla F_i(x_t^i, \xi_i)\, dN_t^i(\xi_i) . \tag{5}$$

3.3 Theoretical analysis of A^{2}CiD^2

We now provide an analysis of our decentralized, asynchronous algorithm. For the sake of simplicity, we will consider that communications and gradients spike as Poisson processes:

Assumption 3.2 (Poisson Processes)**.** N_t^i, M_t^{ij} are independent, Point-wise Poisson Processes.

We also assume that the communication network is connected during the training:

Assumption 3.3 (Strong connectivity)**.** We assume that $\chi_1 < \infty$.

We will now consider two generic assumptions obtained from [21], which allow to specify our lemma to convex and non-convex settings. Note that the non-convex Assumption 3.5 generalizes the assumptions of [27], by taking $M = P = 0$.

Assumption 3.4 (Strongly convex setting)**.** Each f_i is μ-strongly convex and L-smooth, and:

$$\frac{1}{n}\sum_{i=1}^n \mathbb{E}_{\xi_i}[\|\nabla F_i(x,\xi_i) - \nabla f_i(x)\|^2] \leq \sigma^2 \text{ and } \frac{1}{n}\sum_{i=1}^n \|\nabla f_i(x^*) - \nabla f(x^*)\|^2 \leq \zeta^2\,.$$

Assumption 3.5 (Non-convex setting)**.** Each f_i is L-smooth, and there exists $P, M > 0$ such that:

$$\forall x \in \mathbb{R}^d, \frac{1}{n}\sum_{i=1}^n \|\nabla f_i(x) - \nabla f(x)\|^2 \leq \zeta^2 + P\|\nabla f(x)\|^2\,,$$

and,

$$\forall x_1, ..., x_n \in \mathbb{R}^d, \frac{1}{n}\sum_{i=1}^n \mathbb{E}_{\xi_i}\|\nabla F_i(x_i,\xi_i) - \nabla f_i(x_i)\|^2 \leq \sigma^2 + \frac{M}{n}\sum_{i=1}^n \|\nabla f_i(x_i)\|^2\,.$$

We can now state our convergence guarantees. An informal way to understand our proposition, is that while gradient updates are non-convex, the communications updates are linear and thus benefit from local convexity; its proof is delayed to the Appendix.

Proposition 3.6 (Convergence guarantees.)**.** *Assume that $\{x_t, \tilde{x}_t\}$ follow the dynamic Eq. 4 and that Assumption 3.2-3.3 are satisfied. Assume that $\mathbf{1}\bar{x}_0 = x_0 = \tilde{x}_0$ and let T the total running time. Then:*

- ***Non-accelerated setting**, we pick $\eta = 0, \alpha = \tilde{\alpha} = \frac{1}{2}$ and set $\chi = \chi_1$,*
- ***Acceleration (A^{2}CiD^2)**, we set $\eta = \frac{1}{2}\sqrt{\frac{\chi_1}{\chi_2}}, \alpha = \frac{1}{2}, \tilde{\alpha} = \frac{1}{2\sqrt{\chi_1\chi_2}}$, and $\chi = \sqrt{\chi_1\chi_2} \leq \chi_1$.*

Then, there exists a constant step size $\gamma > 0$ such that if:

- *(**strong-convexity**) the Assumption 3.4 is satisfied, then $\gamma \leq \frac{1}{16L(1+\chi)}$ and:*

$$\|\bar{x}_T - x^*\|^2 = \tilde{\mathcal{O}}\left(\|\bar{x}_0 - x^*\|^2 e^{-\frac{\mu T}{16L(1+\chi)}} + \frac{\sigma^2 + \zeta^2(1+\chi)}{\mu^2 T}\right)\,,$$

- *(**non-convexity**) the Assumption 3.5 is satisfied, then there is $c > 0$ which depends only on P, M from the assumptions such that $\gamma \leq \frac{c}{L(\chi+1)}$ and:*

$$\frac{1}{T}\int_0^T \|\nabla f(\bar{x}_t)\|^2\, dt = \mathcal{O}\left(\frac{L(1+\chi)}{T}(f(x_0) - f(x^*)) + \sqrt{\frac{L(f(x_0) - f(x^*))}{T}(\sigma^2 + (1+\chi)\xi^2)}\right)\,.$$

Also, the expected number of gradient step is nT and the number of communications is $\frac{\mathrm{Tr}(\Lambda)}{2}T$.

Remark 3.7. While our results are presented in a static setting for simplicity's sake, we emphasize that this model is actually amenable to both static and time-varying contexts, meaning that the probability for an edge to spike in our gossip model could either be time homogeneous or change during training. The time varying setting allows to recover the non-accelerated version, as generic acceleration is impossible for arbitrarily time-varying graphs [31].

Tab. 1 compares our convergence rates with concurrent works. Compared to every concurrent work, the bias term of $\mathbf{A}^2\mathbf{CiD}^2$ is smaller by a factor $\sqrt{\frac{\chi_1}{\chi_2}} \geq 1$ at least. Yet, as expected, in the non-accelerated setting, we would recover similar rates to those. Compared to [20], the variance terms held no variance reduction with the number of workers; however, this should not be an issue in a DL setting, where it is well-known that variance reduction technique degrade generalization during training [15]. Comparing directly the results of [2] is difficult as they only consider the asymptotic rate, even if the proof framework is similar to [27] and should thus lead to similar rates of convergence.

4 Numerical Experiments

Now, we experimentally compare $\mathbf{A}^2\mathbf{CiD}^2$ to a synchronous baseline All-Reduce SGD (AR-SGD, see [25]) and an *asynchronous baseline* using randomized pairwise communications (a variant of [27]) traditionally used in state-of-the-art decentralized asynchronous training of DNNs. In our case, the *asynchronous baseline* corresponds to the dynamic Eq. (4) without acceleration. Our approach is standard: we empirically study the decentralized training behavior of our asynchronous algorithm by training ResNets [17] for image recognition. Following [2], we pick a ResNet18 for CIFAR-10 [23] and ResNet50 for ImageNet [11]. To investigate how our method scales with the number of workers, we run multiple experiments using up to 64 NVIDIA A100 GPUs, with one worker per GPU.

4.1 Experimental Setup

Algorithm 1: This algorithm block describes our implementation of our Asynchronous algorithm with $\mathbf{A}^2\mathbf{CiD}^2$ on each local machine.

Input: On each machine $i \in \{1, ..., n\}$, gradient oracle ∇f_i, parameters $\eta, \alpha, \tilde{\alpha}, \gamma, T$.

Initialize on each machine $i \in \{1, ..., n\}$:
  Initialize $x^i, \tilde{x}^i \leftarrow x^i, t^i \leftarrow 0$ and **put** $x^i, \tilde{x}^i, t^i$ in shared memory;
  Synchronize the clocks of all machines ;
In parallel *on workers* $i \in \{1, ..., n\}$, **while** $t < T$, **continuously do:**
  In one thread *on worker* i **continuously do:**
    $t \leftarrow clock()$;
    Sample a batch of data via $\xi_i \sim \Xi$;
    $g_i \leftarrow \nabla F_i(x_i, \xi_i)$; `// Compute gradients`
    $\begin{pmatrix} x^i \\ \tilde{x}^i \end{pmatrix} \leftarrow \exp\left((t - t^i)\begin{pmatrix} -\eta & \eta \\ \eta & -\eta \end{pmatrix}\right)\begin{pmatrix} x^i \\ \tilde{x}^i \end{pmatrix}$;
    $x^i \leftarrow x^i - \gamma g_i$; `// Apply` $\mathtt{A^2CiD^2}$
    $\tilde{x}^i \leftarrow \tilde{x}^i - \gamma g_i$; `// Take the grad step`
    $t^i \leftarrow t$;
  In one thread *on worker* i **continuously do:**
    $t \leftarrow clock()$;
    Find available worker j ; `// Synchronize workers` i `and` j
    $m_{ij} \leftarrow (x^i - x^j)$; `// Send` x^i `to` j `and receive` x^j `from` j
    $\begin{pmatrix} x^i \\ \tilde{x}^i \end{pmatrix} \leftarrow \exp\left((t - t^i)\begin{pmatrix} -\eta & \eta \\ \eta & -\eta \end{pmatrix}\right)\begin{pmatrix} x^i \\ \tilde{x}^i \end{pmatrix}$; `// Apply` $\mathtt{A^2CiD^2}$
    $x^i \leftarrow x^i - \alpha m_{ij}$; `// p2p averaging`
    $\tilde{x}^i \leftarrow \tilde{x}^i - \tilde{\alpha} m_{ij}$;
    $t^i \leftarrow t$;
return $(x_T^i)_{1 \leq i \leq n}$.

Hyper-parameters. Training a DNN using multiple workers on a cluster requires several adaptation compared to the standard setting. As the effective batch-size grows linearly with the

Table 2: Training times on CIFAR10 ($\pm$ 6s).

	n	4	8	16	32	64
Ours	t (min)	21.4	10.8	5.7	3.2	1.9
AR	t (min)	21.9	11.1	6.6	3.2	1.8

number of workers n, we use the learning-rate schedule for large batch training of [16] in all our experiments. Following [29], we fixed the local batch size to 128 on both CIFAR-10 and ImageNet. Our goal being to divide the compute load between the n workers, all methods access the same total amount of data samples, regardless of the number of local steps. On CIFAR-10 and ImageNet, this number is set to 300 and 90 epochs respectively, following standard practice [22]. To circumvent the fuzziness of the notion of epoch in the asynchronous decentralized setting, we do not "split the dataset and re-shuffle it among workers at each epoch" as done with our standard All-Reduce baseline. Rather, we give access to the whole dataset to all workers, each one shuffling it with a different random seed. We use SGD with a base learning rate of 0.1, a momentum value set at 0.9 and 5×10^{-4} for weight decay. As advocated in [16], we do not apply weight decay on the learnable batch-norm coefficients. For ImageNet training with the SGD baseline, we decay the learning-rate

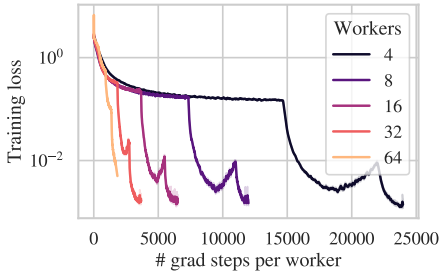


Figure 2: Training loss for CIFAR10 dataset with minibatch size 128, for the complete graph, without $\mathbf{A}^2\mathbf{CiD}^2$. As the number of worker increases, the loss degrades, especially for $n = 64$.

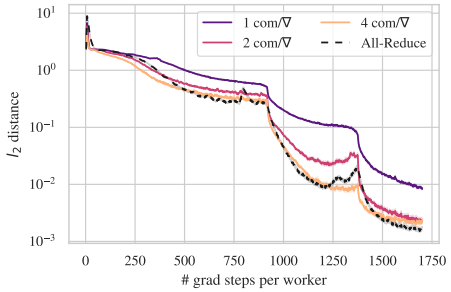


Figure 3: Training loss for CIFAR10 dataset with minibatch size 128, for the complete graph of size $n = 64$, without $\mathbf{A}^2\mathbf{CiD}^2$. As the rate of communication increases, the gap with All-Reduce decreases. With 2 com/∇, a test accuracy of 94.6 ± 0.04 is reached.

by a factor of 10 at epochs 30, 60, 80 (epochs 50, 75 for CIFAR-10), and apply an analogous decay schedule with our asynchronous decentralized methods. All of our neural networks parameters are initialized with the default Pytorch settings, and one All-Reduce averaging is performed before and after the training to ensure consensus at initialization. For our continuous momentum, we also need to set the parameters $\eta, \tilde{\alpha}$. For all our experiments, we use the values given by Prop. 3.6. As advocated, the *asynchronous baseline* correspond to the setting without acceleration, i.e. with $\eta = 0$ and $\alpha = \tilde{\alpha} = \frac{1}{2}$, whereas using $\mathbf{A}^2\mathbf{CiD}^2$ leads to consider $\eta = \frac{1}{2}\sqrt{\frac{\chi_1}{\chi_2}}, \alpha = \frac{1}{2}, \tilde{\alpha} = \frac{1}{2\sqrt{\chi_1\chi_2}}$, where χ_1, χ_2 are computed for the ring graph of corresponding size n and a communication ratio of 1 (*i.e.*, with edges weighted with 0.5). Surprisingly, these values also work for the complete graph, as we will see next.

Table 3: Accuracy of our method on CIFAR10 for a 128 batchsize with an equal number of pairwise communications and gradient computations per worker. We compared a vanilla asynchronous pairwise gossip approach with and without $\mathbf{A}^2\mathbf{CiD}^2$, demonstrating the improvement of our method.

#Workers	4	8	16	32	64
AR-SGD baseline	94.5±0.1	94.4±0.1	94.5±0.2	93.7±0.3	92.8±0.2
Complete graph					
Async. baseline	94.9±0.1	94.9±0.07	94.9±0.04	94.7±0.02	93.4±0.2
Ring graph					
Async. baseline	**95.0**±0.06	**95.0**±0.01	**95.1**±0.07	94.4±0.04	91.9±0.10
$\mathbf{A}^2\mathbf{CiD}^2$	94.9±0.02	**95.0**±0.1	95.0±0.2	**95.0**±0.08	**92.9**±0.2

Practical implementation of the dynamic. The dynamic studied in Eq. (4) is a model displaying many of the properties sought after in practice. In our implementation, described in Algo. 1, each worker i has indeed two independent processes and the DNN parameters $\{x^i, \tilde{x}^i\}$ are locally stored such that both processes can update them at any time. One process continuously performs gradient steps, while the other updates $\{x^i, \tilde{x}^i\}$ via peer-to-peer (p2p) averaging. The gradient process maximizes its throughput by computing forward and backward passes back to back. Ideally, we would also like to communicate as much as authorized by the hardware to reduce the idle time of the communication process, *i.e.* begin a pairwise averaging as soon as 2 neighboring workers in the graph finish their previous communications. This can be done in different ways, *e.g.* by pinging each others at high frequency, or by having a fast access to a shared memory to synchronize pairs of workers. Thus, knowing which edge will spike at what time is largely unpredictable in advance, supporting our model of communications using random processes. However, efficiently implementing those ideas is technically challenging, so we simulate this behaviour by performing pairwise averaging in a pseudo-random manner among the edges of the complete graph by making all workers share a common random seed. This allows to synchronize the communication schedule between workers and avoid deadlocks, leading to the typical timings of Tab. 2. To simulate a ping procedure between pair of workers, we make sure that the pair of nodes which communicate draw a bipartite graph at all times, as done in [27]. Contrary to All-Reduce based methods that require an increasing number of communications with the growing number of workers, inevitably leading to an increasing time between two rounds of computations, we study the case where each worker has a fixed ratio of p2p averaging with respect to its number of computed gradients. Thus, to control this ratio, we either put a barrier between the two processes, or we wait a random time before the next communication by drawing from an exponential random variable. Note that this does not guarantee any specific ordering in the evaluation of the subsequent steps of communications or computations. The parameter of this random variable is adjusted with the measured running average of the duration of the previous gradient steps. This running average is also used to normalize times in $\mathbf{A^2CiD^2}$ as we integrate the continuous dynamic in real time, as seen in Algo. 1.

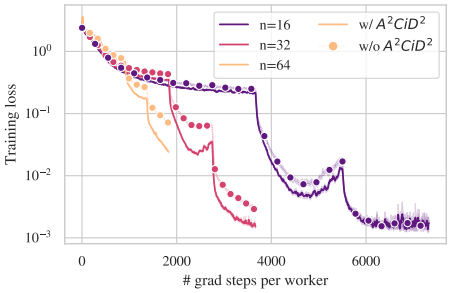


Figure 4: Training loss for CIFAR10 dataset using a minibatch size of 128. We display the training loss for various number of workers (up to 64), using $\mathbf{A^2CiD^2}$, both for the challenging ring graph.

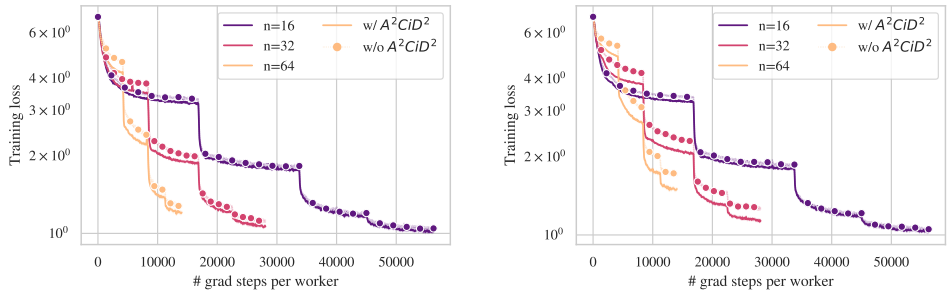
4.2 Evaluation on large scale datasets

CIFAR10. This simple benchmark allows to understand the benefits of our method in a well-controlled environment. Tab. 3 reports our numerical accuracy on the test set of CIFAR10, with a standard deviation calculated over 3 runs. Two scenarios are considered: a complete and a ring graph. In Fig. 2, we observe that with the asynchronous baseline on the complete graph, the more workers, the more the training loss degrades. Fig. 3 hints that it is in part due to an insufficient communication rate, as increasing it allows to lower the loss and close the gap with the All-Reduce baseline. However, this is not the only causative factor as Tab. 3 indicates that accuracy generally degrades as the number of workers increases even for AR-SGD, which is expected for large batch sizes. Surprisingly, even with a worse training loss for $n = 64$, the asynchronous baseline still leads to better generalization than AR-SGD, and consistently improves the test accuracy across all tested values of n. The communication rate being identified as a critical factor at large scale, we tested our continuous momentum on the ring graph, each worker performing one p2p averaging for each gradient step. Fig. 4 shows that incorporating $\mathbf{A^2CiD^2}$ leads

Table 4: Accuracy of our method on the ImageNet dataset for a batchsize of 128. We compared a vanilla asynchronous pairwise gossip approach with and without $\mathbf{A^2CiD^2}$, demonstrating the improvement of our method. We also varied the communication rate to study its impact on $\mathbf{A^2CiD^2}$.

#Workers	#com/#grad	16	32	64
AR-SGD baseline	-	75.5	75.2	74.5
Complete graph				
Async. baseline	1	74.6	73.8	71.3
$\mathbf{A^2CiD^2}$	1	75.1	**74.5**	72.6
$\mathbf{A^2CiD^2}$	2	**75.2**	74.4	**74.3**
Ring graph				
Async. baseline	1	**74.8**	71.6	64.1
$\mathbf{A^2CiD^2}$	1	74.7	**73.4**	**68.0**
Async. baseline	2	74.8	73.7	68.2
$\mathbf{A^2CiD^2}$	2	**75.3**	**74.4**	**71.4**

to a significantly better training dynamic for a large number of workers, which translates into better performances at test time as shown in Tab. 3.



(a) (b)

Figure 5: Training loss for ImageNet using 128 batch size, with an equal number of communications and computations per worker. We display the training loss for various number of workers (up to 64), using $\mathbf{A^2CiD^2}$, both for the complete graph (**left**) and challenging ring graph (**right**).

ImageNet. For validating our method in a real-life environment, we consider the large-scale ImageNet dataset. Tab. 4 reports our accuracy for the complete and ring graphs. Surprisingly, $\mathbf{A^2CiD^2}$ improves even in the case of the complete graph (as $\chi_1 = \chi_2$), which is not predicted by our theory. The case of the ring graph is much more challenging: for $n = 64$ workers, the accuracy drops by 10% compared to the reference baseline given by the AR-SGD. Systematically, with $\mathbf{A^2CiD^2}$, the finally accuracy increases: up to 4% absolute percent in the difficult $n = 64$ setting. This is corroborated by the Fig. 5, which indicates that incorporating $\mathbf{A^2CiD^2}$ significantly improves the training dynamic on ImageNet. However, for reducing the gap with the AR-SGD baseline, it will be necessary to increase the communication rate as discussed next.

Consensus improvement. The bottom of Tab. 4, as well as Fig. 6 study the virtual acceleration thanks to $\mathbf{A^2CiD^2}$. Note that the very first iterations ($\sim$100 batches) of the algorithm lead to a high consensus distance, likely because there is a warmup time to find a consensus. Not only increasing communications combined with $\mathbf{A^2CiD^2}$ allow to obtain competitive performance, but Fig. 1 shows that doubling the rate of communication has an identical effect on the training loss than adding $\mathbf{A^2CiD^2}$. This is verified in Fig. 6 by tracking the consensus distance between workers: $\mathbf{A^2CiD^2}$ significantly reduces it, which validates the results of Sec. 3.3.

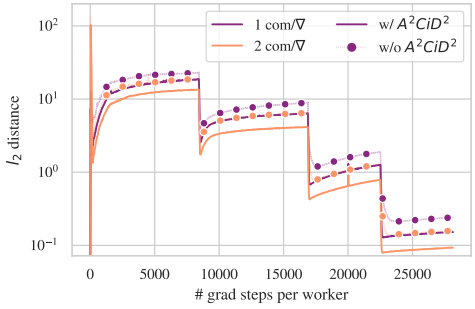


Figure 6: Comparison of consensus distances when $\mathbf{A^2CiD^2}$ is applied versus doubling the rate of communications: applying $\mathbf{A^2CiD^2}$ has the same effect as doubling communications.

5 Conclusion

In this work, we confirmed that increasing the communication rate is a key performance factor to successfully train DNNs at large scale with decentralized asynchronous methods. We introduced $\mathbf{A^2CiD^2}$, a continuous momentum which only adds a minor memory overhead while allowing to mitigate this need. We demonstrated, both theoretically and empirically, that $\mathbf{A^2CiD^2}$ substantially improves performances, especially on challenging network topologies. As we only focused on the cluster setting, in a future work, we would like to extend our study to more heterogeneous environments and refine our implementation to better translate the theoretical gains into practical speedups. To this end, we will open source our code to leverage the interest of the community.

Acknowledgements

EO, AN, and EB's work was supported by Project ANR-21-CE23-0030 ADONIS and EMERG-ADONIS from Alliance SU. This work was granted access to the HPC/AI resources of IDRIS under the allocation AD011013743 made by GENCI. EB and AN acknowledge funding and support from NSERC Discovery Grant RGPIN- 2021-04104 and resources from Compute Canada and Calcul Quebec. In addition, the authors would like to thank Olexa Bilaniuk for his helpful discussions.

References

[1] L. Arnold. Stochastic differential equations. *New York*, 1974.

[2] M. Assran, N. Loizou, N. Ballas, and M. Rabbat. Stochastic gradient push for distributed deep learning. In *International Conference on Machine Learning*, pages 344–353. PMLR, 2019.

[3] E. Belilovsky, M. Eickenberg, and E. Oyallon. Decoupled greedy learning of cnns. In *International Conference on Machine Learning*, pages 736–745. PMLR, 2020.

[4] E. Belilovsky, L. Leconte, L. Caccia, M. Eickenberg, and E. Oyallon. Decoupled greedy learning of cnns for synchronous and asynchronous distributed learning. *arXiv preprint arXiv:2106.06401*, 2021.

[5] M. Blot, D. Picard, M. Cord, and N. Thome. Gossip training for deep learning. In *Advances in Neural Information Processing Systems*, volume 30, 2016.

[6] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE Transactions on Information Theory*, 52(6):2508–2530, 2006.

[7] J. Chen, X. Pan, R. Monga, S. Bengio, and R. Jozefowicz. Revisiting distributed synchronous sgd. *arXiv preprint arXiv:1604.00981*, 2016.

[8] J. Daily, A. Vishnu, C. Siegel, T. Warfel, and V. Amatya. Gossipgrad: Scalable deep learning using gossip communication based asynchronous gradient descent, 2018.

[9] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, et al. Large scale distributed deep networks. *Advances in neural information processing systems*, 25, 2012.

[10] A. Defazio, F. Bach, and S. Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in neural information processing systems*, 27, 2014.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[12] M. Even, R. Berthier, F. Bach, N. Flammarion, H. Hendrikx, P. Gaillard, L. Massoulié, and A. Taylor. A continuized view on nesterov acceleration for stochastic gradient descent and randomized gossip. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[13] M. Even, H. Hendrikx, and L. Massoulie. Decentralized optimization with heterogeneous delays: a continuous-time approach. *arXiv preprint arXiv:2106.03585*, 2021.

[14] A. Ghosh, S. Boyd, and A. Saberi. Minimizing effective resistance of a graph. *SIAM Review*, 50(1):37–66, 2008.

[15] R. M. Gower, M. Schmidt, F. Bach, and P. Richtárik. Variance-reduced methods for machine learning. *Proceedings of the IEEE*, 108(11):1968–1983, 2020.

[16] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015.

[18] H. Hendrikx, F. Bach, and L. Massoulié. An accelerated decentralized stochastic proximal algorithm for finite sums. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alche-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[19] A. Koloskova, T. Lin, and S. U. Stich. An improved analysis of gradient tracking for decentralized machine learning. *Advances in Neural Information Processing Systems*, 34:11422–11435, 2021.

[20] A. Koloskova, T. Lin, S. U. Stich, and M. Jaggi. Decentralized deep learning with arbitrary communication compression. *arXiv preprint arXiv:1907.09356*, 2019.

[21] A. Koloskova, N. Loizou, S. Boreiri, M. Jaggi, and S. Stich. A unified theory of decentralized sgd with changing topology and local updates. In *International Conference on Machine Learning*, pages 5381–5393. PMLR, 2020.

[22] L. Kong, T. Lin, A. Koloskova, M. Jaggi, and S. Stich. Consensus control for decentralized deep learning. In *International Conference on Machine Learning*, pages 5686–5696. PMLR, 2021.

[23] A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, may 2017.

[25] S. Li and T. Hoefler. Near-optimal sparse allreduce for distributed deep learning. In *Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 135–149, 2022.

[26] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. *Advances in neural information processing systems*, 30, 2017.

[27] X. Lian, W. Zhang, C. Zhang, and J. Liu. Asynchronous decentralized parallel stochastic gradient descent. In *International Conference on Machine Learning*, pages 3043–3052. PMLR, 2018.

[28] T. Lin, S. P. Karimireddy, S. U. Stich, and M. Jaggi. Quasi-global momentum: Accelerating decentralized deep learning on heterogeneous data. *arXiv preprint arXiv:2102.04761*, 2021.

[29] T. Lin, S. U. Stich, K. K. Patel, and M. Jaggi. Don't use large mini-batches, use local sgd. In *International Conference on Learning Representations*, 2020.

[30] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[31] D. Metelev, A. Rogozin, D. Kovalev, and A. Gasnikov. Is consensus acceleration possible in decentralized optimization over slowly time-varying networks? *arXiv preprint arXiv:2301.11817*, 2023.

[32] A. Nabli and E. Oyallon. Dadao: Decoupled accelerated decentralized asynchronous optimization. *arXiv preprint arXiv:2208.00779*, 2023.

[33] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alche-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[34] M. Ryabinin, E. Gorbunov, V. Plokhotnyuk, and G. Pekhimenko. Moshpit sgd: Communication-efficient decentralized training on heterogeneous unreliable devices. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 18195–18211. Curran Associates, Inc., 2021.

[35] K. Scaman, F. Bach, S. Bubeck, Y. T. Lee, and L. Massoulié. Optimal algorithms for smooth and strongly convex distributed optimization in networks. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3027–3036. PMLR, 06–11 Aug 2017.

[36] A. Sergeev and M. D. Balso. Horovod: fast and easy distributed deep learning in TensorFlow. *arXiv preprint arXiv:1802.05799*, 2018.

[37] S. U. Stich. Unified optimal analysis of the (stochastic) gradient method, 2019.

[38] T. Vogels, L. He, A. Koloskova, S. P. Karimireddy, T. Lin, S. U. Stich, and M. Jaggi. Relaysum for decentralized deep learning on heterogeneous data. *Advances in Neural Information Processing Systems*, 34:28004–28015, 2021.

[39] Y. Yakimenka, C.-W. Weng, H.-Y. Lin, E. Rosnes, and J. Kliewer. Straggler-resilient differentially-private decentralized learning. In *2022 IEEE Information Theory Workshop (ITW)*, pages 708–713. IEEE, 2022.

[40] B. Ying, K. Yuan, Y. Chen, H. Hu, P. Pan, and W. Yin. Exponential graph is provably efficient for decentralized deep training. *Advances in Neural Information Processing Systems*, 34:13975–13987, 2021.

[41] B. Ying, K. Yuan, H. Hu, Y. Chen, and W. Yin. Bluefog: Make decentralized algorithms practical for optimization and deep learning. *arXiv preprint arXiv:2111.04287*, 2021.

[42] B. Ying, K. Yuan, H. Hu, Y. Chen, and W. Yin. Bluefog: Make decentralized algorithms practical for optimization and deep learning. 2021.

[43] Y. You, I. Gitman, and B. Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.

[44] B. Yuan, Y. He, J. Davis, T. Zhang, T. Dao, B. Chen, P. S. Liang, C. Re, and C. Zhang. Decentralized training of foundation models in heterogeneous environments. *Advances in Neural Information Processing Systems*, 35:25464–25477, 2022.

[45] S. Zhang, A. E. Choromanska, and Y. LeCun. Deep learning with elastic averaging sgd. 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.

Appendix

5.1 Proof of the main result of this paper

We remind the main proposition:

Proposition 5.1 (Convergence guarantees.)**.** *Assume that $\{x_t, \tilde{x}_t\}$ follow the dynamic Eq. 4 and that Assumption 3.2-3.3 are satisfied. Assume that $\mathbf{1}\bar{x}_0 = x_0 = \tilde{x}_0$ and let T the total running time. Then:*

- ***Non-accelerated setting**, we pick $\eta = 0, \alpha = \tilde{\alpha} = \frac{1}{2}$ and set $\chi = \chi_1$,*
- ***Acceleration (A^{2}CiD^2)**, we set $\eta = \frac{1}{2}\sqrt{\frac{\chi_1}{\chi_2}}, \alpha = \frac{1}{2}, \tilde{\alpha} = \frac{1}{2\sqrt{\chi_1\chi_2}}$, and $\chi = \sqrt{\chi_1\chi_2} \leq \chi_1$.*

Then, there exists a constant step size $\gamma > 0$ such that if:

- *(**strong-convexity**) the Assumption 3.4 is satisfied, then $\gamma \leq \frac{1}{16L(1+\chi)}$ and[1]:*

$$\|\bar{x}_T - x^*\|^2 = \tilde{\mathcal{O}}\left(\|\bar{x}_0 - x^*\|^2 e^{-\frac{\mu T}{16L(1+\chi)}} + \frac{\sigma^2 + \zeta^2(1+\chi)}{\mu^2 T}\right),$$

- *(**non-convexity**) the Assumption 3.5 is satisfied, then there is $c > 0$ which depends only on P, M from the assumptions such that $\gamma \leq \frac{c}{L(\chi+1)}$ and:*

$$\frac{1}{T}\int_0^T \|\nabla f(\bar{x}_t)\|^2 \, dt = \mathcal{O}\left(\frac{L(1+\chi)}{T}(f(x_0) - f(x^*)) + \sqrt{\frac{L(f(x_0) - f(x^*))}{T}(\sigma^2 + (1+\chi)\xi^2)}\right).$$

Also, the expected number of gradient step is nT and the number of communications is $\frac{\mathrm{Tr}(\Lambda)}{2}T$.

Proof. The core of the proof is to introduce the appropriate Lyapunov potentials $\phi_i(t, X)$, where X could be x or $\{x, \tilde{x}\}$. Then, we will note that by Ito's lemma, given all the functions are smooth, we obtain (in a similar fashion to [12, 32]), where X_t^+ corresponds to a Poisson update and M the momentum matrix:

$$\phi_i(T, X_T) - \phi_i(0, X_0) = \int_0^T \partial_t \phi_i(t, X_t) + \langle MX_t, \partial_X \phi_i(t, X_t)\rangle + (\phi_i(t, X_t^+) - \phi_i(t, X_t))\, dt + M_t\,,$$

where M_t is a martingale. We remind that:

$$\int_0^t e^{\alpha u}\, du = \frac{1}{\alpha}(e^{\alpha t} - 1) \tag{6}$$

We now present our choice of potential for each cases:

- For the convex case in the non-accelerated setting, we introduce:

$$\phi_1(t, x) \triangleq A_t\|\bar{x} - x^*\|^2 + B_t\|\pi x\|^2.$$

- For the convex case with **A^{2}CiD^2**, we introduce:

$$\phi_2(t, x, \tilde{x}) \triangleq A_t\|\bar{x} - x^*\|^2 + B_t\|\pi x\|^2 + \tilde{B}_t\|\tilde{x}\|_{\Lambda^+}.$$

- For the non-convex case in the non-accelerated setting, we introduce:

$$\phi_3(t, x) \triangleq A_t d_f(\bar{x}, x^*) + B_t\|\pi x\|^2.$$

- For the non-convex case with **A^{2}CiD^2**, we introduce:

$$\phi_4(t, x) \triangleq A_t d_f(\bar{x}, x^*) + B_t\|\pi x\|^2 + \tilde{B}_t\|x\|^2_{\Lambda^+}.$$

[1] $\tilde{\mathcal{O}}$-notation hides constants and polylogarithmic factors.

Notations: We write πx the projection so that $\|\pi x\|^2 = \sum_{i=1}^n \|x_i - \bar{x}\|^2$, the Bregman divergence is defined with $d_f(x, y) = f(x) - f(y) - \langle \nabla f(y), x - y \rangle$, and we introduce $F(x)_i \triangleq f_i(x_i)$ and $\tilde{F}(x, \xi)_i \triangleq F_i(x_i, \xi_i)$.

5.1.1 Some useful upper-bounds

As the same terms appear in several potential, we now prepare some intermediary results which will be helpful for the proofs:

- Let's set:

$$\Delta x \triangleq \sum_{i=1}^n \|\overline{x - \gamma \nabla F_i(x_i)} - x^*\|^2 - \|\bar{x} - x^*\|^2$$

We note that, using Assumption 3.4:

$$\begin{aligned}
\mathbb{E}_{\xi_1,\dots,\xi_n}[\Delta x] &= \sum_{i=1}^n -\frac{2\gamma}{n}\langle \bar{x} - x^*, \nabla f_i(x_i)\rangle + \frac{\gamma^2}{n^2}\mathbb{E}_{\xi_i}[\|\nabla F_i(x_i, \xi_i)\|^2] && (7)\\
&= \sum_{i=1}^n -\frac{2\gamma}{n}\langle \bar{x} - x^*, \nabla f_i(x_i) - \nabla f_i(x^*)\rangle + \frac{\gamma^2}{n^2}\mathbb{E}_{\xi_i}[\|\nabla F_i(x_i, \xi_i)\|^2] && (8)\\
&\leq \frac{\gamma^2}{n}\sigma^2 + \sum_{i=1}^n -\frac{2\gamma}{n}\langle \bar{x} - x^*, \nabla f_i(x_i) - \nabla f_i(x^*)\rangle + \frac{\gamma^2}{n^2}\|\nabla f_i(x_i)\|^2 && (9)\\
&= \frac{\gamma^2}{n}\sigma^2 + \sum_{i=1}^n -\frac{2\gamma}{n}(d_{f_i}(\bar{x}, x^*) + d_{f_i}(x^*, x_i) - d_{f_i}(\bar{x}, x_i)) + \frac{\gamma^2}{n^2}\|\nabla f_i(x_i)\|^2 && (10)\\
&\leq \frac{\gamma^2}{n}\sigma^2 + \sum_{i=1}^n -\frac{2\gamma}{n}(d_{f_i}(\bar{x}, x^*) + d_{f_i}(x^*, x_i) - d_{f_i}(\bar{x}, x_i))\\
&\quad + \frac{2\gamma^2}{n^2}\|\nabla f_i(x^*) - \nabla f_i(x_i)\|^2 + \frac{2\gamma^2}{n^2}\|\nabla f_i(x^*)\|^2 && (11)\\
&\leq \frac{\gamma^2}{n}\sigma^2 + \frac{2\gamma^2}{n}\zeta^2 + \sum_{i=1}^n -\frac{2\gamma}{n}(d_{f_i}(\bar{x}, x^*) + d_{f_i}(x^*, x_i) - d_{f_i}(\bar{x}, x)) + \frac{4L\gamma^2}{n^2}d_{f_i}(x^*, x_i) && (12)\\
&\leq \frac{\gamma^2\sigma^2}{n} + \frac{2\gamma^2}{n}\zeta^2 - \gamma\mu\|\bar{x} - x^*\|^2 + \frac{L\gamma}{n}\|\pi x\|^2 + \sum_{i=1}^n(-\frac{2\gamma}{n} + \frac{4L\gamma^2}{n^2})d_{f_i}(x^*, x_i) && (13)
\end{aligned}$$

- Next, we set:

$$\Delta f \triangleq \sum_{i=1}^n d_f(\bar{x} - \gamma\frac{1}{n}\nabla F_i(x_i, \xi_i), x^*) - d_f(\bar{x}, x^*) \tag{14}$$

First, it is useful to note that under Assumption 3.5:

$$\mathbb{E}_{\xi_1,\dots,\xi_n}\|\nabla \tilde{F}(x)\|^2 \leq n\sigma^2 + (1 + M)\sum_{i=1}^n \|\nabla f_i(x_i)\|^2 \tag{15}$$

Then, we get:

$$\mathbb{E}_{\xi_1,\dots,\xi_n}[\Delta f] = \sum_{i=1}^{n} \mathbb{E}[d_f(\bar{x} - \gamma\frac{1}{n}\nabla F_i(x_i,\xi_i),\bar{x})] - \frac{\gamma}{n}\langle \nabla f_i(x_i), \nabla f(\bar{x})\rangle \tag{16}$$

$$\leq \sum_{i=1}^{n} \frac{1}{2n^2} L\gamma^2 \mathbb{E}[\|\nabla F_i(x_i,\xi_i)\|^2] - \frac{\gamma}{n}\langle \nabla f_i(x_i), \nabla f(\bar{x})\rangle \tag{17}$$

$$\leq \frac{L\gamma^2}{2n}\sigma^2 - \gamma\|\nabla f(\bar{x})\|^2 + \sum_{i=1}^{n} \frac{M+1}{2n^2} L\gamma^2 \|\nabla f_i(x_i)\|^2 - \sum_{i=1}^{n} \frac{\gamma}{n}\langle \nabla f_i(x_i) - \nabla f_i(\bar{x}), \nabla f(\bar{x})\rangle \tag{18}$$

$$\leq \frac{L\gamma^2}{2n}\sigma^2 + \frac{\gamma}{2n}L^2\|\pi x\|^2 - \frac{\gamma}{2}\|\nabla f(\bar{x})\|^2 + \sum_{i=1}^{n} \frac{M+1}{2n^2} L\gamma^2 \|\nabla f_i(x_i)\|^2 \tag{19}$$

As we also have:

$$\sum_{i=1}^{n} \|\nabla f_i(x_i)\|^2 \leq \sum_{i=1}^{n} 3(\|\nabla f_i(x_i) - \nabla f_i(\bar{x})\|^2 + \|\nabla f_i(\bar{x}) - \nabla f(\bar{x})\|^2 + \|\nabla f(\bar{x})\|^2) \tag{20}$$

$$\leq 3L^2\|\pi x\|^2 + 3n\zeta^2 + 3n(1+P)\|\nabla f(\bar{x})\|^2 \tag{21}$$

We get in the end:

$$\mathbb{E}_{\xi_1,\dots,\xi_n}[\Delta f] \leq \frac{L\gamma^2}{2n}\sigma^2 + \frac{3(M+1)}{2n}\gamma^2 L\zeta^2 + (\frac{3(M+1)}{2n^2}L^3\gamma^2 + \frac{\gamma}{2n}L^2)\|\pi x\|^2 + (\frac{3(M+1)}{2n}(1+P)L\gamma^2 - \frac{\gamma}{2})\|\nabla f(\bar{x})\|^2 \tag{22}$$

Remark 5.2. As observed in (5), note that for both the terms $\|\bar{x} - x^*\|^2$ and $d_f(\bar{x}, x^*)$, as we are considering $\bar{x}$ and $\frac{1}{n}\mathbf{1}\mathbf{1}^\mathsf{T}\pi = 0$, Poisson updates from the communication process amount to zero. Moreover, as $\frac{1}{n}\mathbf{1}\mathbf{1}^\mathsf{T}(x - \tilde{x}) = 0$, the update from the momentum is also null for these terms.

- For the $\|\pi x\|^2$ term, we get from the Poisson updates (gradient and communication processes):

$$\mathbb{E}[\Delta_\pi] \triangleq -2\gamma\langle \pi x, \nabla F(x)\rangle + \gamma^2 \mathbb{E}_{\xi_1,\dots,\xi_n}\|\pi\nabla\tilde{F}(x)\|^2 - 2\alpha\langle x, \Lambda x\rangle + 2\alpha^2\|x\|_\Lambda^2 \tag{23}$$

$$\leq 4\chi_1\gamma^2\|\nabla F(x)\|^2 + (\frac{1}{4} - 2\alpha(1-\alpha))\|x\|_\Lambda^2 + \gamma^2\mathbb{E}_{\xi_1,\dots,\xi_n}\|\pi\nabla\tilde{F}(x)\|^2 \tag{24}$$

For $\alpha = \frac{1}{2}$, we get:

$$\mathbb{E}[\Delta_\pi] \leq 4\chi_1\gamma^2\|\nabla F(x)\|^2 - \frac{1}{4\chi_1}\|\pi x\|^2 + \gamma^2\mathbb{E}_{\xi_1,\dots,\xi_n}\|\pi\nabla\tilde{F}(x)\|^2 \tag{25}$$

For $\mathbf{A}^2\mathbf{CiD}^2$, we add the momentum term to define:

$$\Delta_\Lambda^1 \triangleq 2\eta\langle \tilde{x}, \pi x\rangle - 2\eta\|\pi x\|^2 - 2\gamma\langle \pi x, \nabla\tilde{F}(x)\rangle + \gamma^2\|\pi\nabla\tilde{F}(x)\|^2 - 2\alpha(1-\alpha)\|x\|_\Lambda \tag{26}$$

Which leads to:

$$\mathbb{E}[\Delta_\Lambda^1] \leq 2\eta\langle \tilde{x}, \pi x\rangle - \frac{3}{2}\eta\|\pi x\|^2 + \frac{2}{\eta}\gamma^2\|\pi\nabla F(x)\|^2 - 2\alpha(1-\alpha)\|x\|_\Lambda + \gamma^2\mathbb{E}_{\xi_1,\dots,\xi_n}\|\nabla\tilde{F}(x)\|^2 \tag{27}$$

- Finally, we study the $\|\tilde{x}\|^2_{\Lambda^+}$ term. From the Poisson updates and momentum, we get:

$$\Delta^2_\Lambda = 2\eta\langle x-\tilde{x},\tilde{x}\rangle_{\Lambda^+} - 2\gamma\langle\pi\tilde{x},\Lambda^+\nabla\tilde{F}(x)\rangle + \gamma^2\|\pi\nabla\tilde{F}(x)\|^2_{\Lambda^+} - 2\tilde{\alpha}\langle\pi x,\tilde{x}\rangle + \tilde{\alpha}^2\sum_{(i,j)\in\mathcal{E}}\lambda^{ij}\|(e_i-e_j)(e_i-e_j)^\mathsf{T}x\|^2_{\Lambda^+} \tag{28}$$

Noting that, for $(i,j)\in\mathcal{E}$:

$$\|(e_i-e_j)(e_i-e_j)^\mathsf{T}x\|^2_{\Lambda^+} = x^\mathsf{T}(e_i-e_j)(e_i-e_j)^\mathsf{T}\Lambda^+(e_i-e_j)(e_i-e_j)^\mathsf{T}x \tag{29}$$

$$\leq 2\chi_2 x^\mathsf{T}(e_i-e_j)(e_i-e_j)^\mathsf{T}x \tag{30}$$

leads to:

$$\mathbb{E}[\Delta^2_\Lambda] \leq \chi_1\eta\|\pi x\|^2 + (\frac{\eta}{2}-\eta)\|\tilde{x}\|^2_{\Lambda^+} + \frac{2}{\eta}\chi_1\gamma^2\|\pi\nabla F(x)\|^2 - 2\tilde{\alpha}\langle\pi x,\tilde{x}\rangle + 2\chi_2\tilde{\alpha}^2\|x\|_\Lambda + \chi_1\gamma^2\mathbb{E}_{\xi_1,\dots,\xi_n}\|\pi\nabla\tilde{F}(x)\|^2 \tag{31}$$

5.1.2 Resolution: putting everything together

In this part, we combine the terms for each potential. We remind that $\|\pi\nabla F(x)\|^2 \leq \sum_{i=1}^n\|\nabla f_i(x_i)\|^2$ and the fact that, with Assumption 3.4:

$$\sum_{i=1}^n\mathbb{E}_{\xi_i}[\|\nabla F_i(x_i,\xi_i)\|^2] = n\sigma^2 + \sum_{i=1}^n\|\nabla f_i(x_i)\|^2 \tag{32}$$

$$\leq n\sigma^2 + \sum_{i=1}^n 2\|\nabla f_i(x^*)-\nabla f_i(x_i)\|^2 + 2\|\nabla f_i(x^*)\|^2 \tag{33}$$

$$\leq n\sigma^2 + 2n\zeta^2 + 4L\sum_{i=1}^n d_{f_i}(x^*,x_i) \tag{34}$$

Convex case, non-accelerated. We remind that

$$\phi_1(t,x,\tilde{x}) = A_t\|\bar{x}-x^*\|^2 + B_t\|\pi x\|^2.$$

Then, using (13), (25) and defining $\Psi_1 \triangleq \partial_t\phi_1(t,X_t) + \mathbb{E}[A_t\Delta x + B_t\Delta_\pi]$, we have:

$$\Psi_1 \leq \|\bar{x}-x^*\|^2(A'_t - \mu\gamma A_t) \tag{35}$$

$$+ \|\pi x\|^2\left(B'_t + \frac{L\gamma}{n}A_t - \frac{1}{4\chi_1}B_t\right) \tag{36}$$

$$+ \sum_{i=1}^n d_{f_i}(x^*,x_i)\left(-\frac{2\gamma}{n}A_t + \frac{4L\gamma^2}{n^2}A_t + 4L\left(4\chi_1\gamma^2+\gamma^2\right)B_t\right) \tag{37}$$

$$+ \left(\frac{\gamma^2\sigma^2}{n} + \frac{2\gamma^2}{n}\zeta^2\right)A_t + \left(n\gamma^2\sigma^2 + 2n\zeta^2(4\chi_1\gamma^2+\gamma^2)\right)B_t \tag{38}$$

We pick $\alpha = \frac{1}{2}$, $B_t = \frac{1}{n}A_t$, with $A_t = e^{-rt}$ (we denote by r the rate of the exponentials A_t, B_t). Then (35), (36) imply:

$$r \leq \min(\mu\gamma, \frac{1}{4\chi_1} - L\gamma) \tag{39}$$

As we want (37) to be negative, we have:

$$-1 + \gamma\left(\frac{2L}{n} + 2L(4\chi_1+1)\right) \leq 0 \tag{40}$$

which leads to:

$$\gamma \leq \frac{1}{2L\left(\frac{1}{n}+4\chi_1+1\right)} \tag{41}$$

and taking $\gamma \leq \frac{1}{2}\frac{1}{2L(3+4\chi_1+1)} = \frac{1}{16L(1+\chi_1)}$ works. Now, as $\frac{1}{4\chi_1} - L\gamma \geq \frac{3}{16\chi_1}$ and $\mu\gamma \leq \frac{1}{16\chi_1}$, we pick $r = \mu\gamma$. As we have:

$$\mathbb{E}\left[\phi_1(T, x_T) - \phi_1(0, x_0)\right] = \int_0^T \left(A_t'\|\bar{x}_t - x^*\|^2 + B_t'\|\pi x_t\|^2 + A_t\mathbb{E}_{\xi_1,\dots,\xi_n}[\Delta x] + B_t\mathbb{E}[\Delta_\pi]\right) dt \tag{42}$$

using (38) and (6) leads to:

$$\mathbb{E}\|\bar{x}_t - x^*\|^2 \leq e^{-\gamma\mu t}\left(\|\bar{x}_0 - x^*\|^2 + \frac{1}{n}\|\pi x_0\|^2\right) + \frac{\gamma}{\mu}\left(\sigma^2(\frac{1}{n}+1) + 2\zeta^2(\frac{1}{n} + 4\chi_1 + 1)\right) \tag{43}$$

Convex case with $\mathbf{A^2CiD^2}$. We remind that

$$\phi_2(t, x, \tilde{x}) = A_t\|\bar{x} - x^*\|^2 + B_t\|\pi x\|^2 + \tilde{B}_t\|\tilde{x}\|_{\Lambda^+}.$$

Then, using (13), (27), (31) and defining $\Psi_2 \triangleq \partial_t\phi_2(t, X_t) + \mathbb{E}[A_t\Delta x + B_t\Delta_\Lambda^1 + \tilde{B}_t\Delta_\Lambda^2]$, we have:

$$\begin{aligned}
\Psi_2 \leq\ & \|\bar{x} - x^*\|^2 \left(A_t' - \mu\gamma A_t\right) && (44)\\
&+ \|\pi x\|^2 \left(B_t' + \frac{L\gamma}{n}A_t - \frac{3}{2}\eta B_t + \eta\chi_1\tilde{B}_t\right) && (45)\\
&+ \|\tilde{x}\|_{\Lambda^+}^2 \left(\tilde{B}_t' - \frac{\eta}{2}\tilde{B}_t\right) && (46)\\
&+ \|x\|_\Lambda^2 \left(2\tilde{\alpha}^2\chi_2\tilde{B}_t - 2\alpha(1-\alpha)B_t\right) && (47)\\
&+ \langle\tilde{x}, \pi x\rangle \left(2\eta B_t - 2\tilde{\alpha}\tilde{B}_t\right) && (48)\\
&+ \sum_{i=1}^n d_{f_i}(x^*, x_i)\left(-\frac{2\gamma}{n}A_t + \frac{4L\gamma^2}{n^2}A_t + 4L\left(\frac{2\gamma^2}{\eta} + \gamma^2\right)(B_t + \chi_1\tilde{B}_t)\right) && (49)\\
&+ \left(\frac{\gamma^2\sigma^2}{n} + \frac{2\gamma^2}{n}\zeta^2\right)A_t + \left(n\gamma^2\sigma^2 + 2n\zeta^2(\gamma^2 + \frac{2\gamma^2}{\eta})\right)(B_t + \chi_1\tilde{B}_t) && (50)
\end{aligned}$$

Then, we assume $\alpha = \frac{1}{2}$, $\tilde{\alpha} = \frac{1}{2}\sqrt{\frac{\chi_1}{\chi_2}}$, $\eta = \frac{1}{2\sqrt{\chi_1\chi_2}}$, $B_t = \frac{1}{n}A_t$, $\tilde{B}_t = \frac{1}{\chi_1}B_t$, $A_t = e^{-rt}$ (we denote by r the rate of the exponentials $A_t, B_t, \tilde{B}_t$), which satisfies (47) and (48). Then (44), (45), (46) imply:

$$r \leq \min(\mu\gamma, \frac{\eta}{2}, \frac{\eta}{2} - L\gamma) = \min(\mu\gamma, \frac{\eta}{2} - L\gamma) \tag{51}$$

As we want (49) to be negative, we have:

$$-1 + \gamma\left(\frac{2L}{n} + 4L(\frac{2}{\eta} + 1)\right) \leq 0 \tag{52}$$

which leads to:

$$\gamma \leq \frac{1}{2L\left(\frac{1}{n} + \frac{4}{\eta} + 2\right)} \tag{53}$$

and taking $\gamma \leq \frac{1}{2L\left(6+\frac{4}{\eta}+2\right)} = \frac{1}{16L\left(1+\sqrt{\chi_1\chi_2}\right)}$ works. Now, we have:

$$\frac{\eta}{2} - L\gamma \geq \frac{1}{4\sqrt{\chi_1\chi_2}}\left(1 - \frac{4\sqrt{\chi_1\chi_2}}{16\left(1+\sqrt{\chi_1\chi_2}\right)}\right) \geq \frac{3}{16\sqrt{\chi_1\chi_2}} \tag{54}$$

As $\mu\gamma \leq \frac{\mu}{16L\left(1+\sqrt{\chi_1\chi_2}\right)} \leq \frac{1}{16\sqrt{\chi_1\chi_2}}$, taking $r = \mu\gamma$ works. Finally, using (50) and (6) leads to:

$$\mathbb{E}\|\bar{x}_t - x^*\|^2 \leq e^{-\gamma\mu t}\left(\|\bar{x}_0 - x^*\|^2 + \frac{2}{n}\|\pi x_0\|^2\right) + \frac{\gamma}{\mu}\left(\sigma^2(\frac{1}{n}+2) + 2\zeta^2(\frac{1}{n} + 8\sqrt{\chi_1\chi_2} + 2)\right) \tag{55}$$

Non-convex case, non-accelerated. We remind that:

$$\phi_3(t,x) = A_t d_f(\bar{x}, x^*) + B_t \|\pi x\|^2$$

Here, we pick $\alpha = \frac{1}{2}, A_t = 1, B_t = \frac{L}{n}A_t$. Thus, $A'_t = B'_t = 0$. Then, using (22), (25), (21) we obtain:

$$A_t\mathbb{E}[\Delta f] + B_t\mathbb{E}[\Delta_\pi]) \leq \|\nabla f(\bar{x})\|^2 \left(-\frac{\gamma}{2}A_t + \frac{3}{2n}L\gamma^2(M+1)(P+1)A_t + 3n\gamma^2(4\chi_1+M+1)(P+1)B_t\right) \tag{56}$$

$$+ \|\pi x\|^2 \left(\frac{L^2\gamma}{2n}(1+\frac{3}{n}(M+1)L\gamma)A_t + 3L^2\gamma^2(4\chi_1+M+1)B_t - \frac{1}{4\chi_1}B_t\right) \tag{57}$$

$$+ \gamma^2\left(\sigma^2 + 3(M+1)\zeta^2\right)\left(\frac{L}{2n}A_t + nB_t\right) + 12n\chi_1\gamma^2\zeta^2 B_t \tag{58}$$

Our goal is to use half of the negative term of (56) to cancel the positive ones, so that there remains at least $-\frac{\gamma}{4}A_t\|\nabla f(\bar{x})\|^2$ in the end. Thus, we want:

$$3L\gamma^2\left(\frac{(M+1)(P+1)}{2n} + (4\chi_1+M+1)(P+1)\right) \leq \frac{\gamma}{4} \tag{59}$$

and taking $\gamma \leq \frac{1}{48(M+1)(P+1)(1+\chi_1)}$ works. We verify that with γ defined as such, (57) is also negative. Finally, we upper bound (58) with $3L\gamma^2\left(\sigma^2 + 3(M+1+4\chi_1)\zeta^2\right)$. As we have:

$$\mathbb{E}[\phi_3(T,x_T) - \phi_3(0,x_0)] = \int_0^T (A'_t d_f(\bar{x}_t, x^*) + B'_t\|\pi x_t\|^2 + A_t\mathbb{E}[\Delta f] + B_t\mathbb{E}[\Delta_\pi])\,dt \tag{60}$$

we note that if $\gamma \leq \frac{c}{L(\chi_1+1)}$ for some constant $c > 0$ which depends on M, P, we will get:

$$\frac{\gamma}{4}\int_0^T \mathbb{E}[\|\nabla f(\bar{x}_t)\|^2]\,dt \leq \frac{L}{n}\|\pi x_0\|^2 + d_f(x_0,x^*) + \mathcal{O}\left(LT\gamma^2(\sigma^2 + (1+\chi_1)\zeta^2)\right) \tag{61}$$

which also writes:

$$\frac{1}{T}\int_0^T \mathbb{E}[\|\nabla f(\bar{x}_t)\|^2]\,dt \leq \frac{4}{\gamma T}(f(x_0) - f(x^*)) + \mathcal{O}\left(L\gamma(\sigma^2 + (1+\chi_1)\zeta^2)\right) \tag{62}$$

Non-convex case, with $\mathbf{A^2CiD^2}$. We have:

$$\phi_4(t,x) = A_t d_f(\bar{x}, x^*) + B_t\|\pi x\|^2 + \tilde{B}_t\|x\|^2_{\Lambda^+}$$

Here, we pick $\alpha = \frac{1}{2}, \eta = \frac{1}{2\sqrt{\chi_1\chi_2}}, A_t = 1, B_t = \frac{L}{n}A_t, B_t = \chi_1\tilde{B}_t$ and an identical reasoning to the convex setting allows to say we can find a constant $c > 0$ such that if $\gamma \leq \frac{c}{L(1+\sqrt{\chi_1\chi_2})}$, then:

$$\frac{1}{T}\int_0^T \mathbb{E}[\|\nabla f(\bar{x}_t)\|^2]\,dt = \mathcal{O}\left(\frac{1}{\gamma T}(f(x_0) - f(x^*)) + L\gamma(\sigma^2 + (1+\sqrt{\chi_1\chi_2})\zeta^2)\right) \tag{63}$$

5.1.3 Optimizing the step-size

In this part, we follow [37, 21] and optimize the step-size a posteriori. We set $\chi = \chi_1$ for the non-accelerated setting and $\chi = \sqrt{\chi_1\chi_2}$ with $\mathbf{A^2CiD^2}$.

Strongly-convex cases: From (43) and (55), we can write that, for $\gamma \leq \frac{1}{16L(1+\chi)}$ and initializing x_0 such that $\pi x_0 = 0$, we have:

$$\mathbb{E}\|\bar{x}_t - x^*\|^2 = \mathcal{O}\left(\|\bar{x}_0 - x^*\|^2 e^{-\gamma\mu t} + \frac{\gamma}{\mu}(\sigma^2 + \zeta^2(1+\chi))\right) \tag{64}$$

Then, taking the proof of [37] and adapting the threshold, we consider two cases (with $r_0 \triangleq \|\bar{x}_0 - x^*\|^2$):

- if $\frac{1}{16L(1+\chi)} \geq \frac{\log(\max\{2,\mu^2 r_0 T/\sigma^2\})}{\mu T}$, then we set $\gamma = \frac{\log(\max\{2,\mu^2 r_0 T/\sigma^2\})}{\mu T}$.

  In this case, (64) gives:

$$\mathbb{E}\|\bar{x}_T - x^*\|^2 = \tilde{\mathcal{O}}\left(\frac{1}{\mu^2 T}(\sigma^2 + \zeta^2(1+\chi))\right) \tag{65}$$

- if $\frac{1}{16L(1+\chi)} < \frac{\log(\max\{2,\mu^2 r_0 T/\sigma^2\})}{\mu T}$, then we set $\gamma = \frac{1}{16L(1+\chi)}$.

  Then, (64) gives:

$$\mathbb{E}\|\bar{x}_T - x^*\|^2 = \mathcal{O}\left(r_0 e^{-\frac{\mu T}{16L(1+\chi)}} + \frac{1}{\mu}\frac{1}{16L\,(1+\chi)}(\sigma^2 + \zeta^2(1+\chi))\right) \tag{66}$$

$$= \tilde{\mathcal{O}}\left(r_0 e^{-\frac{\mu T}{16L(1+\chi)}} + \frac{1}{\mu^2 T}(\sigma^2 + \zeta^2(1+\chi))\right) \tag{67}$$

Non-convex cases: From (62) and (63), we can write that, for some constant $c > 0$ depending on M, P such that $\gamma \leq \frac{c}{L(1+\chi)}$, we have:

$$\frac{1}{T}\int_0^T \mathbb{E}[\|\nabla f(\bar{x}_t)\|^2]\,dt = \mathcal{O}\left(\frac{1}{\gamma T}(f(x_0) - f(x^*)) + L\gamma(\sigma^2 + (1+\chi)\zeta^2)\right) \tag{68}$$

Then, taking the proof of Lemma 17 in [21] and adapting the threshold, we consider two cases (with $f_0 \triangleq f(x_0) - f(x^*)$):

- if $\frac{c}{L(1+\chi)} < \left(\frac{f_0}{TL(\sigma^2+(1+\chi)\zeta^2)}\right)^{1/2}$, then we take $\gamma = \frac{c}{L(1+\chi)}$, giving:

$$\frac{1}{T}\int_0^T \mathbb{E}[\|\nabla f(\bar{x}_t)\|^2]\,dt = \mathcal{O}\left(\frac{L(1+\chi)}{T}f_0 + L\left(\frac{f_0}{TL(\sigma^2+(1+\chi)\zeta^2)}\right)^{1/2}(\sigma^2 + (1+\chi)\zeta^2)\right) \tag{69}$$

$$= \mathcal{O}\left(\frac{L(1+\chi)}{T}f_0 + \sqrt{\frac{Lf_0}{T}(\sigma^2 + (1+\chi)\zeta^2)}\right) \tag{70}$$

- if $\frac{c}{L(1+\chi)} \geq \left(\frac{f_0}{TL(\sigma^2+(1+\chi)\zeta^2)}\right)^{1/2}$, then we take $\gamma = \left(\frac{f_0}{TL(\sigma^2+(1+\chi)\zeta^2)}\right)^{1/2}$, giving:

$$\frac{1}{T}\int_0^T \mathbb{E}[\|\nabla f(\bar{x}_t)\|^2]\,dt = \mathcal{O}\left(\frac{f_0}{T}\left(\frac{TL(\sigma^2+(1+\chi)\zeta^2)}{f_0}\right)^{1/2} + \sqrt{\frac{Lf_0}{T}(\sigma^2 + (1+\chi)\zeta^2)}\right) \tag{71}$$

$$= \mathcal{O}\left(\sqrt{\frac{Lf_0}{T}(\sigma^2 + (1+\chi)\zeta^2)}\right) \tag{72}$$

□