Noname manuscript No.
(will be inserted by the editor)

Asynchronous Parallel Algorithms for Nonconvex Big-Data Optimization Part I: Model and Convergence

Loris Cannelli · Francisco Facchinei · Vyacheslav Kungurtsev · Gesualdo Scutari

January 17, 2017

Abstract We propose a novel asynchronous parallel algorithmic framework for the minimization of the sum of a smooth nonconvex function and a convex nonsmooth regularizer, subject to both convex and nonconvex constraints. The proposed framework hinges on successive convex approximation techniques and a novel probabilistic model that captures key elements of modern computational architectures and asynchronous implementations in a more faithful way than current state of the art models. Key features of the proposed framework are: i) it accommodates inconsistent read, meaning that components of the vector variables may be written by some cores while being simultaneously read by others; ii) it covers in a unified way several different specific solution methods, and iii) it accommodates a variety of possible parallel computing architectures. Almost sure convergence to stationary solutions is proved. Numerical results, reported in the companion paper [5], on both convex and nonconvex problems show our method can consistently outperform existing parallel asynchronous algorithms.

Keywords Asynchronous algorithms · big-data · inconsistent read · nonconvex constrained optimization

1 Introduction

The computational demands associated with the gathering and storage of ever increasing amounts of data in many fields in the contemporary "big data" age

Loris Cannelli and Gesualdo Scutari, School of Industrial Engineering, Purdue University, USA E-mail: <lcannelli, gscutari>@purdue.edu. · Francisco Facchinei, Department of Computer, Control, and Management Engineering Antonio Ruberti, University of Rome La Sapienza, Roma, Italy E-mail: francisco.facchinei@uniroma1.it. · Vyacheslav Kungurtsev Dept. of Computer Science, Faculty of Electrical Engineering, Czech Technical University in Prague, Czech E-mail: vyacheslav.kungurtsev@fel.cvut.cz. The work of Cannelli and Scutari was supported by the USA NSF under Grants CIF 1564044, CCF 1632599 and CAREER Award No. 1555850, and the Office of Naval Research (ONR) Grant N00014-16-1-2244.

have challenged the optimization community to develop scalable algorithms capable of solving very large problems with comparatively limited hardware memory capacity and processor speed. Because of this, in the past few years the study of asynchronous parallel optimization methods has witnessed a surge of interest. Asynchronous parallel methods reduce the idle times of cores, mitigate communication and memory-access congestion, and make algorithms more fault-tolerant, and so are a natural answer to the challenges posed by data-intensive applications, especially when using modern high-performance computing architectures.

We consider asynchronous, parallel methods for the minimization of the sum of a smooth, possibly nonconvex function f and of a nonsmooth, convex one G:

$$\min_{\mathbf{x}} F(\mathbf{x}) \triangleq f(\mathbf{x}) + G(\mathbf{x}) \quad \mathbf{x} \in \mathcal{X} \tag{1}$$

where $\mathcal{X}$ is a closed, possibly nonconvex set with a Cartesian product structure and G is block-separable: $\mathcal{X} \triangleq \mathcal{X}_1 \times \cdots \mathcal{X}_N \subseteq \mathbb{R}^n$ and $G(\mathbf{x}) \triangleq \sum_{i=1}^N g_i(\mathbf{x}_i)$, with $\mathbf{x}_i \in \mathcal{X}_i \subseteq \mathbb{R}^{n_i}$, for all $i = 1, 2, \ldots, N$. Problem (1) arises in many fields, including compressed sensing, sensor networks, machine learning, data mining, and genomics, just to name a few. Usually, the smooth nonconvex term is some loss function capturing the degree of successful approximation of a model given the parameters $\mathbf{x}$, and the nonsmooth term is used to promote some extra structure on the solution, like sparsity.

Asynchronous methods can be traced back to the seminal paper [6] on *chaotic relaxation* in the solution of linear systems of equations and their study had a first culmination in the book [3]. In [3] asynchronous algorithms are studied in the framework of fixed point problems and two types of results are proven. First, convergence is established under the so-called *totally* asynchronous model−every $\mathbf{x}_i$ is updated infinitely often and obsolete information is eventually purged from the system−but with extremely strong assumptions on the problem. In particular, when optimization problems are considered, the conditions translate to the requirement that the problem be convex with an objective function which is not only smooth but with a gradient satisfying a maximum-norm contraction condition, which is a stronger requirement than strong convexity of the objective function. In a second set of results, convergence is established for smooth optimization problems with convex constraints under the so-called *partially* asynchronous model: a positive integer δ exists such that at most every δ iterations all components $\mathbf{x}_i$ are updated and the information used to update any block-variable at iteration k comes from at most the last δ iterations. We remark that the results in [3] are "deterministic", by which we mean that the block of variables on which a core performs an update is essentially chosen in a deterministic way and that the convergence result are also deterministic (i.e. of the type "every limit point is a stationary point" as opposed to "with probability 1 every limit point is a stationary point"). It is also of interest to note that no complexity results are given in [3]. After [3] research on asynchronous optimization methods somewhat withered away,

the reader can refer to [11] for further developments up to 2000. However, the interest in asynchronous methods revived in more recent years due both to the current trend towards huge scale optimization and to the ever more complex computational architectures being developed that require algorithms to be resilient to unpredictable hardware behaviours. The latter point is probably one of the reasons for which essentially all modern analyses of asynchronous optimization methods use probabilistic models, for they can easily accommodate the erratic behaviours of complex hardware architectures. In addition, a randomized choice of the variables $\mathbf{x}_i$ to be updated at each iteration has proven to be very effective even in synchronous and sequential methods [21]. And indeed, asynchronous parallelism has been successfully applied to many state-of-the-art optimization algorithms including stochastic gradient descent methods [13, 16, 19, 20, 22], randomized (proximal) coordinate descent algorithms [7, 17, 18], ADMM-like schemes [12, 14, 28], and fixed-point methods [2, 11, 23]. We refer the interested reader to [18, 23] for a detailed discussion of these methods. Here it is interesting to note just a few points.

Almost all the above methods handle convex optimization problems or, in the case of fixed point problems, nonexpansive mappings. The exceptions are [16] and [7, 8] that study unconstrained and constrained nonconvex optimization problems, respectively. However, the latter two papers propose algorithms that require, at each iteration, the global solution of nonconvex subproblems that could be extremely hard to solve and potentially as difficult as the original one. Moreover they require a memory lock between the time a core reads the current value of the selected block-component and the time it updates it, which prevents other cores to write that component. This might not be a minor requirement if one considers that, during the aforementioned time window, a core is required to compute the global solution of a nonconvex problem.

A second point of interest is whether asynchronous methods require a "consistent read" or can handle an "inconsistent read". Asynchronous algorithms produce a sequence $\mathbf{x}^k$ and at each iteration a core updates a component $\mathbf{x}_i$ (to be precise, the iteration counter k is increased every time a core updates a component, ties broken randomly, and $\mathbf{x}^{k+1}$ is the vector obtained immediately after the k-th update). In order to perform its update, core c uses a"local estimate", say $\tilde{\mathbf{x}}_c^k$, of $\mathbf{x}^k$. If $\tilde{\mathbf{x}}_c^k = \mathbf{x}^k$ the algorithm is synchronous; if instead $\tilde{\mathbf{x}}_c^k = \mathbf{x}^{k-d^k}$ for some positive integer d^k, the algorithm is asynchronous and it is said to use a "consistent read". If finally we have that each component of $\tilde{\mathbf{x}}_c^k$ coincides with the corresponding component of an older iteration which however may be different from component to component, i.e., if $(\tilde{\mathbf{x}}_c^k)_i = (\mathbf{x}^{k-d_i^k})_i$ for all i with d_i^k positive integers that may be different for different i (and cores), the algorithm is asynchronous and it is said to use an "inconsistent read". The sources of an inconsistent read in asynchronous methods are many, and depend on the computational architecture; two concrete examples are given next. Consider a shared memory multicore architecture, where the memory has registers $1, 2, \ldots, n$ that are accessed by the cores independently. In this setting consistent read means that when a core c accesses the shared memory

to read the vector of variables $\mathbf{x}$ to be used in its computations, thus forming its local copy $\tilde{\mathbf{x}}_c^k$, there is a lock on the memory to prevent other cores from writing until the reading of core c is terminated; inconsistent read occurs instead if different cores can read and write the memory at the same time, with no lock occurring. Experience indicates clearly that, in this setting, requiring a consistent read is a strong limitation that slows down the algorithm; asynchronous implementations naturally call for simultaneous uncoordinated readings and writings of the shared memory. But while in the shared memory setting a consistent read is still a (costly) possibility, there are other settings where inconsistent reads are really the only practical option. Consider for example a message-passing system with n processors (the number of processors need not be equal to the number of blocks, we assumed this here only for simplicity of presentation). Each processor c "owns" a block-variable $\mathbf{x}_c$ and has a local memory where its vector $\tilde{\mathbf{x}}_c^k$ is stored. In this setting, $\mathbf{x}^k$ is simply obtained by staking the value $\mathbf{x}_c$ of each processor c at any given k. Occasionally each processor c will communicate (through a possibly unreliable, time-varying network) the current value of $\mathbf{x}_c$ to the other processors that, when they receive it, update their local copies $\tilde{\mathbf{x}}_c$ accordingly. Note that at this level of description the structure of the network and the communication protocol are immaterial, we will only need that a positive number δ exists such that at most every δ iterations all processors have received an update from all the others. In this message-passing system, it is clear that only inconsistent reads are possible, since, given the complexity of the communication system, it is not sensible to assume processors can form a local copy $\tilde{\mathbf{x}}_c$ which is consistent with some iteration k. We finally observe that because of the physical reasons for a consistent/inconsistent read in a shared memory architecture, algorithms that allow for inconsistent read are also often termed as lock-free; we will use both terms interchangeably.

If one then considers the consistent/inconsistent read issue, again, all the papers mentioned above assume a consistent read, the exceptions being [7, 8, 16, 17, 19, 23] (we should also mention [1] which considers the solution of linear systems of equations). But, with the exception of [19], that only deals with the particularly simple case of the unconstrained minimization of a differentiable, strongly convex function with Lipschitz gradients, the models and analyses in these works are still far from representing a realistic description an inconsistent read asynchronous environment. First and foremost, the analyses in [1, 7, 8, 16, 17, 23] either seem to assume that $\tilde{\mathbf{x}}_c^k$ is deterministic or explicitly or implicitly assume that $\tilde{\mathbf{x}}_c^k$ (or some equivalent vector) is *independent* of the story of the algorithm up to iteration k and that $\tilde{\mathbf{x}}_c^k$ is the *same for all cores*, (i.e. $\tilde{\mathbf{x}}_c^k$ does not depend on c). These two facts go against the very nature of asynchrony and, in our view, put in jeopardy the theoretical significance of the results in [1, 7, 8, 16, 17, 23] (see [19] for objections in the same vein and Remark 2 for more details on this issue).

The above discussion shows that the development of an asynchronous method with inconsistent reads for the nonconvex Problem (1) is still an open problem. In this paper we fill this gap and study a general framework for the

asynchronous solution of Problem (1) based on a Successive Convex Approximation (SCA) idea: each core updates a chosen block-variable by solving a convex "approximation" of the original (possibly nonconvex) problem. Depending on the approximation chosen for the subproblems we can have proximal-gradient steps, Newton-type steps or other type of steps based on subproblems specifically tailored to the problem at hand. The proposed method enjoys several favorable properties: our main contributions are listed next.

1. We propose a novel probabilistic model of parallel asynchronous computation that faithfully captures the essential features of multi-core architectures and overcomes the shortcomings described above. This model is considerably more complex and general than those used previously, but allows us to cover, in a mathematically sound and unified way, both lock-free and not lock-free settings, parallel synchronous and asynchronous implementations (and also serial implementations) in either shared memory or distributed (with no shared memory)-based architectures.

2. We prove almost sure convergence for an algorithm that can be applied to *nonconvex* constrained optimization problems in the form (1). The natural idea underlying our approach is to decompose (1) into a sequence of (simpler) convex subproblems; the subproblems, one for each block of variables, are then solved in a parallel and asynchronous fashion among the cores.

3. All asynchronous methods mentioned so far are based on taking a *fixed step* in certain directions. While this is amenable to complexity analysis, it also has disadvantages. On the one hand the proper choice of the stepsize usually requires the knowledge of unknown constants and, on the other hand, and more importantly, the use of the stepsize that gives theoretical convergence usually leads to extremely slow practical convergence. In contrast to this situation we advocate the use of *diminishing stepsize* rules. This choice frees us from the necessity of a tuning of the stepsize and, in all our numerical experiments, has proven to be highly efficient.

4. Our algorithm is very flexible in the choice of the subproblems, which need not be necessarily "first-order" approximation of the original problem, as it is the case in all current asynchronous algorithms, which are all proximal-gradient or stochastic gradient schemes. We can of course choose to use a proximal-gradient-type update, but we can also decide to use updates based on Newton-type approximation or other approximations, according to the specific structure of the problem at hand. This is an especially appealing feature in asynchronous systems where very often communications costs are the bottleneck. In this case it is then useful to let the cores solve more complex subproblems, thus leading to less communication among the cores.

5. We also consider a fixed-step version of our algorithm and provide a iteration complexity analysis showing that a suitable defined optimality measure (remember we are not assuming convexity) will go below ε in $O(1/\varepsilon)$ iterations.

6. Finally, we provide extensive numerical results showing the our new algorithm seems able to outperform current asynchronous schemes on both convex and nonconvex problems.

Points 1-4 above are discussed in this paper, while points 5 and 6 are the subject of the companion paper [5].

2 Probabilistic Model and Algorithm

In this section we introduce the algorithm along with the underlying probabilistic model. For simplicity of presentation we begin assuming that the constraints are convex, i.e., all $\mathcal{X}_i$ in (1) are convex. In fact, the probabilistic modeling issues are independent of convexity and in this way we can concentrate on the main probabilistic ideas in a simpler setting. The unnecessary convexity assumption will then be removed in Section 4.

The section is organized as follows. We first introduce the assumptions on Problem (1). Then we describe the way a generic core updates a block-variable; and then we present the algorithmic framework and the associated probabilistic model. We conclude the section by presenting several possible instances of the proposed model.

2.1 Assumptions on the problem

We make the following blanket assumptions on Problem (1).

Assumption A.

(A1) Each $\mathcal{X}_i$ is nonempty, closed, and convex;
(A2) f is C^1 on an open set containing $\mathcal{X}$;
(A3) $\nabla_{\mathbf{x}_i} f$ is Lipschitz continuous on $\mathcal{X}$ with constant L_{f_i};
(A4) All $g_i(\mathbf{x}_i)$ are continuous, convex, (possibly nondifferentiable), and Lipschitz continuous with constant L_g on $\mathcal{X}$;
(A5) Problem (1) admits a solution.

As we already mentioned, the convexity in A1 is only made in order to simplify the presentation and be able to concentrate on the main modelling aspects; it will be removed in Section 4. The rest of the assumptions are rather standard. A3 holds trivially if $\mathcal{X}$ is bounded; this boundedness assumption also implies A5, given that all functions are continuous. Note that by assuming all g_i's to be continuous, we are not considering the case of extended-valued functions. The use of extended-valued functions in the objective functions is sometimes used to deal with constraints by formally reducing a constrained problem to an unconstrained one. However, in our approach we will need to deal with constraints explicitly and therefore we avoid the extended-valued function formalism; consequently, in most practical cases the g_i's are norms or polyhedral functions and A4 is satisfied.

2.2 Updating a block-variable $\mathbf{x}_i$

As we discussed in the Introduction, our method is not a standard gradient-like scheme but leverages the SCA techniques in [10]; therefore we begin with

the description of FLEXA [10], a synchronous, parallel method for the solution of Problem (1). Building on FLEXA, we will then introduce our new asynchronous model and algorithm.

Parallel synchronous updates. In the synchronous parallel method FLEXA, given the current iterate $\mathbf{x}^k$, all blocks $\mathbf{x}_i$ are updated at the same time. Each block is updated by solving a convexified version of the original optimization problem (1) wherein the nonconvex objective function F is replaced by a strongly convex surrogate

$$\hat{\mathbf{x}}_i(\mathbf{x}^k) \triangleq \underset{\mathbf{x}_i \in \mathcal{X}_i}{\operatorname{argmin}} \tilde{F}_i(\mathbf{x}_i; \mathbf{x}^k) \triangleq \tilde{f}_i(\mathbf{x}_i; \mathbf{x}^k) + g_i(\mathbf{x}_i), \tag{2}$$

where $\tilde{f}_i : \mathcal{X}_i \times \mathcal{X} \to \mathbb{R}$ is suitably chosen strongly convex approximations of f_i at $\mathbf{x}^k$. We consider the following general class of surrogate functions (we denote by $\nabla \tilde{f}_i$ the partial gradient of $\tilde{f}_i$ with respect to the first argument).

Assumption B (On the surrogate functions). Each $\tilde{f}_i$ is such that

(B1) $\tilde{f}_i(\bullet; \mathbf{x}^k)$ is uniformly strongly convex and continuously differentiable on $\mathcal{X}_i$ for all $\mathbf{x}^k \in \mathcal{X}$, with $c_{\tilde{f}}$ being the strong convexity constant independent of i and k;

(B2) $\nabla \tilde{f}_i(\mathbf{x}_i^k; \mathbf{x}^k) = \nabla_{\mathbf{x}_i} f(\mathbf{x}^k)$, for all $\mathbf{x}^k \in \mathcal{X}$;

(B3) $\nabla \tilde{f}_i(\mathbf{x}_i^k; \bullet)$ is Lipschitz continuous on $\mathcal{X}$, for all $\mathbf{x}_i^k \in \mathcal{X}_i$, with a Lipschitz constant L_B which is independent of i and k.

The above assumptions are quite natural: $\tilde{f}_i$ should be regarded as a (simple) strongly convex local approximations of f around the current iterate $\mathbf{x}^k$ that preserves the first order properties of f. Conditions B3 is a simple Lipschitzianity requirement that is surely satisfied, for example, if the set $\mathcal{X}$ is bounded. We discuss some valid instances of $\tilde{f}_i$'s below and refer the interested reader to [10] for further examples. Here we only note that finding approximations $\tilde{f}_i$ satisfying Assumption B usually offers no difficulty and that in any case it is always possible to choose $\tilde{f}_i(\mathbf{x}_i; \mathbf{x}^k) = \nabla_{\mathbf{x}_i} f(\mathbf{x}^k)^T(\mathbf{x}_i - \mathbf{x}_i^k) + \tilde{c}_i \|\mathbf{x}_i - \mathbf{x}_i^k\|^2$, where $\tilde{c}_i$ is a positive constant, i.e. the classical proximal-gradient update. Having the possibility to use a different $\tilde{f}_i$ may be useful to exploit any structure present in the problem; of course there is in general a trade-off: the more complex the $\tilde{f}_i$ the more information will be contained in $\hat{\mathbf{x}}_i(\mathbf{x}^k)$, but also the more complex its computation is expected to be. On the other hand the solution of more complex subproblems will in general decrease the number of information exchanges in the system which, in some cases, may be a key advantage. An appropriate choice of $\tilde{f}_i$ may be crucial to the numerical efficiency of the overall method, but the most suitable choice is problem dependent. Here we only list a few examples that go beyond the proximal-gradient choice.

• If $f(\mathbf{x}_1, \ldots, \mathbf{x}_N)$ is block-wise uniformly convex, instead of linearizing f one can exploit a second-order approximation and set $\tilde{f}_i(\mathbf{x}_i; \mathbf{x}^k) = f(\mathbf{x}^k) + \nabla_{\mathbf{x}_i} f(\mathbf{x}^k)^{\mathrm{T}}(\mathbf{x}_i - \mathbf{x}_i^k) + \frac{1}{2}(\mathbf{x}_i - \mathbf{x}_i^k)^{\mathrm{T}} \nabla^2_{\mathbf{x}_i \mathbf{x}_i} f(\mathbf{x}^k)(\mathbf{x}_i - \mathbf{x}_i^k) + \tilde{c}\|\mathbf{x}_i - \mathbf{x}_i^k\|^2$;

• In the same setting as above, one can also better preserve the partial convexity of f and set $\tilde{f}_i(\mathbf{x}_i; \mathbf{x}^k) = f(\mathbf{x}_i, \mathbf{x}_{-i}^k) + \tilde{c}\|\mathbf{x}_i - \mathbf{x}_i^k\|^2$, where $\mathbf{x}_{-i} \triangleq (\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_N)$;

• As a last example, suppose that f is the difference of two convex functions $f^{(1)}$ and $f^{(2)}$, i.e., $f(\mathbf{x}) = f^1(\mathbf{x}) - f^2(\mathbf{x})$, one can preserve the partial convexity in f setting $\tilde{f}_i(\mathbf{x}_i; \mathbf{x}^k) = f^{(1)}(\mathbf{x}_i, \mathbf{x}^k_{-i}) - \nabla_{\mathbf{x}_i} f^2(\mathbf{x}^k)^{\mathrm{T}}(\mathbf{x}_i - \mathbf{x}_i^k) + \frac{c}{2}\|\mathbf{x}_i - \mathbf{x}_i^k\|^2$.

Using $\hat{\mathbf{x}}_i(\mathbf{x}^k)$ defined in (2), the synchronous parallel update of each block $\mathbf{x}_i^k$ to $\mathbf{x}_i^{k+1}$ is performed taking a step from $\mathbf{x}_i^k$ along the direction $\hat{\mathbf{x}}_i(\mathbf{x}^k) - \mathbf{x}_i^k$ scaled by a scalar $\gamma^k > 0$, specifically

$$\mathbf{x}_i^{k+1} = \mathbf{x}_i^k + \gamma^k \left(\hat{\mathbf{x}}_i(\mathbf{x}^k) - \mathbf{x}_i^k\right), \quad \forall i = 1, \dots, N. \tag{3}$$

In our algorithm γ^k is chosen according to classical diminishing stepsize rules with an additional, simple safeguard. The precise conditions on γ^k are discussed in Sec. 3.

Asynchronous updates. We break now the synchronization enforced in the parallel updates (3) by allowing the cores to update the block-components $\mathbf{x}_i$ in an independent, random, and asynchronous fashion. We use a global index k to count iterations: whenever a core updates a block-component of the current $\mathbf{x}$, a new iteration $k \to k+1$ is triggered (this iteration counter is not required by the cores themselves to compute the updates). At iteration k, there is a core that updates a block-component $\mathbf{x}_{i^k}$ of $\mathbf{x}^k$ randomly chosen in the set $\mathcal{N} \triangleq \{1, \dots, N\}$ thus generating the vector $\mathbf{x}^{k+1}$. Therefore, $\mathbf{x}^k$ and $\mathbf{x}^{k+1}$ only differ in the i^k-th updated component. To update block i^k a core generally does not have access to the global state $\mathbf{x}^k$, but will instead use the possibly out-of-sync, delayed coordinates $\tilde{\mathbf{x}}^k = \mathbf{x}^{k-\mathbf{d}^k} \triangleq (\mathbf{x}_1^{k-d_1^k}, \dots, \mathbf{x}_N^{k-d_N^k})$, where d_i^k's are some nonnegative integer numbers. More specifically, a core may, at any time, and without communicating with other cores:

1. **Build $\mathbf{x}^{k-\mathbf{d}^k}$:** Build the inconsistently delayed vector $\mathbf{x}^{k-\mathbf{d}^k}$;
2. **Local computation:** Compute $\hat{\mathbf{x}}_{i^k}(\mathbf{x}^{k-\mathbf{d}^k})$ solving the strongly convex optimization problem

$$\hat{\mathbf{x}}_{i^k}(\mathbf{x}^{k-\mathbf{d}^k}) \triangleq \underset{\mathbf{x}_{i^k} \in \mathcal{X}_{i^k}}{\operatorname{argmin}} \tilde{F}_{i^k}(\mathbf{x}_{i^k}; \mathbf{x}^{k-\mathbf{d}^k}) \triangleq \tilde{f}_{i^k}(\mathbf{x}_{i^k}; \mathbf{x}^{k-\mathbf{d}^k}) + g_{i^k}(\mathbf{x}_{i^k});$$

3. **Read:** Read the current iterate $\mathbf{x}_{i^k}^k$, which may be different from what it cached, $\mathbf{x}_{i^k}^{k-\mathbf{d}^k}$, because other cores might have updated $\mathbf{x}$ in the interim;
4. **Update:** Update $\mathbf{x}_{i^k}^{k+1}$ by

$$\mathbf{x}_{i^k}^{k+1} = \mathbf{x}_{i^k}^k + \gamma^k \left(\hat{\mathbf{x}}_{i^k}(\mathbf{x}^{k-\mathbf{d}^k}) - \mathbf{x}_{i^k}^k\right).$$

In step 1 above the word "build" can have different meanings. For example, the formation of the inconsistently delayed vector $\mathbf{x}^{k-\mathbf{d}^k}$ can derive, in a shared memory system, from an inconsistent read of the shared memory, but we could also have simply used the values employed in the previous computations by that core or just read some of the components and use the old ones for the remaining components. Likewise, in step 4, the meaning of the word "update" depends on the environment in which we are operating. For example, referring to the two settings considered in the introduction, if we are in a shared

memory environment, "update" means " write the shared memory" while in the message passing case it means "write the local memory".

It is important to note that from the point of view of the single update the outcome is completely determined by the index i^k and the vector $\mathbf{x}_{i^k}^{k-\mathbf{d}^k}$ used to compute $\hat{\mathbf{x}}_{i^k}$ while the core physically performing the update is immaterial. For this reason, we dropped the dependence on the core in the steps 1-4 above.

2.3 The algorithmic model

We can now introduce our asynchronous algorithm and model. At any time, a core completes Steps 1-4 above. Every update from $\mathbf{x}^k$ to $\mathbf{x}^{k+1}$ is fully determined once $\mathbf{x}^0, \ldots, \mathbf{x}^k$ and the pair $(i^k, \mathbf{d}^k)$ are given. This justifies the model described in Algorithm 1, which we term Asynchronous FLexible ParallEl Algorithm (AsyFLEXA).

Algorithm 1 Asynchronous FLexible ParallEl Algorithm (AsyFLEXA)

Initialization: $k = 0$, $\mathbf{x}^0 \in \mathcal{X}$, $\{\gamma^k\}$
while a termination criterion is not met **do**
(S.1): The random variables $(\underline{i}^k, \underline{\mathbf{d}}^k)$ is realized as $(i, \mathbf{d})$;
(S.2): Compute $\hat{\mathbf{x}}_i(\mathbf{x}^{k-\mathbf{d}})$ [cf. (2)];
(S.3): Read $\mathbf{x}_i^k$;
(S.4): Set

$$\mathbf{x}_i^{k+1} = \mathbf{x}_i^k + \gamma^k(\hat{\mathbf{x}}_i(\mathbf{x}^{k-\mathbf{d}}) - \mathbf{x}_i^k); \tag{4}$$

(S.5): Update the iteration counter $k \leftarrow k + 1$;
end while
return $\mathbf{x}^k$

A few informal comments are in order, before continuing to go over the details.

1. The description of AsyFLEXA is geared towards standard presentations of algorithms and simplifies the notation in the convergence proof. However, it is useful to point out a fact that has some bearing on a proper understanding of our notational conventions and that might be clouded by our tendency to "think in a sequential manner". In the description of AsyFLEXA we assumed in steps S.1 through S.4 that we are at iteration k, but we should keep in mind that the iteration counter is increased (and defined) in the moment in which a core writes a new value for a block-variable in step S.4. In general there are several cores acting in parallel and there is no clock saying "we are now at iteration k and this specific core is performing the operations in S.1-S.4". Rather, the core writing the new value of a block-variables triggers retrospectively what core is actually performing the update and determines all quantities involved in S.1-S.4.

2. If one looks at the read operation in step S.3 and updating in step S.4 one could be lead to think there is a lock on the whole vector $\mathbf{x}^k$ between the time a core reads a current value of a block-variable in S.3 and the time it updates it in S.4, but this is not the case. If one looks at what happens at the

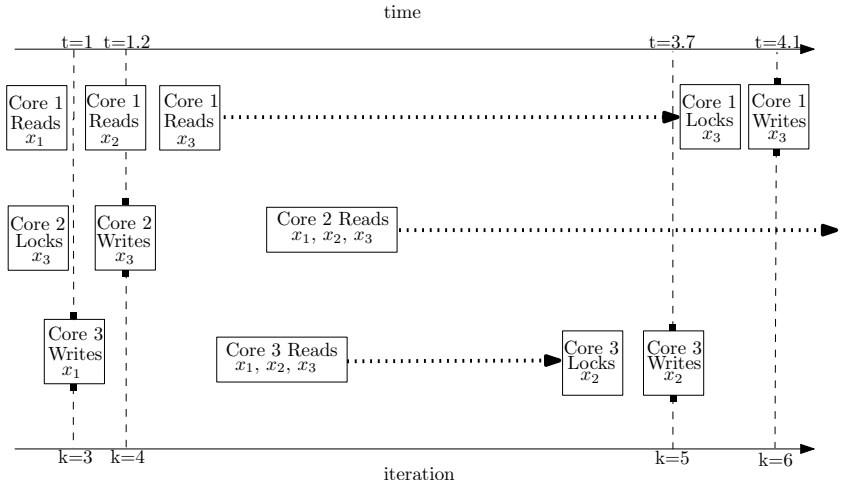

Fig. 1: Three cores performing AsyFLEXA on a vector $\mathbf{x}$ with three blocks. Note 1) the iteration counter increases each time a core finishes a computation and updates a block, 2) when updating a block that block must be locked from other cores modifying it, but it is for a relatively short period of time, and 3) the inconsistent read properties result in vectors $\tilde{\mathbf{x}}$ being used for computation of the subproblem solution whose components have existed at some point in memory, but not necessarily the entire vector.

core level and takes into account the observation in the previous point, one sees that what is really needed here is that whenever a core c reads a block-variable in S.3 then no other core updates that block until c has completed its updating. This is a very minor requirement which is automatically satisfied, for example, in a message passing environment or also in a shared memory setting if each core has exclusive handling of group of variables (a very common setting which is also the one used in our numerical experiments). In principle, in a shared memory setting where all cores can update all variables we may need to impose a lock on a block-variable. In practice the possibility that two different cores update the same block-variables at the same time appears to be negligible and one can safely dispense with such a lock in most concrete situations. We note that this block-wise lock is standard, and in our case is only required for the duration of performing a linear combination of the two vectors whose dimension is the size of block, whereas, for instance [18] and [7, 8] implicitly assume a lock on the block being updated while a prox operation is being performed (in [7, 8] the global optimal solution of a nonconvex problem must be computed) .

Figure 1 illustrates the iteration counter, block locking, and inconsistent read properties in the asynchronous setting. The thick dotted horizontal arrows indicate local core computation (solving the subproblem) and we indicate when the iteration are updated and the time when updates take place. At iteration k=3, Core 3 writes x_1, that is, vector $\mathbf{x}^3$ differs from $\mathbf{x}^2$ in the first component. Core 2 locks x_3 to quickly read and perform the linear combination with $\hat{\mathbf{x}}(\tilde{\mathbf{x}}^3)$,

and updates the 3rd block and so $\mathbf{x}^4$ differs from $\mathbf{x}^3$ in just the 3rd component. Note that Core 2 reads $\mathbf{x}_3^2$ but this is equal to $\mathbf{x}_3^3$ because the writing that Core 3 performs in between the lock and the write is on a different block. At iteration $k = 5$, Core 3 writes x_2. In this case $\tilde{\mathbf{x}}^4$, used for the computation of $\mathbf{x}_2^5 = (1-\gamma^4)\mathbf{x}_2^4 + \gamma^4\hat{\mathbf{x}}_2(\tilde{\mathbf{x}}^4)$, is exactly equal to $\mathbf{x}^4$ since Core 3 reads the vector entirely after the last update. However, when Core 1 performs a lock and write for iteration $k = 6$, we have that the vector it read to perform the computation $\tilde{\mathbf{x}}^5$ satisfies $\tilde{\mathbf{x}}_1^5 = \mathbf{x}_1^2 \neq \mathbf{x}_1^3$, $\tilde{\mathbf{x}}_2^5 = \mathbf{x}_3^2$, and $\tilde{\mathbf{x}}_3^5 = \mathbf{x}_3^4 \neq \mathbf{x}_3^2$, and so $\tilde{\mathbf{x}}^5$ is a vector that has never existed in the shared memory. It can be seen that the vector of delays at $k = 6$ reads $\mathbf{d}^6 = (3, 1, 0)$, since the first block was updated in iteration $k = 3$, which is three iterations earlier, the second block was updated at the previous iteration and otherwise not since it was read, and the third component has remained the same since it was read.

3. We assume that the update in step S.4, i.e. the writing on some memory location, is *atomic*, i.e., a block-coordinate is not further broken to smaller components during an update; they are all updated at once. This is easily true in most modern hardware if the block-variable being updated is a scalar. Whenever a block-variable actually comprises several scalar variables we can satisfy the atomic requirement by either enforcing a coordinate lock or, more satisfyingly, by taking a dual-memory approach as described in [23, Section 1.2.1].

4. The conceptual core of the algorithm model is step S.1. Each step k is characterized by the fact that (a processor "awakes" and) an update of a random index is performed by using some vector $\tilde{\mathbf{x}}^k = \mathbf{x}^{k-\mathbf{d}^k}$ (note that $\tilde{\mathbf{x}}^k$ at iteration k is totally determined by the vector $\mathbf{d}^k$). Thus, there are two unknown elements here: the index i^k which is updated and the vector of "delays" $\mathbf{d}^k$. These two elements are determined by a multitude of facts which cannot be previewed beforehand, such as: Which processor finishes computing a subproblem solution and is ready to perform an update? What index has the processor chosen for the update? What estimate $\tilde{\mathbf{x}}^k$ is used in the update (a vector that in turn depends on all previous updates, on the communication characteristics between the processors, on the way they access the memory, on possible hardware failures, etc.)? Therefore, it seems both natural and necessary to model the sequence of pairs $(i^k, \mathbf{d}^k)$ generated by the algorithmic process as a stochastic process. A formal description of the probabilistic model underlying AsyFLEXA is given next.

Probabilistic model. Given an instance of Problem (1) and an initial point $\mathbf{x}^0$, we assume that the pair index-delay $(i, \mathbf{d})$ in S.1 of Algorithm 1 is a realization of a random vector $\underline{\boldsymbol{\omega}}^k \triangleq (\underline{i}^k, \underline{\mathbf{d}}^k)$, taking values on $\mathcal{N} \times \mathcal{D}$, where $\mathcal{N} \triangleq \{1, \ldots, N\}$ and $\mathcal{D}$ is the set of all possible delay vectors. We will assume that each $\tilde{\mathbf{x}}^k$ does not use information that is older than δ iterations (see Assumption C below); this translates into the requirement that $\mathbf{d}_i^k \leq \delta$ for all k and i. Therefore $\mathcal{D}$ is contained in the set of all possible N-length vectors whose components are integers between 0 and δ. More formally, let

Ω be the sample space of all the sequences $\omega \triangleq \{(i^k, \mathbf{d}^k)\}_k$[1]; let us define a probability space whose sample space is Ω and that can sensibly represents our algorithmic framework. We first introduce some shorthand notation: we set $\underline{\boldsymbol{\omega}}^{0:t} \triangleq (\underline{\boldsymbol{\omega}}^0, \underline{\boldsymbol{\omega}}^1, \dots, \underline{\boldsymbol{\omega}}^t)$ (the first $t+1$ random variables); $\boldsymbol{\omega}^{0:t} \triangleq (\boldsymbol{\omega}^0, \boldsymbol{\omega}^1, \dots, \boldsymbol{\omega}^t)$ ($t+1$ possible values for the random variables $\underline{\boldsymbol{\omega}}^{0:t}$); and $\boldsymbol{\omega}_{0:t} \triangleq (\boldsymbol{\omega}_0, \boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_t)$ (the first $t+1$ elements of ω). In order to define a σ-algebra on Ω we consider, for every $k \geq 0$ and every $\boldsymbol{\omega}^{0:k} \in \mathcal{N} \times \mathcal{D}$, the cylinder

$$C^k(\boldsymbol{\omega}^{0:k}) \triangleq \{\omega \in \Omega : \boldsymbol{\omega}_{0:k} = \boldsymbol{\omega}^{0:k}\},$$

i.e., $C^k(\boldsymbol{\omega}^{0:k})$ is the subset of Ω of all elements ω whose first k elements are equal to $\boldsymbol{\omega}^0, \dots \boldsymbol{\omega}^k$. Let us now denote by $\mathcal{C}^k$ the set of all possible $C^k(\boldsymbol{\omega}^{0:k})$ when $\boldsymbol{\omega}^t$, $t = 0, \dots, k$, takes all possible values; note, for future reference, that $\mathcal{C}^k$ is a partition of Ω. Denoting by $\sigma\left(\mathcal{C}^k\right)$ the sigma-algebra generated by $\mathcal{C}^k$, define for all k,

$$\mathcal{F}^k \triangleq \sigma\left(\mathcal{C}^k\right) \qquad \text{and} \qquad \mathcal{F} \triangleq \sigma\left(\cup_{t=0}^{\infty} \mathcal{C}^t\right). \tag{5}$$

We have $\mathcal{F}^k \subseteq \mathcal{F}^{k+1} \subseteq \mathcal{F}$ for all k. The latter inclusion is obvious, the former derives easily from the fact that any cylinder in $\mathcal{C}^{k-1}$ can be obtained as a finite union of cylinders in $\mathcal{C}^k$.

The desired probability space is fully defined once the probabilities $\mathbb{P}(C^k(\boldsymbol{\omega}^{0:k}))$ of all cylinders are given. Of course these probabilities should satisfy some very natural, minimal consistency properties, namely: (i) the probabilities of the union of a finite number of disjoint cylinders should be equal to the sum of the probabilities assigned to each cylinder; and (ii) suppose that a cylinder $C^k(\boldsymbol{\omega}^{0:k})$ is contained in the union U of a countably infinite number of other cylinders, then $\mathbb{P}(C^k(\boldsymbol{\omega}^{0:k})) \leq P(U)$. Suppose now these probabilities are given, under the above described conditions (i) and (ii), classical theorems (see for example [15, Theorem 1.53]) ensure that we can extend these probabilities to a probability measure P over the measurable space $(\Omega, \mathcal{F})$ thus turning this into a probability space $A \triangleq (\Omega, \mathcal{F}, P)$ which will be our standing working space. By appropriately choosing the probabilities of the cylinders we can model in a unified way many cases of practical interest, see next subsection.

Given A, we can finally define the discrete-time, discrete-value stochastic process $\underline{\boldsymbol{\omega}}$ where $\{\underline{\boldsymbol{\omega}}^k(\omega)\}_{k \in \mathbb{N}_+}$ is a sample path of the process. The k-th entry $\underline{\boldsymbol{\omega}}^k(\omega)$ of $\underline{\boldsymbol{\omega}}(\omega)$−the k-th element of the sequence ω−is a realization of the random vector $\underline{\boldsymbol{\omega}}^k = (\underline{i}^k, \underline{\mathbf{d}}^k) : \Omega \mapsto \mathcal{N} \times \mathcal{D}$. This process fully describes the evolution of Algorithm 1. Indeed, given an instance of Problem (1) and a starting point, the trajectories of the variables $\mathbf{x}^k$ and $\mathbf{x}^{k-\mathbf{d}}$ are completely determined once a sample path $\{(i^k, \mathbf{d}^k)\}_{k \in \mathbb{N}_+}$ is drawn from $\underline{\boldsymbol{\omega}}$.

Note that the joint probability

$$p_{\underline{\boldsymbol{\omega}}^{0:k}}(\boldsymbol{\omega}^{0:k}) \triangleq \mathbb{P}(\underline{\boldsymbol{\omega}}^{0:k} = \boldsymbol{\omega}^{0:k})$$

[1] With a little abuse of notation, in order to avoid cluttering the notation, we indicate by $\boldsymbol{\omega}_k$ the k-th element of the sequence $\omega \in \Omega$ and by $\boldsymbol{\omega}^k$ the value taken by the random variable $\underline{\boldsymbol{\omega}}^k$ over ω, i.e. $\underline{\boldsymbol{\omega}}^k(\omega) = \boldsymbol{\omega}$.

is simply the probability of the corresponding cylinder: $C^k(\boldsymbol{\omega}^{0:k})$. We will often need to consider the conditional probabilities $p((i,\mathbf{d})\,|\,\boldsymbol{\omega}^{0:k}) \triangleq \mathbb{P}(\underline{\boldsymbol{\omega}}^{k+1} = (i,\mathbf{d})|\underline{\boldsymbol{\omega}}^{0:k} = \boldsymbol{\omega}^{0:k})$. Note that we have

$$p((i,\mathbf{d})\,|\,\boldsymbol{\omega}^{0:k}) = \frac{\mathbb{P}(C^{k+1}(\boldsymbol{\omega}^{0:k+1}))}{\mathbb{P}(C^k(\boldsymbol{\omega}^{0:k}))}, \tag{6}$$

where, as usual, we tacitly assume $p((i,\mathbf{d})\,|\,\boldsymbol{\omega}^{0:k}) = 0$, if $\mathbb{P}(C^k(\boldsymbol{\omega}^{0:k})) = 0$. We remark that these probabilities need not be known in practice. We only require that the hardware architecture and the software implementation guarantee the underlying probabilistic model is the one just given and further assume some additional minimal conditions are satisfied, as stated next.

Assumption C. Given Algorithm 1 and the stochastic process $\underline{\boldsymbol{\omega}}$, suppose that

(C1) There exists a $\delta \in \mathbb{N}_+$, such that $d_i^k \leq \delta$, for all i and k;
(C2) For all $i = 1,\dots,N$ and $\boldsymbol{\omega}^{0:k-1}$ such that $p_{\underline{\boldsymbol{\omega}}^{0:k-1}}(\boldsymbol{\omega}^{0:k-1}) > 0$, it holds

$$\sum_{\mathbf{d}\in\mathcal{D}} p((i,\mathbf{d})\,|\,\boldsymbol{\omega}^{0:k-1}) \geq p_{\min},$$

for some $p_{\min} > 0$;
(C3) It holds that

$$\mathbb{P}\left(\left\{\omega \in \Omega \,:\, \liminf_{k\to\infty}\, p(\boldsymbol{\omega}\,|\,\boldsymbol{\omega}^{0:k-1}) > 0\right\}\right) = 1.$$

These are quite reasonable assumptions with very intuitive interpretations. C1 just limits the age of the old information used in the updates. Condition C2 guarantees that at each iteration every block-index i has a non negligible positive probability to be updated. C3 simply says, roughly speaking, that the probability of the event that comprises all ω such that for an infinite number of iterations the algorithm picks-up a pair index-delay whose probability decreases to zero, must have zero probability. Very loosely speaking, C3 requires that the overall probability of increasingly unlikely possible evolutions of the algorithm be zero. A very simple case in which C2 and C3 are automatically satisfied is when the probabilities $p((i,\mathbf{d})\,|\,\boldsymbol{\omega}^{0:k})$ are independent of history $\boldsymbol{\omega}^{0:k}$, for all k.

Remark 1 The probability space A is meant to capture all the possible evolutions of Algorithm 1. Once an instance of Problem (1) and a starting point $\mathbf{x}^0 \in \mathcal{X}$ are given, every $\omega \in \Omega$ corresponds to a possible evolution of the algorithm. Indeed, given $\mathbf{x}^0$ and $\omega \in \Omega$, the sequence $\{\mathbf{x}^k\}$ generated by the algorithm is fully determined. Note that knowledge of the probability space A is in no way required by the cores themselves to compute the updates. The meaning of the model and of the corresponding convergence result in Theorem 1 (cf. Sec. 3) is to say, supposing that the physical hardware, the communication protocols and the overall implementation of the method are such that the

very mild assumptions C1-C3 hold, then the convergence results in Theorem 1 are true. In essence the probabilistic model is only used in the convergence proof and does not imply any additional complexity in the implementation.

Remark 2 Our model is more general and complex that those adopted so far in the literature, see [7,8,16,17,23]. In our view, this additional complexity is necessary to give a reliable foundation to the analysis of lock-free asynchronous optimization methods and in order to be able to cover computational and algorithmic settings that can not be analyzed using current results. Indeed, we believe that some probabilistic aspects have been somewhat overlooked in recent papers on the subject and in fact one of our aims in this paper was to rectify this situation. Below we discuss more in detail the seminal paper [17], but similar comments apply also to [7,8,16,23], since the models therein share many main common features with that in [17]; for simplicity we also refer to our notation to denote the corresponding quantities in [17].

In [17] all (block-)variables are scalar and it is assumed that the algorithm repeatedly chooses *uniformly at random* an index i and then performs an update of that variable, using the information given by $\tilde{\mathbf{x}}^k$. The assumption that the variables are updated with uniform probability seems very demanding and difficult to enforce in most practical settings, for example in a message passing environment or in a shared memory architecture where each core is in exclusive charge of the updating of a group of variables, the latter being the computational setting used in most computational tests of asynchronous methods. In fact, even assuming that each core chooses uniformly at random an index i every time it awakes, and even assuming a perfect "hardware symmetry", it could happen, for example, that computing the partial gradient with respect to the first variable takes the double of the time necessary to compute the partial gradient with respect to the second variable; in this case it is clear that the second variable will be updated more often. Note that the problem arises exactly because of the multicore architecture; the same uniformity assumption is perfectly legitimate in, say, a randomized sequential block-coordinate descent method. See Sec. 2.4 below for more discussion on this point and some examples.

At a more fundamental level, a key issue is that the only random variables in [17] appear to be the indices i, but the nature of $\tilde{\mathbf{x}}^k$ and of $\mathbf{d}^k$ is left somewhat ambiguous. While it is clear from our discussion so far that also $\tilde{\mathbf{x}}^k$ and $\mathbf{d}^k$ should be modeled as random variables, with $\tilde{\mathbf{x}}^k$ depending on the whole history of the algorithm up to iteration k and on $\mathbf{d}^k$, in the proofs of [17] $\tilde{\mathbf{x}}^k$ is treated as a deterministic quantity, as it is shown by the way it is dealt with when handling some conditional expectations. Furthermore, the proofs in [17] deal with $\tilde{\mathbf{x}}^k$ as if it were the same whatever the index selected; at the hardware level this is equivalent to requiring that all cores use the same $\tilde{\mathbf{x}}^k$, a fact clearly against the very idea of asynchrony, with cores acting independently. All these problems seem to derive from the probabilistic description adopted in [7, 8, 16, 17, 23] that appears to be too simplistic to capture all the aspects of lock-free asynchronous minimization algorithms.

We finally underline that, in spite of some possible mathematical deficiencies, [7, 8, 16, 17, 23] are a real milestone in the analysis of asynchronous optimization methods and opened the way to a new kind of approach to this class of algorithm.

2.4 Examples and special cases

The proposed algorithm model encompasses several instances of practical interest, some of which are discussed next. These examples clearly show that we can easily recover settings analyzed in previous papers but also deal in a sound way with some settings that to date could not be reliably analyzed. The crucial point here is not so much showing that we can give exact values to the probabilities that define our model, but rather that these probabilities exist, satisfy our assumptions and therefore the corresponding model can be analyzed using the results in this paper.

1. Deterministic sequential cyclic block-coordinate descent method: This is of course a limit case, but it is very interesting that we can nevertheless fit it into our framework; furthermore it allows us to present in the simplest possible setting the way we can deal with deterministic algorithms. In a deterministic, sequential, cyclic method there is only one core that cyclically updates all block-variables in a given order, for simplicity we assume the natural order, from 1 to N. Since there is only one core, the reading is always consistent and there are no delays: $\mathcal{D} = \{\mathbf{0}\}$. To represent the cyclic choice it is now enough to assign probability 1 too all cylinders of the type

$$C^k = \{\omega : \omega_0 = (1, \mathbf{0}), \omega_1 = (2, \mathbf{0}), \dots, \omega_k = ((k \; mod \; N) + 1, \mathbf{0})\}$$

and probability zero to all others. It is easy to see that Assumption C is satisfied. This can be seen as a probabilistic model of the deterministic algorithm in [27]; the difference is that while we know that the deterministic algorithms we described converges in a deterministic sense, by applying Theorem 1 we get a.s. convergence; but this is a natural consequence of describing a deterministic situation by using a probabilistic approach.

2. Randomized sequential block-coordinate descent method: Suppose there is only one core and therefore, as in the previous case, at each iteration $\tilde{\mathbf{x}}^k = \mathbf{x}^k$ or, equivalently, $\mathcal{D} = \{\mathbf{0}\}$. At difference with the previous case, assume however that at each iteration the core selects randomly an index i with a probability that is bounded away from zero. The resulting scheme obviously corresponds to a situation where the cylinders are assigned arbitrary probabilities with the only requirement that they respect all the conditions given in previous subsection, the defining point here being $\mathcal{D} = \{\mathbf{0}\}$.

3. Randomized parallel block-coordinate descent method: Consider a multicore processor with C cores and denote by $\mathbf{0}$, $\mathbf{1}$, $\dots$, $\boldsymbol{C-1}$ the N-length vectors whose components are, respectively, all $0, 1, \dots, C-1$; set

$\mathcal{D} = \{\mathbf{0}, \mathbf{1}, \ldots, \boldsymbol{C-1}\}$. Assign to the cylinders the following probabilities

$$
\begin{aligned}
&\mathbb{P}(C^0((i_0,\mathbf{0}))) = 1/N, && \forall i_0 \in \mathcal{N}\\
&\mathbb{P}(C^1((i_0,\mathbf{0}),(i_1,\mathbf{1}))) = 1/N^2, && \forall i_0,i_1 \in \mathcal{N}\\
&\ldots\\
&\mathbb{P}(C^{C-1}((i_0,\mathbf{0}),(i_1,\mathbf{1}),\ldots,(i_{C-1},\boldsymbol{C-1}))) = 1/N^C, && \forall i_0,\ldots,i_{C-1} \in \mathcal{N}\\
&\mathbb{P}(C^{C}((i_0,\mathbf{0}),(i_1,\mathbf{1}),\ldots,(i_{C-1},\boldsymbol{C-1}),(i_C,\mathbf{0}))) = 1/N^{C+1}, && \forall i_0,\ldots,i_{C} \in \mathcal{N}\\
&\mathbb{P}(C^{C+1}((i_0,\mathbf{0}),(i_1,\mathbf{1}),\ldots,(i_{C},\mathbf{0}),(i_{C+1},\mathbf{1}))) = 1/N^{C+2}, && \forall i_0,\ldots,i_{C+1} \in \mathcal{N}\\
&\ldots\\
&\mathbb{P}(C^{2C-1}((i_0,\mathbf{0}),(i_1,\mathbf{1}),\ldots,(i_{2C-1},\boldsymbol{C-1}))) = 1/N^{2C}, && \forall i_0,\ldots,i_{2C-1} \in \mathcal{N}\\
&\ldots
\end{aligned}
$$

What we are saying is that in the first C iterations (from $k=0$ to $k=C-1$) all updates are performed using the same point $\tilde{\mathbf{x}}^k = \mathbf{x}^0$, and at each iteration any index has probability $1/N$ to be selected. This situation is then repeated for the following C iterations, this time using the same $\tilde{\mathbf{x}}^k = \mathbf{x}^C$ and so on. This model clearly corresponds to a randomized parallel block-coordinate descent method wherein C processors update C block-variables chosen uniformly at random. Note that assumption C is trivially satisfied.

The example above clearly shows that defining probabilities by using the cylinders can be quite tedious even in simple cases. Using (6) we can equivalently define the probabilistic model by giving the conditional probabilities $p((i,\mathbf{d}) \,|\, \boldsymbol{\omega}^{0:k})$, which is particularly convenient when at every iteration k the probability of the random variable $\underline{\boldsymbol{\omega}}^k$ taking the value $(i,\mathbf{d})$ is independent of $\boldsymbol{\omega}^{0:k-1}$. We exemplify this alternative approach in the following examples.

4. Asynchronous block-coordinate descent in shared memory system with inconsistent read: Consider a generic shared memory system. Since we do not make any particular assumption (contrast this to the next two examples), the set $\mathcal{D}$ is given by all N-length vectors whose components are any non negative integers between 0 and δ. Suppose that, at every k, all cores select an index uniformly at random, but we leave the possibility that the probabilities with respect to delays are different. In other words, we suppose that for every given $k \geq 0$, $\boldsymbol{\omega}^{0:k}$, and $i \in \mathcal{N}$, we have

$$\sum_{\mathbf{d}\in\mathcal{D}} p((i,\mathbf{d}) \,|\, \boldsymbol{\omega}^{0:k}) = \frac{1}{N}.$$

Note that this setting corresponds to the one analyzed in [7, 8, 17]. However, we can easily change the values of the $\sum_{\mathbf{d}\in\mathcal{D}} p((i,\mathbf{d}) \,|\, \boldsymbol{\omega}^{0:k})$ and make the probabilities of selecting indices different one from the other and/or depend on the iteration and/or on the history of the process; these options are not available to [7,8,16,17]. This possibility has important practical ramifications, since the assumption that the indices are selected with uniform probability is extremely strong, as we already commented on in Remark 2. We make here another example hinting at this difficulty. Consider a message passing system; it will be in general true that cores performing operations are different and on top, the number of block-variables dealt by each core may also be different; in

this situation uniformity of the probabilities of selecting an index cannot be expected.

5. Asynchronous block-coordinate descent in shared memory system with consistent read: This is a variant of the previous case, where we assume consistent read. The consistent read requirement simply translates in the fact that the set $\mathcal{D}$ is formed by vectors with all components equal to a common nonnegative integer. Recalling that the maximum delay is δ we therefore have $\mathcal{D} = \{\mathbf{0}, \mathbf{1}, \dots, \boldsymbol{\delta}\}$.

6. Asynchronous block-coordinate descent in shared memory system with inconsistent read and block-variables partitioning: This is the setting most often used in numerical experiments, since it has proven to be most effective in practice; it also models a message passing architecture. Suppose we have C cores and that we partition block-variables in C groups $I_1, I_2, \dots, I_C$. Each block I_c is assigned to core c and only core c can update block-variables in I_c. Our mathematical translation of this setting is exactly the one in the asynchronous block-descent in shared memory system with inconsistent read discussed in Example 4 above with an additional provision. In fact, since no core $c\prime \neq c$ can update the variables in I_c, all this variables can always be assumed to have zero delay. To model this fact it is enough to set, for all $\boldsymbol{\omega}^{0:k}$ and i, with $i \in I_c$,

$$p((i, \mathbf{d}) \,|\, \boldsymbol{\omega}^{0:k}) = 0, \text{ if even just a } \mathbf{d}_j,\ j \in I_c, \text{ is not zero.}$$

Note that in [17] the numerical experiments are carried out for a *without replacement* setting which is just a variant of this setting and for which no theoretical analysis is offered in [17]. In this *without replacement* setting, the block-variables are partitioned among cores and then, at each "epoch", each core performs a random reordering of its variables and then updates them all in order. Note also, once again, that because cores might be different, might have different access time to the memory, can be assigned a different number of variables, and because the computation of $\hat{\mathbf{x}}_i$ might require different times for different i's, the uniform probability assumption on the selection of the indices is unlikely to be satisfied. However, we can easily cover also this case by simply merging the reasonings exposed here and in example 2 above.

We stop here with the detailed example but we could easily cover more complex situations. Consider for example a network of multicore processors. Each processor p is assigned a group I_p of variables and p is the only processor that can update the variables in I_p (with the I_p forming of course a partition of the variables). Each processor updates its own variables following the scheme of a shared memory situation with inconsistent read while receiving information on the variables not in I_p in a message passing fashion from the other processors. It is quite easy to mix the two situations analyzed separately in the examples above and model this common architecture within our framework. In addition, note that we can take advantage, in this framework, of varying computational power and communication speed of different processors by changing the form (complexity) of the subproblem depending on which core does the update.

3 Convergence Results

We present now our main convergence theorem, under Assumptions A-C. The extension to the case of nonconvex constraints is addressed in Sec. 4.

Before stating convergence, we need to specify how to choose the stepsize γ^k. Since line-search methods are out of question in an asynchronous environment, there are two other options, namely: i) a fixed (sufficiently small) step; and ii) a diminishing stepsize rule. A fixed stepsize has been adopted universally in all papers dealing with asynchronous methods. While such a choice is instrumental to prove iteration complexity bounds, it suffers from several drawbacks. First, the choice of the proper stepsize requires the knowledge of unknown parameters, whose (generally loose) estimate might result in a very small value of the stepsize. In fact, in our experience (see, e.g., [5]), algorithms equipped with a constant stepsize behave poorly and are extremely slow in practice. For all these reasons, we consider here, for the first time in asynchronous methods, the use of diminishing stepsize rules. In the companion paper [5] we provide complexity results, under constant stepsize rules.

For the time being, we make the following assumption on the stepsizes γ^k.

Assumption D (On the stepsize). The sequence $\{\gamma^k\}$ is chosen so that

(D1) $\gamma^k \downarrow 0$; $\sum_{k=0}^{\infty} \gamma^k = +\infty$; and $\sum_{k=0}^{\infty} (\gamma^k)^2 < +\infty$;
(D2) $\frac{\gamma^{k+1}}{\gamma^k} \geq \eta \in (0,1)$.

Conditions in D1 are standard and satisfied by most practical diminishing stepsize rules; see, e.g., [4]. Condition D2 dictates that $\sum_{k=0}^{\kappa} \gamma^k \geq \gamma^0(\eta^0 + \eta^1 + \cdots + \eta^\kappa)$, that is the partial sums $\sum_{k=0}^{\kappa} \gamma^k$ must be minorized be a *convergent* geometric series. Given that $\sum_{k=0}^{\infty} \gamma^k = +\infty$, D2 is clearly very mild and indeed it is also satisfied by most classical diminishing stepsize rules. For example, the following rule satisfies Assumption D and has been found very effective in our experiments: $\gamma^{k+1} = \gamma^k \left(1 - \mu\gamma^k\right)$, with $\gamma^0 \in (0,1]$ and $\mu \in (0,1)$.

We can now state the convergence result of Algorithm 1.

Theorem 1 *Let Problem* (1) *be given, along with Algorithm 1 and the stochastic process* $\underline{\boldsymbol{\omega}}$. *Let* $\{\mathbf{x}^k\}$ *be the sequence generated by the algorithm, given* $\mathbf{x}^0 \in \mathcal{X}$. *Suppose that i) Assumptions A-D hold; and ii)* $\{\mathbf{x}^k\}$ *is bounded almost surely (a.s.). Then the following hold:*

(a) Every limit point of $\{\mathbf{x}^k\}$ *is a stationary solution of Problem* (1) *a.s.;*
(b) The sequence of objective function values $\{F(x^k)\}$ *converges a.s..*

Proof See Appendix.

In addition to Assumptions A-D, Theorem 1 requires the sequence $\{\mathbf{x}^k\}$ to be bounded a.s.. The following corollary provides some concrete conditions for this boundedness to hold.

Corollary 1 *The same conclusions of Theorem 1 hold if the assumption on the a.s. boundedness of* $\{\mathbf{x}^k\}$ *is replaced by either of the following two:*

i) F is coercive on $\mathcal{X}$; $\{\gamma^k\}$ is such that $\gamma^{k+1} \leq \gamma^k$, for sufficiently large k; and $\|\nabla_{\mathbf{x}_i} f\| \leq L_{\nabla_i}$ on $\mathcal{X}_i$, for any $i \in \mathcal{N}$;
ii) F is coercive on $\mathcal{X}$ and $\mathbf{x}_i^k = \tilde{\mathbf{x}}_i^k$, for any $i \in \mathcal{N}$ and $k \geq 0$.

While condition i) is easy to understand, condition ii) needs some comments. The equality $\mathbf{x}_i^k = \tilde{\mathbf{x}}_i^k$ is automatically satisfied in a message passing setting or in a shared memory-based architecture if the variables are partitioned and assigned to different cores (see Example 6 in Sec. 2.4). In other settings, $\mathbf{x}_i^k = \tilde{\mathbf{x}}_i^k$ can be enforced by a software lock that, once a core c has read a block-variable $\mathbf{x}_i$, prevents other cores to change it until c has performed its update. Note that this is much less restrictive than a lock on the entire vector $\mathbf{x}$, which instead is used when a consistent read is forced. In practice, the aforementioned componentwise lock is not likely to affect in any way the performance of the algorithm, since in most of the settings the number of cores is much smaller than the number of block-variables.

4 Nonconvex constraints

In this section, we generalize our convergence results to the case of nonconvex constraints. Specifically, we consider the following more general problem:

$$\begin{aligned} \min_{\mathbf{x}} \quad & F(\mathbf{x}) \triangleq f(\mathbf{x}) + G(\mathbf{x}) \\ & \left.\begin{aligned} & \mathbf{x} \in \mathcal{X} \\ & c_{j_1}(\mathbf{x}_1) \leq 0, \quad j_1 = 1, \ldots, m_1 \\ & \vdots \\ & c_{j_N}(\mathbf{x}_N) \leq 0, \quad j_N = m_{N-1}+1, \ldots, m_N \end{aligned}\right\} \triangleq \mathcal{K} \end{aligned} \tag{7}$$

where there are now nonconvex private constraints in the form $c_{j_i}(\mathbf{x}_i) \leq 0$, for $j_i = m_{i-1}, \ldots, m_i$, $i = 1, \ldots, N$, and $m_0 = 0$, with each $c_{j_i} : \mathcal{X}_i \to \mathbb{R}$. We define by $\mathcal{K}_i \triangleq \{\mathbf{x}_i \in \mathcal{X}_i : c_{j_i}(\mathbf{x}_i) \leq 0,\ j_i = m_{i-1}+1, \ldots, m_i\}$ the set of private constraint of block i. Problem (9) is motivated by several applications in signal processing, machine learning, and networking; see, e.g., [26] and references therein for some concrete examples.

4.1 Assumptions on the problem

We study Problem (7) under the following assumptions (for ease of reference we repeat in full Assumption A).

Assumption A′. Suppose that

(A1′) Each $\mathcal{X}_i$ is nonempty, closed, and convex;
(A2′) f is C^1 on an open set containing $\mathcal{X}$;
(A3′) $\nabla_{\mathbf{x}_i} f$ is Lipschitz continuous on $\mathcal{X}$ with constant L_{f_i};

(A4′) $G(\mathbf{x}) \triangleq \sum_i g_i(\mathbf{x}_i)$. All $g_i(\mathbf{x}_i)$ are continuous, convex, (possibly nondifferentiable), and Lipschitz continuous with constant L_g on $\mathcal{X}_i$ (possibly nondifferentiable);
(A5′) $\mathcal{K}$ is compact;
(A6′) Each c_{j_i} is continuously differentiable on $\mathcal{X}_i$.
Assumptions A1′-A4′ are the same as A1-A4 (cf. Sec. 2); A5′ is stronger than A5; while A6′ is a standard differentiability requirement. In order to study convergence in the presence of nonconvex constraints, we also need some *regularity* of the constraints.

Definition 1 A point $\bar{\mathbf{x}} \in \mathcal{K}$ satisfies the Mangasarian-Fromovitz Constraint Qualification (MFCQ) if the following implication is satisfied:

$$\left.\begin{array}{l} \mathbf{0} \in \sum_{i=1}^{N} \sum_{j_i \in \bar{J}_i} \mu_{j_i} \nabla_{\mathbf{x}} c_{j_i}(\bar{\mathbf{x}}_i) + N_{\mathcal{X}}(\bar{\mathbf{x}}) \\ \mu_{j_i} \geq 0,\ \forall j \in \bar{J}_i,\ \forall i \in \mathcal{N} \end{array}\right\} \Rightarrow \mu_{j_i} = 0,\ \forall j \in \bar{J}_i,\ \forall i \in \mathcal{N}, \quad (8)$$

where $N_{\mathcal{X}}(\bar{\mathbf{x}}) \triangleq \{\mathbf{z} \in \mathcal{X} : \mathbf{z}^{\mathrm{T}}(\mathbf{y} - \bar{\mathbf{x}}) \leq 0,\ \forall \mathbf{y} \in \mathcal{X}\}$ is the normal cone to $\mathcal{X}$ at $\bar{\mathbf{x}}$, and $\bar{J}_i \triangleq \{j \in \{m_{i-1}+1, \ldots, m_i\} : c_j(\bar{\mathbf{x}}_i) = 0\}$ is the index set of nonconvex constraints that are active at $\bar{\mathbf{x}}$.

According to the above definition, we require the following extra assumption:
Assumption A′ (cont'd)
(A7′) All feasible points of problem (7) satisfy the MFCQ.
We remark that one could relax this assumption and require regularity only at specific points, but at the cost of more convoluted statements; we leave this task to the reader.

4.2 The algorithmic model

To deal with the nonconvex constraints, our approach still consists in leveraging SCA techniques: at each iteration, a suitably chosen "convex approximation" of the original problem is solved. The difference with the method introduced in Sec. 2.2 is that now we will approximate also the nonconvex constraints. More specifically, the subproblem solved to update block i reads [cf. (2)]: given $\mathbf{x}^k$,

$$\hat{\mathbf{x}}_i(\mathbf{x}^k) \triangleq \underset{\mathbf{x}_i \in \mathcal{K}_i(\mathbf{x}_i^k)}{\arg\min}\ \tilde{F}_i(\mathbf{x}_i; \mathbf{x}^k) \triangleq \tilde{f}_i(\mathbf{x}_i; \mathbf{x}^k) + g_i(\mathbf{x}_i), \qquad i = 1, \ldots, N, \quad (9)$$

where $\tilde{f}_i : \mathcal{X}_i \times \mathcal{K} \to \mathbb{R}$ is a convex surrogate of f_i, and $\mathcal{K}_i(\mathbf{x}_i^k)$ is a convex approximation of $\mathcal{K}_i$ at $\mathbf{x}^k$, defined as

$$\mathcal{K}_i(\mathbf{x}_i^k) \triangleq \{\mathbf{x}_i \in \mathcal{X}_i : \tilde{c}_{j_i}(\mathbf{x}_i; \mathbf{x}_i^k) \leq 0,\ j_i = m_{i-1}+1, \ldots, m_i\}$$

where $\tilde{c}_{j_i} : \mathcal{X}_i \times \mathcal{K}_i \to \mathbb{R}$ is a suitably chosen surrogate of c_{j_i}.
The assumptions we require on the surrogate functions $\tilde{f}_i$ and $\tilde{c}_{j_i}$ are very similar to those introduced in Sec. 2. More specifically, we have the following.
Assumption B′ (On the surrogate functions $\tilde{f}_i$'s). Suppose that:

(B1′) $\tilde{f}_i(\bullet;\mathbf{x}^k)$ is uniformly strongly convex and continuously differentiable on $\mathcal{X}_i$ for all $\mathbf{x}^k \in \mathcal{K}$ with a strong convexity constant $c_{\tilde{f}}$ which is independent of i and k;

(B2′) $\nabla\tilde{f}_i(\mathbf{x}_i^k;\mathbf{x}^k) = \nabla_{\mathbf{x}_i} f(\mathbf{x}^k)$, for all $\mathbf{x}^k \in \mathcal{K}$;

(B3′) $\nabla\tilde{f}_i(\mathbf{x}_i^k;\bullet)$ is Lipschitz continuous on $\mathcal{K}$, for all $\mathbf{x}_i^k \in \mathcal{X}_i$, with a Lipschitz constant L_B which is independent of i and k;

(B4′) $\nabla\tilde{f}_i(\bullet;\mathbf{x}^k)$ is Lipschitz continuous on $\mathcal{X}_i$, for all $\mathbf{x}^k \in \mathcal{K}$, with a Lipschitz constant L_E which is independent of i and k.

Note that B1′-B3′ are the same as B1-B3, and now also B4′ is required. The functions $\tilde{c}_{j_i}$ are chosen to satisfy the following (we denote by $\nabla\tilde{c}_{j_i}$ the partial gradient of $\tilde{c}_{j_i}$ with respect to the first argument).

Assumption E (On the surrogate functions $\tilde{c}_{j_i}$'s).

(E1) $\tilde{c}_{j_i}(\bullet;\mathbf{y})$ is convex on $\mathcal{X}_i$ for all $\mathbf{y} \in \mathcal{K}_i$;

(E2) $\tilde{c}_{j_i}(\mathbf{y};\mathbf{y}) = c_{j_i}(\mathbf{y})$, for all $\mathbf{y} \in \mathcal{K}_i$;

(E3) $c_{j_i}(\mathbf{z}) \leq \tilde{c}_{j_i}(\mathbf{z};\mathbf{y})$ for all $\mathbf{z} \in \mathcal{X}_i$ and $\mathbf{y} \in \mathcal{K}_i$;

(E4) $\tilde{c}_{j_i}(\bullet;\bullet)$ is continuous on $\mathcal{X}_i \times \mathcal{K}_i$;

(E5) $\nabla_{\mathbf{y}_i} c_{j_i}(\mathbf{y}) = \nabla\tilde{c}_{j_i}(\mathbf{y};\mathbf{y})$, for all $\mathbf{y} \in \mathcal{K}_i$;

(E6) $\nabla\tilde{c}_{j_i}(\bullet;\bullet)$ is continuous on $\mathcal{X}_i \times \mathcal{K}_i$;

(E7) Each $\tilde{c}_{j_i}(\bullet;\bullet)$ is Lipschitz continuous on $\mathcal{X}_i \times \mathcal{K}_i$

These assumptions may look cumbersome, but it is quite easy to find suitable $\tilde{c}$'s in most practical problems; we refer the reader to [25] for several examples and applications. Here, we only remark that E2-E3 say that $\tilde{c}_{j_i}$ must be an *upper* approximation of c_{j_i} that agrees with c_{j_i} in the base point $\mathbf{y}$. This guarantees that $\mathcal{K}_i(\mathbf{x}_i^k)$ is an inner (convex) approximation of $\mathcal{K}_i$, which is a key property to preserve feasibility of the iterates and avoid the use of complex techniques, like penalty functions for example, to guarantee eventual feasibility; indeed, the use of such complex tools would be extremely problematic in an asynchronous environment.

AsyFLEXA-NCC: We are now ready to present the algorithm to solve Problem (7): it is simply Algorithm 1 wherein the best-response map in S.2 is replaced by (9); we term it AsyFLEXA-NCC (where NCC stands for Non Convex Constraints). Note that, in order to preserve feasibility of the iterates, we need $\mathbf{x}_i^k = \tilde{\mathbf{x}}_i^k$, for any $i \in \mathcal{N}$ and $k \geq 0$ [cf. Corollary 1(ii)]; see remark after Corollary 1, Sec. 3, for a discussion on this constraint and its implementation in practice. Note that the probabilistic model concerning the choice of the index and the delay vector at each iteration is the same as the one we used in the case of convex constraints, see Sec. 2.3. Convergence of AsyFLEXA-NCC is stated in the following theorem.

Theorem 2 *Let Problem (7) be given, along with AsyFLEXA-NCC and the stochastic process $\underline{\boldsymbol{\omega}}$. Let $\{\mathbf{x}^k\}$ be the sequence generated by the algorithm, given $\mathbf{x}^0 \in \mathcal{X}$. Suppose that Assumptions C-E, Assumptions A′-B′ and condition ii) of Corollary 1 hold. Then the following hold a.s.:*

(a) Feasibility: $\mathbf{x}^k \in \mathcal{K}_1(\mathbf{x}_1^k) \times \ldots \times \mathcal{K}_N(\mathbf{x}_N^k) \subseteq \mathcal{K}$ for all $k \geq 0$;

(b) Convergence: $\{\mathbf{x}^k\}$ *is bounded and every of its limit point is a stationary solution of Problem* (7)

Proof See Appendix.

5 Conclusions

We proposed a novel model for the parallel asynchronous minimization of the sum of a nonconvex smooth function and a convex nonsmooth one subject to nonconvex constraints. Our model captures the essential features of modern multi-core architectures by providing a more realistic description of asynchronous methods with inconsistent read that those currently available. Building on this model, we proved almost sure convergence of the proposed parallel asynchronous scheme. Complexity and numerical results are provided in the companion paper [5].

References

1. H. Avron, A. Druinsky, and A. Gupta. Revisiting asynchronous linear solvers: Provable convergence rate through randomization. *Journal of the ACM (JACM)*, 62(6):51, 2015.
2. G. M. Baudet. Asynchronous iterative methods for multiprocessors. *Journal of the ACM (JACM)*, 25(2):226–244, 1978.
3. D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and distributed computation: numerical methods*, volume 23. Prentice hall Englewood Cliffs, NJ, 1989.
4. D.P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, USA, 2th Ed., 1999.
5. L. Cannelli, F. Facchinei, V. Kungurtsev, and G. Scutari. Asynchronous parallel algorithms for nonconvex big-data optimization: Complexity and numerical results. *Submitted to Mathematical Programming*, 2017.
6. D. Chazan and W. Miranker. Chaotic relaxation. *Linear algebra and its applications*, 2(2):199–222, 1969.
7. D. Davis. The asynchronous palm algorithm for nonsmooth nonconvex problems. *arXiv preprint arXiv:1604.00526*, 2016.
8. D. Davis, B. Edmunds, and M. Udell. The sound of apalm clapping: Faster nonsmooth nonconvex optimization with stochastic asynchronous palm. *arXiv preprint arXiv:1606.02338*, 2016.
9. R. Durrett. *Probability: theory and examples*. Cambridge university press, 2010.
10. F. Facchinei, G. Scutari, and S. Sagratella. Parallel selective algorithms for nonconvex big data optimization. *IEEE Transactions on Signal Processing*, 63(7):1874–1889, 2015.
11. A. Frommer and D. B. Szyld. On asynchronous iterations. *Journal of computational and applied mathematics*, 123(1):201–216, 2000.
12. M. Hong. A distributed, asynchronous and incremental algorithm for nonconvex optimization: An admm based approach. *arXiv preprint arXiv:1412.6058*, 2014.
13. Z. Huo and H. Huang. Asynchronous stochastic gradient descent with variance reduction for non-convex optimization. *arXiv preprint arXiv:1604.03584*, 2016.
14. F. Iutzeler, P. Bianchi, P. Ciblat, and W. Hachem. Asynchronous distributed optimization using a randomized alternating direction method of multipliers. In *52nd IEEE Conference on Decision and Control*, pages 3671–3676. IEEE, 2013.
15. A. Klenke. *Probability theory: a comprehensive course*. Springer Science & Business Media, 2013.
16. X. Lian, Y. Huang, Y. Li, and J. Liu. Asynchronous parallel stochastic gradient for nonconvex optimization. In *Advances in Neural Information Processing Systems*, pages 2719–2727, 2015.

17. J. Liu and S. J. Wright. Asynchronous stochastic coordinate descent: Parallelism and convergence properties. *SIAM Journal on Optimization*, 25(1):351–376, 2015.
18. J. Liu, S. J. Wright, C. Ré, V. Bittorf, and S. Sridhar. An asynchronous parallel stochastic coordinate descent algorithm. *The Journal of Machine Learning Research*, 16(1):285–322, 2015.
19. H. Mania, X. Pan, D. Papailiopoulos, B. Recht, K. Ramchandran, and M. I. Jordan. Perturbed iterate analysis for asynchronous stochastic optimization. *arXiv:1507.06970*, 2016.
20. A. Nedić, D. P. Bertsekas, and V. S. Borkar. Distributed asynchronous incremental subgradient methods. *Studies in Computational Mathematics*, 8:381–407, 2001.
21. Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.
22. F. Niu, B. Recht, C. Re, and S. J. Wright. Hogwild: a lock-free approach to parallelizing stochastic gradient descent. *Advances in Neural Information Processing Systems*, pages 693–701, 2011.
23. Z. Peng, Y. Xu, M. Yan, and W. Yin. Arock: an algorithmic framework for asynchronous parallel coordinate updates. *arXiv preprint arXiv:1506.02396*, 2015.
24. H. Robbins and D. Siegmund. A convergence theorem for non negative almost supermartingales and some applications. In *Herbert Robbins Selected Papers*, pages 111–135. Springer, 1985.
25. G. Scutari, F. Facchinei, and L. Lampariello. Parallel and distributed methods for constrained nonconvex optimization-part i: Theory. *IEEE Transactions on Signal Processing, published online DOI: 10.1109/TSP.2016.2637317*, 2016.
26. G. Scutari, F. Facchinei, L. Lampariello, S. Sardellitti, and P. Song. Parallel and distributed methods for constrained nonconvex optimization–part ii: Applications in communications and machine learning. *IEEE Transactions on Signal Processing, published online DOI: 10.1109/TSP.2016.2637314*, 2016.
27. G. Scutari, F. Facchinei, P. Song, D. P. Palomar, and J.-S. Pang. Decomposition by partial linearization: Parallel optimization of multi-agent systems. *IEEE Transactions on Signal Processing*, 62(3):641–656, 2014.
28. E. Wei and A. Ozdaglar. On the o (1= k) convergence of asynchronous distributed alternating direction method of multipliers. In *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*, pages 551–554. IEEE, 2013.
29. W. H. Young. On classes of summable functions and their fourier series. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 87(594):225–229, 1912.

6 Appendix

6.1 Preliminaries

We first introduce some preliminary definitions and results that will be instrumental to prove Theorem 1 and Theorem 2.

In the rest of the Appendix it will be convenient to use the following notation for the random variables and their realizations: underlined symbols denote random variables, e.g., $\underline{\mathbf{x}}^k$, $\underline{\tilde{\mathbf{x}}}^k$ whereas the same symbols with no underline are the corresponding realizations, e.g., $\mathbf{x}^k \triangleq \underline{\mathbf{x}}^k(\omega)$ and $\tilde{\mathbf{x}}^k \triangleq \underline{\tilde{\mathbf{x}}}^k(\omega)$.

1. On conditional probabilities. We begin with a simple result concerning conditional expectations in our setting. In our developments we will consider the conditional expectation of random variables $\mathbf{Z}$ on Ω of the type $\mathbb{E}(\mathbf{Z}|\mathcal{F}^k)$. The following simple fact holds.

Proposition 1 *let Z be a random variable defined on Ω which actually is fully determined by $\boldsymbol{\omega}^{0:k+1}$, and let $\mathcal{F}^k$ be defined by* (5)*. Then*

$$\mathbb{E}(\mathbf{Z}|\mathcal{F}^k)(\omega) = \sum_{(i,\mathbf{d})\in\mathcal{N}\times\mathcal{D}} p((i,\mathbf{d})\,|\boldsymbol{\omega}^{0:k})\mathbf{Z}((i,\mathbf{d}),\boldsymbol{\omega}^{0:k}) \tag{10}$$

Proof Recall that $\mathcal{F}^k$ is the $\sigma-$algebra generated by $\mathcal{C}^k$ which is a finite partition of Ω. Therefore, see for example [9, Example 5.1.3], we can write

$$\mathbb{E}(\mathbf{Z}|\mathcal{F}^k) = \frac{\mathbb{E}(\mathbf{Z};C^k(\boldsymbol{\omega}^{0:k}))}{\mathbb{P}(C^k(\boldsymbol{\omega}^{0:k}))} \qquad \text{on } C^k(\boldsymbol{\omega}^{0:k}).$$

Given $\omega\in\Omega$, we have $\omega\in C^k(\boldsymbol{\omega}^{0:k})$ and therefore, taking into account that Z actually depends only on $\boldsymbol{\omega}^{0:k+1}$ and can only take a finite number of values, and recalling (6), the thesis easily follows.

2. Properties of the best response $\hat{\mathbf{x}}(\bullet)$. We introduce next some basic
properties of the best-response maps defined in (2) and (9).

Proposition 2 ([10]) *Let us define the best-response map: $\hat{\mathbf{x}}(\bullet)\triangleq(\hat{\mathbf{x}}_i(\bullet))_{i=1}^N$ where $\hat{\mathbf{x}}_i(\bullet)$ is defined in* (2)*, for every $i=1,\dots,N$. Under Assumptions A-B, the following hold.*
(a) [Optimality]: For any $i\in\mathcal{N}$ and $\mathbf{y}\in\mathcal{X}$,

$$(\hat{\mathbf{x}}_i(\mathbf{y})-\mathbf{y}_i)^T\nabla_{\mathbf{y}_i}f(\mathbf{y})+g_i(\hat{\mathbf{x}}_i(\mathbf{y}))-g_i(\mathbf{y}_i)\leq -c_{\tilde{f}}\|\hat{\mathbf{x}}_i(\mathbf{y})-\mathbf{y}_i\|_2^2\,. \tag{11}$$

(b) [Lipschitz continuity]: For any $i\in\mathcal{N}$ and $\mathbf{y},\mathbf{z}\in\mathcal{X}$,

$$\|\hat{\mathbf{x}}_i(\mathbf{y})-\hat{\mathbf{x}}_i(\mathbf{z})\|_2\leq L_{\hat{\mathbf{x}}}\|\mathbf{y}-\mathbf{z}\|_2, \tag{12}$$

with $L_{\hat{\mathbf{x}}}=\frac{L_B}{c_{\tilde{f}}}$.
(c) [Fixed-point characterization]: The set of fixed-points of $\hat{\mathbf{x}}(\bullet)$ coincides with the set of stationary solutions of the optimization problem (1)*. Therefore $\hat{\mathbf{x}}(\bullet)$ has at least one fixed point.*
(d) [Boundedness]: If $\nabla_{\mathbf{x}_i}f$ is bounded on $\mathcal{X}_i$ for some $L_{\nabla_i}\in(0,+\infty)$, then so is:

$$\|\hat{\mathbf{x}}_i(\mathbf{y})-\mathbf{y}_i\|_2^2\leq M_i \tag{13}$$

for any $i\in\mathcal{N}$ and $\mathbf{y}\in\mathcal{X}$, with $M_i=\frac{L_{\nabla_i}+L_g}{c_{\tilde{f}}}$

Proposition 3 ([25]) *Let us define the best-response map: $\hat{\mathbf{x}}(\bullet)\triangleq(\hat{\mathbf{x}}_i(\bullet))_{i=1}^N$ where $\hat{\mathbf{x}}_i(\bullet)$ is defined in* (9) *for every $i=1,\dots,N$. Under Assumptions A′-B′, E the following hold.*
(a) [Optimality]: For any $i\in\mathcal{N}$ and $\mathbf{y}\in\mathcal{K}$,

$$(\hat{\mathbf{x}}_i(\mathbf{y})-\mathbf{y}_i)^T\nabla_{\mathbf{y}_i}f(\mathbf{y})+g_i(\hat{\mathbf{x}}_i(\mathbf{y}))-g_i(\mathbf{y}_i)\leq -c_{\tilde{f}}\|\hat{\mathbf{x}}_i(\mathbf{y})-\mathbf{y}_i\|_2^2\,. \tag{14}$$

(b) [Lipschitz continuity]: If, in addition, $\mathcal{K}$ is compact, for any $i \in \mathcal{N}$ and $\mathbf{y}, \mathbf{z} \in \mathcal{K}$:

$$\|\hat{\mathbf{x}}_i(\mathbf{y}) - \hat{\mathbf{x}}_i(\mathbf{z})\|_2 \leq \tilde{L}\|\mathbf{y} - \mathbf{z}\|_2^{1/2}, \tag{15}$$

with $\tilde{L} > 0$.

3. Young's Inequality [29]. Consider the Young's inequality in the following form:

$$\mu_1\mu_2 \leq \frac{1}{2}(\alpha\mu_1^2 + \alpha^{-1}\mu_2^2)\,, \tag{16}$$

for any $\alpha, \mu_1, \mu_2 > 0$.

4. Martingale Theorem. To prove our main convergence results, we leverage the following standard convergence theorem for non-negative almost supermartingales.

Theorem 3 ([24]) *Let $(\Omega, \mathcal{T}, \mathbb{P})$ be a probability space. Let $\mathscr{T} = \{\mathcal{T}^k\}$ be a sequence of sub-sigma algebras of $\mathcal{T}$ such that $\forall k \geq 0$, $\mathcal{T}^k \subset \mathcal{T}^{k+1}$. Define $l_+(\mathscr{T})$ as the set of sequences of $[0, +\infty)$-valued random variables $\{\underline{\xi}^k\}$, where $\underline{\xi}^k$ is $\mathcal{T}^k$-measurable, and $l_+^1(\mathscr{T}) \coloneqq \{\{\underline{\xi}^k\} \in l_+(\mathscr{T}) | \sum_{k=1}^{+\infty} \underline{\xi}^k < +\infty \quad a.s.\}$. Let $\{\underline{\alpha}^k\}$, $\{\underline{\nu}^k\} \in l_+(\mathscr{T})$, and $\{\underline{\eta}^k\} \in l_+^1(\mathscr{T})$ be such that*

$$\mathbb{E}(\underline{\alpha}^{k+1}|\mathcal{T}^k) + \underline{\nu}^k \leq \underline{\alpha}^k + \underline{\eta}^k. \tag{17}$$

Then $\{\underline{\nu}^k\} \in l_+^1(\mathscr{T})$, and $\{\underline{\alpha}^k\}$ converges to a $[0, +\infty)$-valued random variable almost surely. □

5. Further definitions. We introduce the following two sets. Given $\omega = \{(i^k, \mathbf{d}^k)\} \in \Omega$ and some $\delta(\omega) > 0$, let

$$\mathcal{V}^k(\omega) = \left\{(i, \mathbf{d}) \in \mathcal{N} \times \mathcal{D} : p\left((i, \mathbf{d})|\boldsymbol{\omega}^{0:k-1}\right) \geq \delta(\omega)\right\}, \tag{18}$$

and

$$\tilde{\mathcal{V}}^k(\omega) \triangleq \left\{((j, \mathbf{d}_j))_{j=1}^N \in \mathcal{V}^k(\omega) : (i, \mathbf{d}_i) = (i^k, \mathbf{d}^k) \text{ for some } i\right\}. \tag{19}$$

Note that $\tilde{\mathcal{V}}^k(\omega)$ is not uniquely defined when there exist multiple vectors $\mathbf{d}_j$ such that $(j, \mathbf{d}_j) \in \mathcal{V}^k(\omega)$, with $j \neq i$. In such a case, the specific choice of $\mathbf{d}_j$ is irrelevant for our developments; however, to eliminate any ambiguity in the definition (19) we tacitly assume that the aforementioned $\mathbf{d}_j$ chosen in (19) is the one that maximizes $p((j, \bullet)|\boldsymbol{\omega}^{0:k-1})$.

Definition 2 The set $\mathcal{V}^k(\omega)$ is said to *cover* $\mathcal{N}$ if $(i, \mathbf{d}) \in \mathcal{V}^k(\omega)$, for some $\mathbf{d} \in \mathcal{D}$ and all $i \in \mathcal{N}$.

Note that under C3, there exists a $\bar{\Omega} \subseteq \Omega$, with $\mathbb{P}(\bar{\Omega}) = 1$, such that the following holds: for every $\omega \in \bar{\Omega}$, one can find a $\delta(\omega) > 0$, such that the resulting $\mathcal{V}^k(\omega)$ covers $\mathcal{N}$ and consequently $\tilde{\mathcal{V}}^k(\omega)$ has cardinality N.

6. Inconsistent read. For any given $\omega \in \Omega$, recall that for simplicity of notation we define: $\tilde{\mathbf{x}}^k = \mathbf{x}^{k-\mathbf{d}^k}$. Since at each iteration only one block of variables is updated, it is not difficult to see that $\tilde{\mathbf{x}}^k$ can be written as

$$\tilde{\mathbf{x}}^k = \mathbf{x}^k + \sum_{l \in K(\mathbf{d}^k)} (\mathbf{x}^l - \mathbf{x}^{l+1}), \tag{20}$$

where $K(\mathbf{d}^k) \subseteq \{k-\delta, \dots, k-1\}$ [cf. Assumption C1]. When we need to explicitly specify the dependence of $\tilde{\mathbf{x}}^k$ on a given realization $\mathbf{d}$ of $\underline{\mathbf{d}}^k$, we will write $\tilde{\mathbf{x}}^k(\mathbf{d})$.

6.2 Proof of Theorem 1

We prove the theorem only under condition i) of Corollary 1. The other cases can be similarly studied. In the rest of this proof, the best-response map $\hat{\mathbf{x}}(\bullet)$ is the one defined in (2).

For any given realization $\omega \in \Omega$ and $k \geq \delta$, the following holds:

$$\begin{aligned}
F(\mathbf{x}^{k+1}) &= f(\mathbf{x}^{k+1}) + g(\mathbf{x}^{k+1}) \overset{(a)}{=} f(\mathbf{x}^{k+1}) + \sum_{i \neq i^k} g_i(\mathbf{x}_i^{k+1}) + g_{i^k}(\mathbf{x}_{i^k}^{k+1}) \\
&\overset{(b)}{=} f(\mathbf{x}^{k+1}) + \sum_{i \neq i^k} g_i(\mathbf{x}_i^{k}) + g_{i^k}(\mathbf{x}_{i^k}^{k+1}) \\
&\overset{(c)}{\leq} f(\mathbf{x}^k) + \gamma^k \nabla_{\mathbf{x}_{i^k}} f(\mathbf{x}^k)^{\mathrm{T}} (\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{i^k}^k) + \frac{(\gamma^k)^2 L_f}{2} \|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{i^k}^k\|_2^2 \\
&+ \sum_{i \neq i^k} g_i(\mathbf{x}_i^k) + g_{i^k}(\mathbf{x}_{i^k}^{k+1}) = f(\mathbf{x}^k) + \gamma^k \nabla_{\mathbf{x}_{i^k}} f(\tilde{\mathbf{x}}^k)^{\mathrm{T}} (\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{i^k}^k) \qquad (21) \\
&+ (\nabla_{\mathbf{x}_{i^k}} f(\mathbf{x}^k) - \nabla_{\mathbf{x}_{i^k}} f(\tilde{\mathbf{x}}^k))^{\mathrm{T}} (\mathbf{x}_{i^k}^{k+1} - \mathbf{x}_{i^k}^k) + \frac{(\gamma^k)^2 L_f}{2} \|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{i^k}^k\|_2^2 \\
&+ \sum_{i \neq i^k} g_i(\mathbf{x}_i^k) + g_{i^k}(\mathbf{x}_{i^k}^{k+1}) \overset{(d)}{\leq} f(\mathbf{x}^k) + \gamma^k \nabla_{\mathbf{x}_{i^k}} f(\tilde{\mathbf{x}}^k)^{\mathrm{T}} (\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \tilde{\mathbf{x}}_{i^k}^k) \\
&+ (\nabla_{\mathbf{x}_{i^k}} f(\mathbf{x}^k) - \nabla_{\mathbf{x}_{i^k}} f(\tilde{\mathbf{x}}^k))^{\mathrm{T}} (\mathbf{x}_{i^k}^{k+1} - \mathbf{x}_{i^k}^k) + \gamma^k \nabla_{\mathbf{x}_{i^k}} f(\tilde{\mathbf{x}}^k)^{\mathrm{T}} (\tilde{\mathbf{x}}_{i^k}^k - \mathbf{x}_{i^k}^k) \\
&+ \frac{(\gamma^k)^2 L_f}{2} \|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{i^k}^k\|_2^2 + \sum_{i \neq i^k} g_i(\mathbf{x}_i^k) + \gamma^k g_{i^k}(\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k)) + (1-\gamma^k) g_{i^k}(\mathbf{x}_{i^k}^k) \\
&+ \gamma^k (g_{i^k}(\tilde{\mathbf{x}}_{i^k}^k) - g_{i^k}(\tilde{\mathbf{x}}_{i^k}^k)) \\
&\overset{(e)}{\leq} F(\mathbf{x}^k) - \gamma^k c_{\bar{f}} \|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \tilde{\mathbf{x}}_{i^k}^k\|_2^2 + L_f \|\mathbf{x}^k - \tilde{\mathbf{x}}^k\|_2 \|\mathbf{x}_{i^k}^{k+1} - \mathbf{x}_{i^k}^k\|_2 \\
&+ \gamma^k (L_\nabla + L_g) \|\mathbf{x}^k - \tilde{\mathbf{x}}^k\|_2 + \frac{(\gamma^k)^2 L_f}{2} \|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{i^k}^k\|_2^2 \\
&\overset{(f)}{\leq} F(\mathbf{x}^k) - \gamma^k c_{\bar{f}} \|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \tilde{\mathbf{x}}_{i^k}^k\|_2^2 + \frac{L_f}{2} \|\mathbf{x}^k - \tilde{\mathbf{x}}^k\|_2^2 + \frac{L_f}{2} \|\mathbf{x}_{i^k}^{k+1} - \mathbf{x}_{i^k}^k\|_2^2 \\
&+ \gamma^k (L_\nabla + L_g) \|\mathbf{x}^k - \tilde{\mathbf{x}}^k\|_2 + \frac{(\gamma^k)^2 L_f}{2} \|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{i^k}^k\|_2^2
\end{aligned}$$

$$= F(\mathbf{x}^k) - \gamma^k c_{\tilde{f}} \|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \tilde{\mathbf{x}}_{i^k}^k\|_2^2 + \frac{L_f}{2}\|\mathbf{x}^k - \tilde{\mathbf{x}}^k\|_2^2$$
$$+ (\gamma^k)^2 L_f \|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{i^k}^k\|_2^2 + \gamma^k (L_\nabla + L_g)\|\mathbf{x}^k - \tilde{\mathbf{x}}^k\|_2$$

where in (a) we used the separability of g; (b) follows from the updating rule of the algorithm; in (c) we applied the Descent Lemma [3] on f with $L_f = \max_{i\in\mathcal{N}} L_{f_i}$; (d) comes from the convexity of g_i; in (e) we used Proposition 2, Assumptions A3-A4, and condition i) of Corollary 1, with $L_\nabla = \max_{i\in\mathcal{N}} L_{\nabla_i}$; and (f) is due to the Young's inequality, with $\alpha = 1$.
We bound now the term $-\|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \tilde{\mathbf{x}}_{i^k}^k\|_2^2$ in (21). We have:

$$\begin{aligned}
&-\|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \tilde{\mathbf{x}}_{i^k}^k\|_2^2 = -\|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{i^k}^k + \mathbf{x}_{i^k}^k - \tilde{\mathbf{x}}_{i^k}^k\|_2^2 \\
&\leq -\left(\|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{i^k}^k\|_2^2 + \|\mathbf{x}_{i^k}^k - \tilde{\mathbf{x}}_{i^k}^k\|_2^2\right) + 2\|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{i^k}^k\|_2\|\mathbf{x}_{i^k}^k - \tilde{\mathbf{x}}_{i^k}^k\|_2 \\
&\leq -\|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{i^k}^k\|_2^2 + \alpha\|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{i^k}^k\|_2^2 + \alpha^{-1}\|\mathbf{x}_{i^k}^k - \tilde{\mathbf{x}}_{i^k}^k\|_2^2,
\end{aligned} \tag{22}$$

where α is any given positive constant. Substituting (22) in (21) we have:

$$\begin{aligned}
&F(\mathbf{x}^{k+1}) \leq F(\mathbf{x}^k) - \gamma^k c_{\tilde{f}}(1-\alpha)\|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{i^k}^k\|_2^2 + \gamma^k c_{\tilde{f}}\alpha^{-1}\|\mathbf{x}_{i^k}^k - \tilde{\mathbf{x}}_{i^k}^k\|_2^2 \\
&+ \frac{L_f}{2}\|\mathbf{x}^k - \tilde{\mathbf{x}}^k\|_2^2 + (\gamma^k)^2 L_f\|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{i^k}^k\|_2^2 + \gamma^k(L_\nabla + L_g)\|\mathbf{x}^k - \tilde{\mathbf{x}}^k\|_2
\end{aligned} \tag{23}$$
$$\begin{aligned}
&\leq F(\mathbf{x}^k) - \gamma^k(c_{\tilde{f}}(1-\alpha) - \gamma^k L_f)\|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{i^k}^k\|_2^2 \\
&+ \left(\frac{L_f}{2} + \gamma^k c_{\tilde{f}}\alpha^{-1}\right)\|\mathbf{x}^k - \tilde{\mathbf{x}}^k\|_2^2 + \gamma^k(L_\nabla + L_g)\|\mathbf{x}^k - \tilde{\mathbf{x}}^k\|_2
\end{aligned}$$

We bound $\|\mathbf{x}^k - \tilde{\mathbf{x}}^k\|_2^2$ as follows:

$$\begin{aligned}
&\|\mathbf{x}^k - \tilde{\mathbf{x}}^k\|_2^2 \overset{(a)}{\leq} \left(\sum_{l=k-\delta}^{k-1}\|\mathbf{x}^{l+1} - \mathbf{x}^l\|_2\right)^2 \overset{(b)}{\leq} \delta\sum_{l=k-\delta}^{k-1}\|\mathbf{x}^{l+1} - \mathbf{x}^l\|_2^2 \\
&= \delta\left(\sum_{l=k-\delta}^{k-1}(l-(k-1)+\delta)\|\mathbf{x}^{l+1} - \mathbf{x}^l\|_2^2 - \sum_{l=k+1-\delta}^{k}(l-k+\delta)\|\mathbf{x}^{l+1} - \mathbf{x}^l\|_2^2\right) \\
&+ \delta^2\|\mathbf{x}^{k+1} - \mathbf{x}^k\|_2^2
\end{aligned} \tag{24}$$
$$\begin{aligned}
&= \delta\left(\sum_{l=k-\delta}^{k-1}(l-(k-1)+\delta)\|\mathbf{x}^{l+1} - \mathbf{x}^l\|_2^2 - \sum_{l=k+1-\delta}^{k}(l-k+\delta)\|\mathbf{x}^{l+1} - \mathbf{x}^l\|_2^2\right) \\
&+ \delta^2(\gamma^k)^2\|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{i^k}^k\|_2^2
\end{aligned}$$

where (a) comes from (20) and (b) is due to the Jensen's inequality. Similarly we have:

$$\|\mathbf{x}^k - \tilde{\mathbf{x}}^k\|_2 \leq \sum_{l=k-\delta}^{k-1}\|\mathbf{x}^l - \mathbf{x}^{l+1}\|_2$$

$$
\begin{aligned}
&= \sum_{l=k-\delta}^{k-1} (l-(k-1)+\delta)\|\mathbf{x}^{l+1}-\mathbf{x}^{l}\|_2 - \sum_{l=k+1-\delta}^{k} (l-k+\delta)\|\mathbf{x}^{l+1}-\mathbf{x}^{l}\|_2 \\
&+ \delta\|\mathbf{x}^{k+1}-\mathbf{x}^{k}\|_2 \\
&= \sum_{l=k-\delta}^{k-1} (l-(k-1)+\delta)\|\mathbf{x}^{l+1}-\mathbf{x}^{l}\|_2 - \sum_{l=k+1-\delta}^{k} (l-k+\delta)\|\mathbf{x}^{l+1}-\mathbf{x}^{l}\|_2 \quad (25) \\
&+ \delta\gamma^k\|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k)-\mathbf{x}^k_{i^k}\|_2 \\
&\leq \sum_{l=k-\delta}^{k-1} (l-(k-1)+\delta)\|\mathbf{x}^{l+1}-\mathbf{x}^{l}\|_2 - \sum_{l=k+1-\delta}^{k} (l-k+\delta)\|\mathbf{x}^{l+1}-\mathbf{x}^{l}\|_2 \\
&+ \delta\gamma^k\|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k)-\tilde{\mathbf{x}}^k_{i^k}\|_2 + \delta\gamma^k\|\mathbf{x}^k-\tilde{\mathbf{x}}^k\|_2 \\
&\overset{(a)}{\leq} \sum_{l=k-\delta}^{k-1} (l-(k-1)+\delta)\|\mathbf{x}^{l+1}-\mathbf{x}^{l}\|_2 - \sum_{l=k+1-\delta}^{k} (l-k+\delta)\|\mathbf{x}^{l+1}-\mathbf{x}^{l}\|_2 + \delta\gamma^k M \\
&+ \delta\gamma^k\|\mathbf{x}^k-\tilde{\mathbf{x}}^k\|_2,
\end{aligned}
$$

where (a) comes from Proposition 2 with $M = \max_{i\in\mathcal{N}} M_i$. We can rearrange the terms in (25) to get:

$$
\begin{aligned}
&(1-\delta\gamma^k)\|\mathbf{x}^k-\tilde{\mathbf{x}}^k\|_2 \\
&\leq \sum_{l=k-\delta}^{k-1} (l-(k-1)+\delta)\|\mathbf{x}^{l+1}-\mathbf{x}^{l}\|_2 - \sum_{l=k+1-\delta}^{k} (l-k+\delta)\|\mathbf{x}^{l+1}-\mathbf{x}^{l}\|_2 + \delta\gamma^k M
\end{aligned} \quad (26)
$$

Invoking D1, there exists a sufficiently large k, say $\bar{k}$, such that:$(1-\delta\gamma^{\bar{k}-1}) > 0$. For any $k \geq \bar{k}$ we also get: $(1-\delta\gamma^k) > (1-\delta\gamma^{\bar{k}-1}) > 0$. From these considerations and (26) we have:

$$
\begin{aligned}
\|\mathbf{x}^k-\tilde{\mathbf{x}}^k\|_2 \;\leq\; & (1-\delta\gamma^k)^{-1}\Big(\sum_{l=k-\delta}^{k-1} (l-(k-1)+\delta)\|\mathbf{x}^{l+1}-\mathbf{x}^{l}\|_2 \\
& - \sum_{l=k+1-\delta}^{k} (l-k+\delta)\|\mathbf{x}^{l+1}-\mathbf{x}^{l}\|_2 + \delta\gamma^k M\Big) \\
& \leq (1-\delta\gamma^{k-1})^{-1} \sum_{l=k-\delta}^{k-1} (l-(k-1)+\delta)\|\mathbf{x}^{l+1}-\mathbf{x}^{l}\|_2 \quad (27) \\
& - (1-\delta\gamma^k)^{-1} \sum_{l=k+1-\delta}^{k} (l-k+\delta)\|\mathbf{x}^{l+1}-\mathbf{x}^{l}\|_2 + (1-\delta\gamma^k)^{-1}\delta\gamma^k M
\end{aligned}
$$

Let us now define the following Lyapunov function. Let $k \geq \bar{k}$:

$$
\tilde{F}(\mathbf{x}^k,\dots,\mathbf{x}^{k-\delta}) = F(\mathbf{x}^k) + \delta\Big(\frac{L_f}{2} + c_{\tilde{f}}\alpha^{-1}\Big)\left(\sum_{l=k-\delta}^{k-1} (l-(k-1)+\delta)\|\mathbf{x}^{l+1}-\mathbf{x}^{l}\|_2^2\right)
$$

$$+ (L_\nabla + L_g)(1 - \delta\gamma^{k-1})^{-1} \sum_{l=k-\delta}^{k-1} (l - (k-1) + \delta)\|\mathbf{x}^{l+1} - \mathbf{x}^l\|_2$$

Using (24) and (27) in (23) and rearranging the terms, the following holds almost surely: for all $k \geq \bar{k}$

$$\mathbb{E}(\tilde{F}(\underline{\mathbf{x}}^{k+1} \ldots, \mathbf{x}^{k+1-\delta})|\mathcal{F}^{k-1}) \leq \tilde{F}(\underline{\mathbf{x}}^k, \ldots, \underline{\mathbf{x}}^{k-\delta}) \tag{28}$$
$$- \gamma^k(c_{\bar{f}}(1-\alpha) - \gamma^k(L_f + \frac{\delta^2 L_f}{2} + \gamma^k\delta^2 c_{\tilde{f}}\alpha^{-1}))\mathbb{E}(\|\hat{\mathbf{x}}_{\underline{i}^k}(\tilde{\underline{\mathbf{x}}}^k) - \mathbf{x}^k_{\underline{i}^k}\|_2^2|\mathcal{F}^{k-1}) + Z^k,$$

where $Z^k \triangleq \frac{(\gamma^k)^2(L_\nabla + L_g)\delta M}{1-\delta\gamma^k}$. Since: $\lim_{k\to+\infty} \frac{Z^k}{(\gamma^k)^2} = (L_\nabla + L_g)\delta M$ and $\sum_{k=0}^{+\infty}(\gamma^k)^2 < +\infty$, we note that: $\sum_{k=0}^{+\infty} Z^k < +\infty$.

Invoking D1 and choosing $\alpha < 1$, we have that for sufficiently large k there exists some positive constant β_1 such that the following holds almost surely:

$$\begin{aligned}&\mathbb{E}(\tilde{F}(\underline{\mathbf{x}}^{k+1}, \ldots, \mathbf{x}^{k+1-\delta})|\mathcal{F}^{k-1}) \\ &\leq \tilde{F}(\underline{\mathbf{x}}^k, \ldots, \underline{\mathbf{x}}^{k-\delta}) - \beta_1\gamma^k\mathbb{E}(\|\hat{\mathbf{x}}_{\underline{i}^k}(\tilde{\underline{\mathbf{x}}}^k) - \mathbf{x}^k_{\underline{i}^k}\|_2^2|\mathcal{F}^{k-1}) + Z^k.\end{aligned} \tag{29}$$

We can now apply Theorem 3 with the following identifications: $\{\underline{\alpha}^k\} = \{\tilde{F}(\underline{\mathbf{x}}^k, \ldots, \underline{\mathbf{x}}^{k-\delta})\}$, $\{\underline{\eta}^k\} = \{Z^k\}$ and $\{\underline{\nu}^k\} = \{\gamma^k\mathbb{E}(\|\hat{\mathbf{x}}_{\underline{i}^k}(\tilde{\underline{\mathbf{x}}}^k) - \mathbf{x}^k_{\underline{i}^k}\|_2^2|\mathcal{F}^{k-1})\}$ (note that we assumed without loss of generality that $F(\underline{\mathbf{x}}^k) \geq 0$; indeed, since is uniformly bounded from below, one can always find a positive constant c and write instead $F(\underline{\mathbf{x}}^k) + c \geq 0$). We deduce that i) $\{\tilde{F}(\underline{\mathbf{x}}^k, \ldots, \underline{\mathbf{x}}^{k-\delta})\}$ converges almost surely; and ii).

$$\lim_{k\to+\infty} \sum_{t=\bar{k}}^{k} \left(\gamma^t\,\mathbb{E}(\|\hat{\mathbf{x}}_{\underline{i}^t}(\tilde{\underline{\mathbf{x}}}^t) - \mathbf{x}^t_{\underline{i}^t}\|_2^2|\mathcal{F}^{t-1})\right) < +\infty \quad \text{a. s..} \tag{30}$$

Since $\tilde{F}(\underline{\mathbf{x}}^k, \ldots, \underline{\mathbf{x}}^{k-\delta})$ is coercive from condition i) of Corollary 1, this implies that the sequence $\{\mathbf{x}^k\}$ is bounded almost surely.
Since $\sum_{k=0}^{+\infty} \gamma^k = +\infty$, it follows from (30) that there exists a $\bar{\Omega} \subseteq \Omega$, with $\mathbb{P}(\bar{\Omega}) = 1$, such that for any $\omega \in \bar{\Omega}$:

$$\liminf_{k\to+\infty} \mathbb{E}(\|\hat{\mathbf{x}}_{\underline{i}^k}(\tilde{\underline{\mathbf{x}}}^k) - \mathbf{x}^k_{\underline{i}^k}\|_2^2|\mathcal{F}^{k-1})(\omega) = 0. \tag{31}$$

Since we have from Proposition (1):

$$\mathbb{E}(\|\hat{\mathbf{x}}_{\underline{i}^k}(\tilde{\underline{\mathbf{x}}}^k) - \mathbf{x}^k_{\underline{i}^k}\|_2^2|\mathcal{F}^{k-1})(\omega) = \sum_{(i,\mathbf{d})\in\mathcal{N}\times\mathcal{D}} p\left((i,\mathbf{d})|\boldsymbol{\omega}^{0:k-1}\right)\|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^k(\mathbf{d})) - \mathbf{x}^k_i\|_2^2 \tag{32}$$

(recall that we are using the shorthand notation:
$\mathbf{x}^k = \underline{\mathbf{x}}^k(\omega)$ and $\tilde{\mathbf{x}}^k(\mathbf{d}) = (\underline{\tilde{\mathbf{x}}}^k(\mathbf{d}))(\omega)$), it follows from (31) that

$$\liminf_{k\to+\infty} \sum_{(i,\mathbf{d})\in\mathcal{N}\times\mathcal{D}} p\left((i,\mathbf{d})|\omega^{0:k-1}\right) \|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^k(\mathbf{d})) - \mathbf{x}_i^k\|_2^2 = 0. \tag{33}$$

Under C3, there exists a $\delta(\omega) > 0$ such that [cf. (19)]

$$\begin{aligned} &\sum_{(i,\mathbf{d})\in\mathcal{N}\times\mathcal{D}} p\left((i,\mathbf{d})|\omega^{0:k-1}\right) \|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^k(\mathbf{d})) - \mathbf{x}_i^k\|_2^2 \qquad (34)\\ &\geq\ \delta(\omega) \sum_{(i,\mathbf{d})\in\mathcal{V}^k(\omega)} \|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^k(\mathbf{d})) - \mathbf{x}_i^k\|_2^2 \ \geq\ \delta(\omega) \sum_{(i,\mathbf{d})\in\tilde{\mathcal{V}}^k(\omega)} \|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^k(\mathbf{d})) - \mathbf{x}_i^k\|_2^2. \end{aligned}$$

Using (33) and (34), we obtain $\liminf_{k\to+\infty} \sum_{(i,\mathbf{d})\in\tilde{\mathcal{V}}^k(\omega)} \|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^k(\mathbf{d})) - \mathbf{x}_i^k\|_2 = 0$. We prove next that $\lim_{k\to+\infty} \sum_{(i,\mathbf{d})\in\tilde{\mathcal{V}}^k(\omega)} \|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^k(\mathbf{d})) - \mathbf{x}_i^k\|_2 = 0$. To do so, it is sufficient to show that the limsup of the same quantity goes to zero. Let us set for notational simplicity $z^k \triangleq \sum_{(i,\mathbf{d})\in\tilde{\mathcal{V}}^k(\omega)} \|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^k(\mathbf{d})) - \mathbf{x}_i^k\|_2$. We show next that $\lim_{k\to+\infty} z^k = 0$. Suppose by contradiction that $\limsup_{k\to+\infty} z^k > 0$. Since it is also $\liminf_{k\to+\infty} z^k = 0$, there exists a $\upsilon > 0$ such that $z^k > 2\upsilon$ for infinitely many k and also $z^k < \upsilon$ for infinitely many k. Therefore, one can always find an infinite set of indexes, say $\mathcal{S}$, having the following properties: for any $k \in \mathcal{S}$, there exists an integer $j_k > k$ such that

$$z^k < \upsilon, \qquad z^{j_k} > 2\upsilon \tag{35}$$

$$\upsilon \leq z^j \leq 2\upsilon \qquad k < j < j_k. \tag{36}$$

Given the above bounds, the following holds: for all $k \in \mathcal{S}$,

$$\begin{aligned} \upsilon < z^{j_k} - z^k &= \sum_{(i,\mathbf{d})\in\tilde{\mathcal{V}}^{j_k}(\omega)} \|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^{j_k}(\mathbf{d})) - \mathbf{x}_i^{j_k}\|_2 - \sum_{(i,\mathbf{d})\in\tilde{\mathcal{V}}^k(\omega)} \|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^k(\mathbf{d})) - \mathbf{x}_i^k\|_2\\ &\overset{(a)}{\leq} \sum_{i=1}^N \|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^{j_k}(\mathbf{d}_i')) - \mathbf{x}_i^{j_k} - \hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^k(\mathbf{d}_i)) + \mathbf{x}_i^k\|_2\\ &\leq \sum_{i=1}^N \|\mathbf{x}_i^{j_k} - \mathbf{x}_i^k\|_2 + \sum_{i=1}^N \|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^{j_k}(\mathbf{d}_i')) - \hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^k(\mathbf{d}_i))\|_2\\ &\overset{(b)}{\leq} N\|\mathbf{x}^{j_k} - \mathbf{x}^k\|_2 + L_{\hat{\mathbf{x}}} \sum_{i=1}^N \|\tilde{\mathbf{x}}^{j_k}(\mathbf{d}_i') - \tilde{\mathbf{x}}^k(\mathbf{d}_i)\|_2 \overset{(c)}{=} N \sum_{t=k}^{j_k-1} \gamma^t \|\hat{\mathbf{x}}_{i^t}(\tilde{\mathbf{x}}^t(\mathbf{d}^t)) - \mathbf{x}_{i^t}^t\|_2\\ &+ L_{\hat{\mathbf{x}}} \sum_{i=1}^N \|\mathbf{x}^{j_k} + \sum_{l\in K^{j_k}(\mathbf{d}_i')} (\mathbf{x}^l - \mathbf{x}^{l+1}) - \mathbf{x}^k + \sum_{h\in K^k(\mathbf{d}_i)} (\mathbf{x}^{h+1} - \mathbf{x}^h)\|_2 \end{aligned}$$

$$
\begin{aligned}
&\leq N \sum_{t=k}^{j_k-1} \gamma^t \|\hat{\mathbf{x}}_{i^t}(\tilde{\mathbf{x}}^t(\mathbf{d}^t)) - \mathbf{x}_{i^t}^t\|_2 + L_{\hat{\mathbf{x}}} N \|\mathbf{x}^{j_k} - \mathbf{x}^k\|_2 \\
&+ L_{\hat{\mathbf{x}}} \sum_{i=1}^{N} \left(\sum_{l \in K^{j_k}(\mathbf{d}'_i)} \|\mathbf{x}^{l+1} - \mathbf{x}^l\|_2 + \sum_{h \in K^k(\mathbf{d}_i)} \|\mathbf{x}^{h+1} - \mathbf{x}^h\|_2 \right) \\
&\leq (1 + L_{\hat{\mathbf{x}}}) N \sum_{t=k}^{j_k-1} \gamma^t \|\hat{\mathbf{x}}_{i^t}(\tilde{\mathbf{x}}^t(\mathbf{d}^t)) - \mathbf{x}_{i^t}^t\|_2 \\
&+ L_{\hat{\mathbf{x}}} N \left(\sum_{l=j_k-\delta}^{j_k-1} \|\mathbf{x}^{l+1} - \mathbf{x}^l\|_2 + \sum_{h=k-\delta}^{k-1} \|\mathbf{x}^{h+1} - \mathbf{x}^h\|_2 \right) \overset{\text{(d)}}{\leq} 2(1 + L_{\hat{\mathbf{x}}}) N \upsilon \sum_{t=k}^{j_k-1} \gamma^t \\
&+ L_{\hat{\mathbf{x}}} N \left(\sum_{l=j_k-\delta}^{j_k-1} \gamma^l \|\hat{\mathbf{x}}_{i^l}(\tilde{\mathbf{x}}^l(\mathbf{d}^l)) - \mathbf{x}_{i^l}^l\|_2 + \sum_{h=k-\delta}^{k-1} \gamma^h \|\hat{\mathbf{x}}_{i^h}(\tilde{\mathbf{x}}^h(\mathbf{d}^h)) - \mathbf{x}_{i^h}^h\|_2 \right) \\
&\overset{\text{(e)}}{\leq} 2(1 + L_{\hat{\mathbf{x}}}) N \upsilon \sum_{t=k-\delta}^{j_k-1} \gamma^t + 2 L_{\hat{\mathbf{x}}} N \bar{M} \sum_{t=k-\delta}^{j_k-1} \gamma^t \\
&\leq 2N(\upsilon(1 + L_{\hat{\mathbf{x}}}) + L_{\hat{\mathbf{x}}} \bar{M}) \sum_{t=k-\delta}^{j_k-1} \gamma^t,
\end{aligned} \tag{37}
$$

where in (a) we used the definition of the sets $\tilde{\mathcal{V}}^{j_k}(\omega)$ and $\tilde{\mathcal{V}}^k(\omega)$ and we denoted by $\mathbf{d}'_i$ and $\mathbf{d}_i$ the d-component of the pairs $(i, \mathbf{d}') \in \tilde{\mathcal{V}}^{j_k}(\omega)$ and $(i, \mathbf{d}) \in \tilde{\mathcal{V}}^k(\omega)$, respectively; in (b) we used Proposition 2; (c) follows from (20) and Step 4 of Algorithm 1; (d) is due to (35) and (36); and in (e) we used the the boundedness of the iterates implying $\|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{i^k}^k\|_2 < \bar{M}$, for some finite constant $\bar{M} > 0$ and any k. It follows from (37) that

$$
\liminf_{k \to +\infty} \sum_{t=k-\delta}^{j_k-1} \gamma^t \geq \frac{\upsilon}{2N(\upsilon(1 + L_{\hat{\mathbf{x}}}) + L_{\hat{\mathbf{x}}} \bar{M})} > 0. \tag{38}
$$

Consider now the following term: for $k \in \mathcal{S}$, let

$$
\begin{aligned}
&z^{k+1} - z^k \;\leq\; N \|\mathbf{x}^{k+1} - \mathbf{x}^k\|_2 \\
&+ L_{\hat{x}} N \left(\|\mathbf{x}^{k+1} - \mathbf{x}^k\|_2 + \sum_{l=k-\delta+1}^{k} \|\mathbf{x}^{l+1} - \mathbf{x}^l\|_2 + \sum_{h=k-\delta}^{k-1} \|\mathbf{x}^{h+1} - \mathbf{x}^h\|_2 \right) \\
&\leq N(1 + 2L_{\hat{\mathbf{x}}}) \gamma^k \|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k(\mathbf{d}^k)) - \mathbf{x}_{i^k}^k\|_2 + 2\bar{M} L_{\hat{\mathbf{x}}} N \sum_{t=k-\delta}^{k-1} \gamma^t \\
&\overset{\text{(a)}}{\leq} \gamma^k N(1 + 2L_{\hat{\mathbf{x}}}) \left(\|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k(\mathbf{d}^k)) - \mathbf{x}_{i^k}^k\|_2 + \tfrac{2\bar{M} L_{\hat{\mathbf{x}}}}{1 + 2L_{\hat{\mathbf{x}}}} \sum_{t=k-\delta}^{k-1} \left(\tfrac{1}{\eta} \right)^{k-t} \right) \\
&\overset{\text{(b)}}{=} \gamma^k N(1 + 2L_{\hat{\mathbf{x}}}) \left(\|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k(\mathbf{d}^k)) - \mathbf{x}_{i^k}^k\|_2 + \tfrac{2\bar{M} L_{\hat{\mathbf{x}}}}{1 + 2L_{\hat{\mathbf{x}}}} \theta \right) \\
&\leq \gamma^k N(1 + 2L_{\hat{\mathbf{x}}}) \left(z^k + \tfrac{2\bar{M} L_{\hat{\mathbf{x}}}}{1 + 2L_{\hat{\mathbf{x}}}} \theta \right);
\end{aligned} \tag{39}
$$

where in (a) we used D2 and in (b) we defined $\theta \triangleq \sum_{t=k-\delta}^{k-1} \left(\frac{1}{\eta}\right)^{k-t} = \sum_{l=1}^{\delta} \left(\frac{1}{\eta}\right)^{l}$. It turns out that for sufficiently large $k \in \mathcal{S}$ so that $\gamma^k N(1+2L_{\hat{\mathbf{x}}}) < \frac{\upsilon}{\upsilon+2\frac{2ML_{\hat{\mathbf{x}}}}{1+2L_{\hat{\mathbf{x}}}}\theta}$, it must be $z^k \geq \upsilon/2$; otherwise the condition $z^{k+1} \geq \upsilon$ would be violated [cf.(36)]. In the same way, using $z^k \geq \upsilon/2$, one can now show that it must be $z^{k-1} \geq \upsilon/4$, otherwise $z^k \geq \upsilon/2$ would be violated. By applying iteratively this same reasoning one can show that for each $j \in [k-\delta, k]$: $z^j \geq \upsilon/2^{k+1-j}$. For the ω we are considering, (30) holds true; therefore, $\lim_{k\to+\infty} \sum_{t=k-\delta}^{j_k-1} \gamma^t z^t = 0$, that together with (36) and the relation $z^j \geq \upsilon/2^{k+1-j}$. just proved, implies $\lim_{k\to+\infty} \sum_{t=k-\delta}^{j_k-1} \gamma^t = 0$. But this equality contradicts (38). Hence $\limsup_{k\to+\infty} z^k = 0$ and consequently $\lim_{k\to+\infty} z^k = 0$.

Using C2 and $\lim_{k\to+\infty} z^k = 0$, let us prove now that $\lim_{k\to+\infty} \|\hat{\mathbf{x}}(\mathbf{x}^k) - \mathbf{x}^k\|_2 = 0$. To begin, notice that $\lim_{k\to+\infty} z^k = 0$ implies $\lim_{k\to+\infty} \sum_{l=k-\delta}^{k} z^l = 0$. Let us now analyze the following term

$$
\begin{aligned}
\|\hat{\mathbf{x}}(\mathbf{x}^k) - \mathbf{x}^k\|_2 \;&\leq\; \sum_{i=1}^{N} \|\hat{\mathbf{x}}_i(\mathbf{x}^k) - \mathbf{x}_i^k\|_2 \\
&= \sum_{(i,\mathbf{d})\in\tilde{\mathcal{V}}^k(\omega)} \|\hat{\mathbf{x}}_i(\mathbf{x}^k) + \hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^k(\mathbf{d})) - \hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^k(\mathbf{d})) - \mathbf{x}_i^k\|_2 \\
&\leq \sum_{(i,\mathbf{d})\in\tilde{\mathcal{V}}^k(\omega)} \|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^k(\mathbf{d})) - \mathbf{x}_i^k\|_2 + L_{\hat{\mathbf{x}}} \sum_{i=1}^{N} \|\mathbf{x}^k - \tilde{\mathbf{x}}^k(\mathbf{d}_i)\|_2 \qquad (40) \\
&\overset{(a)}{\leq} \sum_{(i,\mathbf{d})\in\tilde{\mathcal{V}}^k(\omega)} \|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^k(\mathbf{d})) - \mathbf{x}_i^k\|_2 + NL_{\hat{\mathbf{x}}} \sum_{l=k-\delta}^{k-1} \|\hat{\mathbf{x}}_{i^l}(\tilde{\mathbf{x}}^l(\mathbf{d}^l)) - \mathbf{x}_{i^l}^l\|_2 \\
&\leq z^k + NL_{\hat{\mathbf{x}}} \sum_{l=k-\delta}^{k-1} z^l \;\leq\; \beta_2 \sum_{l=k-\delta}^{k} z^l,
\end{aligned}
$$

for some $\beta_2 > 0$, where (a) follows from (20) and Step 4 of Algorithm 1. The bound (40) together with $\lim_{k\to+\infty} \sum_{l=k-\delta}^{k} z^l = 0$. leads to $\lim_{k\to+\infty} \|\hat{\mathbf{x}}(\mathbf{x}^k) - \mathbf{x}^k\|_2 = 0$. Finally, since the sequence $\{\mathbf{x}^k\}$ is bounded, it has a limit point in $\mathcal{X}$, denoted by $\bar{\mathbf{x}}$. By the continuity of $\hat{\mathbf{x}}(\bullet)$ (cf. Proposition 2) and the equality $\lim_{k\to+\infty} \|\hat{\mathbf{x}}(\mathbf{x}^k) - \mathbf{x}^k\|_2 = 0$. we just proved, it must be $\hat{\mathbf{x}}(\bar{\mathbf{x}}) = \bar{\mathbf{x}}$. By Proposition 2, $\bar{\mathbf{x}}$ is also a stationary solution of Problem (1).

The above result holds for any $\omega \in \bar{\Omega}$, which completes the proof.

6.3 Proof of Theorem 2

We introduce first the following intermediate results. In this section, the best-response map $\hat{\mathbf{x}}(\bullet)$ is the one defined in (9).

Lemma 1 *Consider AsyFLEXA-NCC under Assumptions A′-B′, C-E and condition ii) of Corollary 1, then the following properties hold.*
(i) $\mathbf{y} \in \mathcal{K}_i(\mathbf{y}) \subseteq \mathcal{K}_i$ *for all* $\mathbf{y} \in \mathcal{K}_i$*;*
(ii) $\hat{\mathbf{x}}_i(\mathbf{y}) \in \mathcal{K}_i(\mathbf{y}_i) \subseteq \mathcal{K}_i$ *for all* $\mathbf{y} \in \mathcal{K}$*.*
Moreover, the sequence $\{\mathbf{x}^k\}$ *generated by the proposed algorithm is such that:*
(iii) $\mathbf{x}_i^k \in \mathcal{K}_i$
(iv) $\mathbf{x}_i^{k+1} \in \mathcal{K}_i(\mathbf{x}_i^k) \cap \mathcal{K}_i(\mathbf{x}_i^{k+1})$

Proof The proof is similar to the proof for Lemma 6 in [25].
(i) The first implication $\mathbf{y} \in \mathcal{K}_i(\mathbf{y})$ follows from $\tilde{c}_{j_i}(\mathbf{y};\mathbf{y}) = c_{j_i}(\mathbf{y}) \leq 0$, for all $j_i = 1, \ldots, m_N$ (due to E2). For the inclusion $\mathcal{K}_i(\mathbf{y}) \subseteq \mathcal{K}_i$, it suffices to recall that, by E3, we have $c_{j_i}(\mathbf{z}) \leq \tilde{c}_{j_i}(\mathbf{z};\mathbf{y})$ for all $\mathbf{z} \in \mathcal{X}_i$, $\mathbf{y} \in \mathcal{K}_i$, and $j_i = 1, \ldots, m_N$, implying that, if $\mathbf{z} \in \mathcal{K}_i(\mathbf{y})$, then $\mathbf{z} \in \mathcal{K}_i$.
(ii) $\hat{\mathbf{x}}_i(\mathbf{y}) \in \mathcal{K}_i(\mathbf{y}_i)$ since it is the optimal solution of a problem of the form (9) (and thus also feasible).
(iii) In view of (i), (ii) and condition ii) of Corollary 1, it follows by induction and the fact that $\mathbf{x}_i^{k+1}$ is a convex combination of $\mathbf{x}_i^k \in \mathcal{K}_i(\mathbf{x}_i^k)$ and $\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^k) \in \mathcal{K}_i(\mathbf{x}_i^k)$, which is a convex subset of $\mathcal{K}_i$.
(iv) By (iii), $\mathbf{x}_i^{k+1} \in \mathcal{K}_i(\mathbf{x}_i^k)$. Furthermore, we have $\tilde{c}_{j_i}(\mathbf{x}_i^{k+1};\mathbf{x}_i^{k+1}) = c_{j_i}(\mathbf{x}_i^{k+1}) \leq 0$, for all $j_i = 1, \ldots, N$, where the equality follows from E2 and the inequality is due to $\mathbf{x}_i^{k+1} \in \mathcal{K}_i$; thus, $\mathbf{x}_i^{k+1} \in \mathcal{K}_i(\mathbf{x}_i^{k+1})$.

Theorem 4 *Consider AsyFLEXA-NCC and the stochastic process* $\underline{\omega}$*. Given the initial point* $\mathbf{x}^0 \in \mathcal{K}$*, let* $\{\mathbf{x}^k\}$ *be the sequence generated by the algorithm. Suppose that Assumptions A′-B′, C-E hold true and the following holds:*

$$\lim_{k \to +\infty} \|\hat{\mathbf{x}}(\underline{\mathbf{x}}^k) - \underline{\mathbf{x}}^k\|_2 = 0 \quad a.s.. \tag{41}$$

Then, there exists a set $\bar{\Omega}$*, with* $\mathbb{P}(\bar{\Omega}) = 1$*, such that for any* $\omega \in \bar{\Omega}$ *the following holds: every limit point of* $\{\mathbf{x}^k\}$ *is a stationary solution of* (7)*.*

Proof The proof is a simple extension of the proof of Theorem 11 in [25], and thus is omitted.

Proof of Theorem 2. The proof of the theorem is similar to the one of Theorem 1; however, the presence of nonconvex constraints changes many steps. We thus present the proof below.
Proof of (a): It follows directly from Lemma 1.
Proof of (b): For any given realization $\omega \in \Omega$ and $k \geq \delta$, the following holds:

$$F(\mathbf{x}^{k+1}) \;=\; f(\mathbf{x}^{k+1}) + g(\mathbf{x}^{k+1}) \;\overset{(a)}{=}\; f(\mathbf{x}^{k+1}) + \sum_{i \neq i^k} g_i(\mathbf{x}_i^{k+1}) + g_{i^k}(\mathbf{x}_{i^k}^{k+1})$$

$$
\begin{aligned}
&\overset{(b)}{=} f(\mathbf{x}^{k+1}) + \sum_{i \neq i^k} g_i(\mathbf{x}_i^k) + g_{i^k}(\mathbf{x}_{i^k}^{k+1}) \\
&\overset{(c)}{\leq} f(\mathbf{x}^k) + \gamma \nabla_{\mathbf{x}_{i^k}} f(\mathbf{x}^k)^{\mathrm{T}} (\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{i^k}^k) + \frac{\gamma^2 L_f}{2} \|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{i^k}^k\|_2^2 \\
&+ \sum_{i \neq i^k} g_i(\mathbf{x}_i^k) + g_{i^k}(\mathbf{x}_{i^k}^{k+1}) \;=\; f(\mathbf{x}^k) + \gamma \nabla_{\mathbf{x}_{i^k}} f(\tilde{\mathbf{x}}^k)^{\mathrm{T}} (\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{i^k}^k) \\
&+ (\nabla_{\mathbf{x}_{i^k}} f(\mathbf{x}^k) - \nabla_{\mathbf{x}_{i^k}} f(\tilde{\mathbf{x}}^k))^{\mathrm{T}} (\gamma(\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{i^k}^k)) + \frac{\gamma^2 L_f}{2} \|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{i^k}^k\|_2^2 \\
&+ \sum_{i \neq i^k} g_i(\mathbf{x}_i^k) + g_{i^k}(\mathbf{x}_{i^k}^{k+1}) \overset{(d)}{\leq} f(\mathbf{x}^k) + \gamma \nabla_{\mathbf{x}_{i^k}} f(\tilde{\mathbf{x}}^k)^{\mathrm{T}} (\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \tilde{\mathbf{x}}_{i^k}^k) \\
&+ (\nabla_{\mathbf{x}_{i^k}} f(\mathbf{x}^k) - \nabla_{\mathbf{x}_{i^k}} f(\tilde{\mathbf{x}}^k))^{\mathrm{T}} (\gamma(\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \tilde{\mathbf{x}}_{i^k}^k)) + \frac{\gamma^2 L_f}{2} \|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \tilde{\mathbf{x}}_{i^k}^k\|_2^2 \\
&+ \sum_{i \neq i^k} g_i(\mathbf{x}_i^k) + \gamma g_{i^k}(\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k)) + g_{i^k}(\mathbf{x}_{i^k}^k) - \gamma g_{i^k}(\tilde{\mathbf{x}}_{i^k}^k) \\
&\overset{(e)}{\leq} F(\mathbf{x}^k) - \gamma(c_{\tilde{f}} - \frac{\gamma L_f}{2}) \|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \tilde{\mathbf{x}}_{i^k}^k\|_2^2 + L_f \|\mathbf{x}^k - \tilde{\mathbf{x}}^k\|_2 \|\gamma(\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \tilde{\mathbf{x}}_{i^k}^k)\|_2 \\
&\overset{(f)}{\leq} F(\mathbf{x}^k) - \gamma(c_{\tilde{f}} - \gamma L_f) \|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \tilde{\mathbf{x}}_{i^k}^k\|_2^2 + \frac{L_f}{2} \|\mathbf{x}^k - \tilde{\mathbf{x}}^k\|_2^2 \\
&\overset{(g)}{=} F(\mathbf{x}^k) - \gamma(c_{\tilde{f}} - \gamma L_f) \|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{i^k}^k\|_2^2 + \frac{L_f}{2} \|\mathbf{x}^k - \tilde{\mathbf{x}}^k\|_2^2,
\end{aligned}
$$

where in (a) we used the separability of g; (b) follows from the updating rule of the algorithm; in (c) we applied the Descent Lemma [3] on f; (d) comes from the convexity of g_i and condition ii) in Corollary 1; in (e) we used Proposition 3 and Assumption A3; (f) is due to the Young's inequality, with $\alpha = 1$; and (g) comes from condition ii) in Corollary 1.

We bound $\|\mathbf{x}^k - \tilde{\mathbf{x}}^k\|_2^2$ as follows:

$$
\begin{aligned}
\|\mathbf{x}^k - \tilde{\mathbf{x}}^k\|_2^2 &\overset{(a)}{\leq} \left(\sum_{l=k-\delta}^{k-1} \|\mathbf{x}^{l+1} - \mathbf{x}^l\|_2 \right)^2 \overset{(b)}{\leq} \delta \sum_{l=k-\delta}^{k-1} \|\mathbf{x}^{l+1} - \mathbf{x}^l\|_2^2 \\
&= \delta \left(\sum_{l=k-\delta}^{k-1} (l - (k-1) + \delta) \|\mathbf{x}^{l+1} - \mathbf{x}^l\|_2^2 - \sum_{l=k+1-\delta}^{k} (l - k + \delta) \|\mathbf{x}^{l+1} - \mathbf{x}^l\|_2^2 \right) \\
&+ \delta^2 \|\mathbf{x}^{k+1} - \mathbf{x}^k\|_2^2 \qquad (42) \\
&= \delta \left(\sum_{l=k-\delta}^{k-1} (l - (k-1) + \delta) \|\mathbf{x}^{l+1} - \mathbf{x}^l\|_2^2 - \sum_{l=k+1-\delta}^{k} (l - k + \delta) \|\mathbf{x}^{l+1} - \mathbf{x}^l\|_2^2 \right) \\
&+ \delta^2 (\gamma^k)^2 \|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{i^k}^k\|_2^2
\end{aligned}
$$

where (a) comes from (20) and (b) is due to the Jensen's inequality.

Let us now define the following Lyapunov function. Let $k \geq \bar{k}$:

$$\tilde{F}(\mathbf{x}^k, \dots, \mathbf{x}^{k-\delta}) = F(\mathbf{x}^k) + \delta \frac{L_f}{2} \left(\sum_{l=k-\delta}^{k-1} (l - (k-1) + \delta) \|\mathbf{x}^{l+1} - \mathbf{x}^l\|_2^2 \right) \quad (43)$$

Using (42) and rearranging the terms, the following holds a.s.: for all $k \geq \bar{k}$

$$\begin{aligned} &\mathbb{E}(\tilde{F}(\underline{\mathbf{x}}^{k+1} \dots, \mathbf{x}^{k+1-\delta}) | \mathcal{F}^{k-1}) \leq \tilde{F}(\underline{\mathbf{x}}^k, \dots, \underline{\mathbf{x}}^{k-\delta}) \\ &-\gamma^k (c_{\bar{f}} - \gamma^k (L_f + \tfrac{\delta^2 L_f}{2})) \mathbb{E}(\|\hat{\mathbf{x}}_{\underline{i}^k}(\tilde{\underline{\mathbf{x}}}^k) - \mathbf{x}_{\underline{i}^k}^k\|_2^2 | \mathcal{F}^{k-1}). \end{aligned} \quad (44)$$

Invoking D1, we have that for sufficiently large k it there exists some positive constant β_1 such that the following holds almost surely:

$$\begin{aligned} &\mathbb{E}(\tilde{F}(\underline{\mathbf{x}}^{k+1} \dots, \mathbf{x}^{k+1-\delta}) | \mathcal{F}^{k-1}) \\ &\leq \tilde{F}(\underline{\mathbf{x}}^k, \dots, \underline{\mathbf{x}}^{k-\delta}) - \beta_1 \gamma^k \mathbb{E}(\|\hat{\mathbf{x}}_{\underline{i}^k}(\tilde{\underline{\mathbf{x}}}^k) - \mathbf{x}_{\underline{i}^k}^k\|_2^2 | \mathcal{F}^{k-1}). \end{aligned} \quad (45)$$

We can now apply Theorem 3 with the following identifications: $\{\underline{\alpha}^k\} = \{\tilde{F}(\underline{\mathbf{x}}^k, \dots, \underline{\mathbf{x}}^{k-\delta})\}$ and $\{\underline{\nu}^k\} = \{\gamma^k \mathbb{E}(\|\hat{\mathbf{x}}_{\underline{i}^k}(\tilde{\underline{\mathbf{x}}}^k) - \mathbf{x}_{\underline{i}^k}^k\|_2^2 | \mathcal{F}^{k-1})\}$ (note that we assumed without loss of generality that $F(\underline{\mathbf{x}}^k) \geq 0$; indeed, since is uniformly bounded from below, one can always find a positive constant c and write instead $F(\underline{\mathbf{x}}^k) + c \geq 0$). We deduce that i) $\{\tilde{F}(\underline{\mathbf{x}}^k, \dots, \underline{\mathbf{x}}^{k-\delta})\}$ converges almost surely; and ii)

$$\lim_{k \to +\infty} \sum_{t=\bar{k}}^{k} \left(\gamma^t \, \mathbb{E}(\|\hat{\mathbf{x}}_{\underline{i}^t}(\tilde{\underline{\mathbf{x}}}^t) - \mathbf{x}_{\underline{i}^t}^t\|_2^2 | \mathcal{F}^{t-1}) \right) < +\infty \quad \text{a. s..} \quad (46)$$

Since $\tilde{F}(\underline{\mathbf{x}}^k, \dots, \underline{\mathbf{x}}^{k-\delta})$ is coercive from condition ii) of Corollary 1, this implies that the sequence $\{\mathbf{x}^k\}$ is bounded almost surely.
Since $\sum_{k=0}^{+\infty} \gamma^k = +\infty$, it follows from (30) that it there exists a $\bar{\Omega} \subseteq \Omega$, with $\mathbb{P}(\bar{\Omega}) = 1$, such that for any $\omega \in \bar{\Omega}$:

$$\liminf_{k \to +\infty} \mathbb{E}(\|\hat{\mathbf{x}}_{\underline{i}^k}(\tilde{\underline{\mathbf{x}}}^k) - \mathbf{x}_{\underline{i}^k}^k\|_2^2 | \mathcal{F}^{k-1})(\omega) = 0. \quad (47)$$

Since we have from Proposition 1:

$$\begin{aligned} \mathbb{E}(\|\hat{\mathbf{x}}_{\underline{i}^k}(\tilde{\underline{\mathbf{x}}}^k) - \mathbf{x}_{\underline{i}^k}^k\|_2^2 | \mathcal{F}^{k-1})(\omega) &= \sum_{(i,\mathbf{d}) \in \mathcal{N} \times \mathcal{D}} \frac{p\left((i,\mathbf{d}), \omega^{0:k-1}\right)}{p(\omega^{0:k-1})} \|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^k(\mathbf{d})) - \mathbf{x}_i^k\|_2^2 \\ &= \sum_{(i,\mathbf{d}) \in \mathcal{N} \times \mathcal{D}} p\left((i,\mathbf{d}) | \omega^{0:k-1}\right) \|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^k(\mathbf{d})) - \mathbf{x}_i^k\|_2^2 \end{aligned} \quad (48)$$

(recall that we are using the shorthand notation: $\mathbf{x}^k = \underline{\mathbf{x}}^k(\omega)$ and $\tilde{\mathbf{x}}^k(\mathbf{d}) = \underline{\tilde{\mathbf{x}}}^k(\mathbf{d})(\omega)$), it follows from (47) that

$$\liminf_{k\to+\infty} \sum_{(i,\mathbf{d})\in\mathcal{N}\times\mathcal{D}} p\left((i,\mathbf{d})|\omega^{0:k-1}\right) \|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^k(\mathbf{d})) - \mathbf{x}_i^k\|_2^2 = 0. \tag{49}$$

Under C3, there exits a $\delta(\omega) > 0$ such that [cf. (19)]

$$\begin{aligned}&\sum_{(i,\mathbf{d})\in\mathcal{N}\times\mathcal{D}} p\left((i,\mathbf{d})|\omega^{0:k-1}\right) \|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^k(\mathbf{d})) - \mathbf{x}_i^k\|_2^2 \\ &\geq \delta(\omega) \sum_{(i,\mathbf{d})\in\mathcal{V}^k(\omega)} \|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^k(\mathbf{d})) - \mathbf{x}_i^k\|_2^2 \;\geq\; \delta(\omega) \sum_{(i,\mathbf{d})\in\tilde{\mathcal{V}}^k(\omega)} \|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^k(\mathbf{d})) - \mathbf{x}_i^k\|_2^2.\end{aligned} \tag{50}$$

Using (49) and (50), we obtain $\liminf_{k\to+\infty} \sum_{(i,\mathbf{d})\in\tilde{\mathcal{V}}^k(\omega)} \|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^k(\mathbf{d})) - \mathbf{x}_i^k\|_2 = 0$. We prove next that actually $\lim_{k\to+\infty} \sum_{(i,\mathbf{d})\in\tilde{\mathcal{V}}^k(\omega)} \|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^k(\mathbf{d})) - \mathbf{x}_i^k\|_2 = 0$. To do so, it is sufficient to show that the limsup of the same quantity goes to zero. Let us set, for notational simplicity, $z^k \triangleq \sum_{(i,\mathbf{d})\in\tilde{\mathcal{V}}^k(\omega)} \|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^k(\mathbf{d})) - \mathbf{x}_i^k\|_2$. We show next that $\lim_{k\to+\infty} z^k = 0$. Suppose by contradiction that $\limsup_{k\to+\infty} z^k > 0$. Since it is also $\liminf_{k\to+\infty} z^k = 0$, there exists a $\upsilon > 0$ such that $z^k > 2\upsilon$ for infinitely many k and also $z^k < \upsilon$ for infinitely many k. Therefore, one can always find an infinite set of indexes, say $\mathcal{S}$, having the following properties: for any $k \in \mathcal{S}$, there exists an integer $j_k > k$ such that

$$z^k < \upsilon, \qquad z^{j_k} > 2\upsilon \tag{51}$$

$$\upsilon \leq z^j \leq 2\upsilon \qquad k < j < j_k. \tag{52}$$

Given the above bounds, the following holds: for all $k \in \mathcal{S}$,

$$\begin{aligned}\upsilon < z^{j_k} - z^k &= \sum_{(i,\mathbf{d})\in\tilde{\mathcal{V}}^{j_k}(\omega)} \|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^{j_k}(\mathbf{d})) - \mathbf{x}_i^{j_k}\|_2 - \sum_{(i,\mathbf{d})\in\tilde{\mathcal{V}}^k(\omega)} \|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^k(\mathbf{d})) - \mathbf{x}_i^k\|_2 \\ &\overset{(a)}{\leq} \sum_{i=1}^N \|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^{j_k}(\mathbf{d}_i')) - \mathbf{x}_i^{j_k} - \hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^k(\mathbf{d}_i)) + \mathbf{x}_i^k\|_2 \\ &\leq \sum_{i=1}^N \|\mathbf{x}_i^{j_k} - \mathbf{x}_i^k\|_2 + \sum_{i=1}^N \|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^{j_k}(\mathbf{d}_i')) - \hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^k(\mathbf{d}_i))\|_2 \\ &\overset{(b)}{\leq} N\|\mathbf{x}^{j_k} - \mathbf{x}^k\|_2 + \tilde{L}\sum_{i=1}^N \|\tilde{\mathbf{x}}^{j_k}(\mathbf{d}_i') - \tilde{\mathbf{x}}^k(\mathbf{d}_i)\|_2^{1/2} \overset{(c)}{=} N\sum_{t=k}^{j_k-1} \gamma^t \|\hat{\mathbf{x}}_{i^t}(\tilde{\mathbf{x}}^t(\mathbf{d}^t)) - \mathbf{x}_{i^t}^t\|_2 \\ &+ \tilde{L}\sum_{i=1}^N \|\mathbf{x}^{j_k} + \sum_{l\in K^{j_k}(\mathbf{d}_i')} (\mathbf{x}^l - \mathbf{x}^{l+1}) - \mathbf{x}^k + \sum_{h\in K^k(\mathbf{d}_i)} (\mathbf{x}^{h+1} - \mathbf{x}^h)\|_2^{1/2}\end{aligned} \tag{53}$$

$$
\begin{aligned}
&\leq N \sum_{t=k}^{j_k-1} \gamma^t \|\hat{\mathbf{x}}_{i^t}(\tilde{\mathbf{x}}^t(\mathbf{d}^t)) - \mathbf{x}_{i^t}^t\|_2 \\
&+ \tilde{L} \sum_{i=1}^{N} \left(\|\mathbf{x}^{j_k} - \mathbf{x}^k\|_2 + \sum_{l \in K^{j_k}(\mathbf{d}_i')} \|\mathbf{x}^{l+1} - \mathbf{x}^l\|_2 + \sum_{h \in K^k(\mathbf{d}_i)} \|\mathbf{x}^{h+1} - \mathbf{x}^h\|_2 \right)^{1/2} \\
&\leq N \sum_{t=k}^{j_k-1} \gamma^t \|\hat{\mathbf{x}}_{i^t}(\tilde{\mathbf{x}}^t(\mathbf{d}^t)) - \mathbf{x}_{i^t}^t\|_2 + \tilde{L}N \Big(\sum_{t=k}^{j_k-1} \gamma^t \|\hat{\mathbf{x}}_{i^t}(\tilde{\mathbf{x}}^t(\mathbf{d}^t)) - \mathbf{x}_{i^t}^t\|_2 \\
&+ \sum_{l=j_k-\delta}^{j_k-1} \|\mathbf{x}^{l+1} - \mathbf{x}^l\|_2 + \sum_{h=k-\delta}^{k-1} \|\mathbf{x}^{h+1} - \mathbf{x}^h\|_2 \Big)^{1/2} \\
&\overset{\text{(d)}}{\leq} 2N\upsilon \sum_{t=k}^{j_k-1} \gamma^t + \tilde{L}N \Big(2\upsilon \sum_{t=k}^{j_k-1} \gamma^t + \sum_{l=j_k-\delta}^{j_k-1} \gamma^l \|\hat{\mathbf{x}}_{i^l}(\tilde{\mathbf{x}}^l(\mathbf{d}^l)) - \mathbf{x}_{i^l}^l\|_2 \\
&+ \sum_{h=k-\delta}^{k-1} \gamma^h \|\hat{\mathbf{x}}_{i^h}(\tilde{\mathbf{x}}^h(\mathbf{d}^h)) - \mathbf{x}_{i^h}^h\|_2 \Big)^{1/2} \overset{\text{(e)}}{\leq} 2N\upsilon \sum_{t=k-\delta}^{j_k-1} \gamma^t + \tilde{L}N \left(2(\upsilon + \bar{M}) \sum_{t=k-\delta}^{j_k-1} \gamma^t \right)^{1/2} \\
&\leq 2N\upsilon \sum_{t=k-\delta}^{j_k-1} \gamma^t + \tilde{L}N \sqrt{2(\upsilon + \bar{M})} \left(\sum_{t=k-\delta}^{j_k-1} \gamma^t \right)^{1/2}
\end{aligned}
$$

where in (a) we used the definition of the sets $\tilde{\mathcal{V}}^{j_k}(\omega)$ and $\tilde{\mathcal{V}}^k(\omega)$ and we denoted by $\mathbf{d}_i'$ and $\mathbf{d}_i$ the d-component of the pairs $(i, \mathbf{d}') \in \tilde{\mathcal{V}}^{j_k}(\omega)$ and $(i, \mathbf{d}) \in \tilde{\mathcal{V}}^k(\omega)$, respectively; in (b) we used the fact Proposition 3; (c) follows from (20) and the updating rule of the algorithm; (d) is due to (51) and (52); and in (e) we used the the boundedness of the iterates implying $\|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{i^k}^k\|_2 < \bar{M}$, for some finite constant $\bar{M} > 0$ and any k. It follows from (53) that

$$
\liminf_{k \to +\infty} \left(2N\upsilon \sum_{t=k-\delta}^{j_k-1} \gamma^t + \tilde{L}N \sqrt{2(\upsilon + \bar{M})} \sqrt{\sum_{t=k-\delta}^{j_k-1} \gamma^t} \right) > 0. \tag{54}
$$

Consider now the following term: for $k \in \mathcal{S}$, let

$$
\begin{aligned}
z^{k+1} - z^k \;&\leq\; N\|\mathbf{x}^{k+1} - \mathbf{x}^k\|_2 \\
&+ \tilde{L}N \left(\|\mathbf{x}^{k+1} - \mathbf{x}^k\|_2 + \sum_{l=k-\delta+1}^{k} \|\mathbf{x}^{l+1} - \mathbf{x}^l\|_2 + \sum_{h=k-\delta}^{k-1} \|\mathbf{x}^{h+1} - \mathbf{x}^h\|_2 \right)^{1/2} \\
&\leq N\gamma^k \|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k(\mathbf{d}^k)) - \mathbf{x}_{i^k}^k\|_2 + \sqrt{2}\tilde{L}N\sqrt{\gamma^k} \|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k(\mathbf{d}^k)) - \mathbf{x}_{i^k}^k\|_2^{1/2} \\
&+ \sqrt{2\bar{M}}\tilde{L}N \sum_{t=k-\delta}^{k-1} \sqrt{\gamma^t} \overset{\text{(a)}}{\leq} N\gamma^k \|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k(\mathbf{d}^k)) - \mathbf{x}_{i^k}^k\|_2 \\
&+ \sqrt{2}\tilde{L}N\sqrt{\gamma^k} \left(\|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k(\mathbf{d}^k)) - \mathbf{x}_{i^k}^k\|_2^{1/2} + \sqrt{\bar{M}} \sum_{t=k-\delta}^{k} \left(\frac{1}{\eta} \right)^{\frac{k-t}{2}} \right)
\end{aligned}
$$

$$\begin{aligned}
&\overset{(b)}{=} N\gamma^k \|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k(\mathbf{d}^k)) - \mathbf{x}_{i^k}^k\|_2 \\
&+ \sqrt{2}\tilde{L}N\sqrt{\gamma^k}\left(\|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k(\mathbf{d}^k)) - \mathbf{x}_{i^k}^k\|_2^{1/2} + \sqrt{\bar{M}}\theta\right) \\
&\leq N\gamma^k z^k + \sqrt{2}\tilde{L}N\sqrt{\gamma^k}\left(\sqrt{z^k} + \sqrt{\bar{M}}\theta\right);
\end{aligned}$$

where in (a) we used D2 and in (b) we defined $\theta \triangleq \sum_{t=k-\delta}^{k-1}\left(\frac{1}{\eta}\right)^{\frac{k-t}{2}} = \sum_{l=1}^{\delta}\left(\frac{1}{\eta}\right)^{\frac{l}{2}}$. Now, for sufficiently large $k \in \mathcal{S}$, so that $N\gamma^k\frac{\upsilon}{2} + \sqrt{2}\tilde{L}N\sqrt{\gamma^k}\left(\sqrt{\frac{\upsilon}{2}} + \sqrt{\bar{M}}\theta\right) < \frac{\upsilon}{2}$, suppose by contradiction that $z^k < \frac{\upsilon}{2}$; this would give $z^{k+1} < \frac{\upsilon}{2}$ and condition (52) (or, in case, (51)) would be violated. Then it must be $z^k \geq \frac{\upsilon}{2}$. In the same way one can now show that it must be $z^{k-1} \geq \frac{\upsilon}{4}$, otherwise $z^k \geq \frac{\upsilon}{2}$ would be violated. By applying iteratively this same reasoning, one can show that for each $j \in [k-\delta, k]$: $z^j \geq \upsilon/2^{k+1-j}$. For the ω we are considering, (46) holds; therefore, $\lim_{k\to+\infty}\sum_{t=k-\delta}^{j_k-1}\gamma^t z^t = 0$; which together with (52) and $z^j \geq \upsilon/2^{k+1-j}$.implies $\lim_{k\to+\infty}\sum_{t=k-\delta}^{j_k-1}\gamma^t = 0$. But the last equation contradicts (54). Therefore $\limsup_{k\to+\infty} z^k = 0$ and consequently $\lim_{k\to+\infty} z^k = 0$. Using C2 and $\lim_{k\to+\infty} z^k = 0$, let us prove now that $\lim_{k\to+\infty}\|\hat{\mathbf{x}}(\mathbf{x}^k) - \mathbf{x}^k\|_2 = 0$. To begin, notice that $\lim_{k\to+\infty} z^k = 0$ implies $\lim_{k\to+\infty}\sqrt{z^k} = 0$ and

$$\lim_{k\to+\infty}\sum_{l=k-\delta}^{k-1}\sqrt{z^l} = 0, \tag{55}$$

for any $k \geq 0$. We also have

$$\begin{aligned}
\|\hat{\mathbf{x}}(\mathbf{x}^k) - \mathbf{x}^k\|_2 \;&\leq\; \sum_{i=1}^{N}\|\hat{\mathbf{x}}_i(\mathbf{x}^k) - \mathbf{x}_i^k\|_2 \\
&= \sum_{(i,\mathbf{d})\in\tilde{\mathcal{V}}^k(\omega)}\|\hat{\mathbf{x}}_i(\mathbf{x}^k) + \hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^k(\mathbf{d})) - \hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^k(\mathbf{d})) - \mathbf{x}_i^k\|_2 \\
&\overset{(a)}{\leq} \sum_{(i,\mathbf{d})\in\tilde{\mathcal{V}}^k(\omega)}\|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^k(\mathbf{d})) - \mathbf{x}_i^k\|_2 + \tilde{L}\sum_{i=1}^{N}\|\mathbf{x}^k - \tilde{\mathbf{x}}^k(\mathbf{d}_i)\|_2^{1/2} \qquad (56) \\
&\overset{(b)}{\leq} \sum_{(i,\mathbf{d})\in\tilde{\mathcal{V}}^k(\omega)}\|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^k(\mathbf{d})) - \mathbf{x}_i^k\|_2 + N\tilde{L}\sum_{l=k-\delta}^{k-1}\|\hat{\mathbf{x}}_{i^l}(\tilde{\mathbf{x}}^l(\mathbf{d}^l)) - \mathbf{x}_{i^l}^l\|_2^{1/2} \\
&\leq z^k + N\tilde{L}\sum_{l=k-\delta}^{k-1}\sqrt{z^l},
\end{aligned}$$

where (a) follows from Proposition 3; and (b) follows from (20) and the updating rule of the algorithm. Bound (56) together with $\lim_{k\to+\infty} z^k = 0$. and (55) leads to $\lim_{k\to+\infty}\|\hat{\mathbf{x}}(\mathbf{x}^k) - \mathbf{x}^k\|_2 = 0$. Since this result holds for any $\omega \in \bar{\Omega}$, it is now sufficient to invoke Theorem 4 in order to complete the proof.