More Iterations per Second, Same Quality – Why Asynchronous Algorithms may Drastically Outperform Traditional Ones

Robert Hannah* and Wotao Yin†

Department of Mathematics, University of California, Los Angeles, CA 90095, USA

August 18, 2017

Abstract

In this paper, we consider the convergence of a very general asynchronous-parallel algorithm called ARock [1], that takes many well-known asynchronous algorithms as special cases (gradient descent, proximal gradient, Douglas Rachford, ADMM, etc.). In asynchronous-parallel algorithms, the computing nodes simply use the most recent information that they have access to, instead of waiting for a full update from all nodes in the system. This means that nodes do not have to waste time waiting for information, which can be a major bottleneck, especially in distributed systems. When the system has p nodes, asynchronous algorithms may complete $\Theta(\ln(p))$ more iterations than synchronous algorithms in a given time period ("more iterations per second").

Although asynchronous algorithms may compute more iterations per second, there is error associated with using outdated information. How many more iterations in total are needed to compensate for this error is still an open question. The main results of this paper aim to answer this question. We prove, loosely, that as the size of the problem becomes large, the number of additional iterations that asynchronous algorithms need becomes negligible compared to the total number ("same quality" of the iterations). Taking these facts together, our results provide solid evidence of the potential of asynchronous algorithms to vastly speed up certain distributed computations.

1 Introduction

Designing efficient algorithms to solve large-scale optimization problem is an increasingly important area of research. However parallel algorithms are more challenging to analyze and implement because there is a host of additional considerations and issues that only arise in parallel settings.

The vast majority of parallel algorithms are synchronous. At each iteration, all processors will compute an update, and then share this update with all others. The next iteration can proceed only when all processors are finished computing an update. This synchronization can be extremely expensive at scale or on a congested network. Network latency, packet loss, loss of a node, unexpected drains on computational resources that affect even one node will cause the entire system to slow down. Asynchronous algorithms overcome the

*Email: RobertHannah89@math.ucla.edu
†Email: wotaoyin@math.ucla.edu

problem of synchronization by simply computing their next update with the most recent information they have available. Though this will eliminate synchronization penalty, there is a slowdown or penalty associated with using outdated information. It is not immediately clear whether asynchronous versions of algorithms will be faster, or will even converge to a solution.

In this paper, we examine the theoretical performance of asynchronous algorithms compared to traditional synchronous ones. We do this by analyzing a very general asynchronous-parallel algorithm called ARock [1]. ARock takes many popular algorithms as special cases, such as asynchronous block gradient descent, forward backward, proximal point, etc. Hence our results will apply to all of these algorithms.

1.1 Main argument

This paper aims to provide a solid theoretical evidence that asynchronous algorithms will drastically outperform synchronous ones at scale under a wide range of scenarios that ARock encompasses. These include gradient descent, forward backward, etc. for strongly convex objectives with Lipschitz gradient, or any other algorithm that can be written in the form of a block fixed-point algorithm on a contractive operator. Our argument involves a series of steps, that together will bolster our conclusion.

1. We first argue that synchronous algorithms have significant synchronization penalty with scale under a well-justified model. That is, as the number of processing nodes p increases, a larger and larger portion time will be spend waiting instead of computing. In fact, it will be shown that synchronous algorithms progress at least $\Theta(\ln(p))$ fewer epochs[1] per second than asynchronous algorithms.

2. It has always been plausible that asynchronous algorithms may progress more epochs per second. However, it has always been an open question whether an increased number of iterations per second is worth a potential iteration complexity penalty for using outdated information. In this paper, we provide a surprising answer: the iteration complexity of ARock is asymptotically the same as the corresponding synchronous algorithm, even under certain kinds of unbounded delays.

3. Since asynchronous algorithms allow for far more iterations per second (more iterations), and these iterations make the nearly same progress as synchronous ones *per iteration* (same quality), asynchronous algorithms may drastically outperform synchronous ones in large-scale applications. Since ARock is extremely general, this argument applies to wide variety of asynchronous algorithms, such as block gradient descent, proximal gradient, Douglas-Rachford, etc.

The remainder of this section will introduce the general setting, notation, and background for our results. We will discuss related work in Section 2.5.

1.2 Fixed-point algorithms, and their generality

ARock is a fixed-point algorithm, which allows it to be very general. This is because most optimization problems and algorithms can be written in the fixed-point form. Many popular algorithms such as asynchronous block gradient descent, forward backward, proximal point, etc. are special cases of ARock. They differ only in their choice of fixed-point operator T (see [2] for a list of applications and special cases). Hence our results will apply to all of these algorithms.

[1] We measure the iteration complexity in terms of **epochs**. This is a context-dependent unit of computation which loosely correspond to one computation of Sx for some operator S and vector x, e.g., the calculation of a full gradient for gradient descent.

Take an operator $T : \mathbb{H} \to \mathbb{H}$ with Lipschitz constant $0 < r \leq 1$. Such an operator is called nonexpansive. The aim is to find a **fixed-point** of this operator: That is, a point $x^* \in \mathbb{H}$ such that $Tx^* = x^*$. For example, smooth minimization of a convex function $f : \mathbb{H} \to \mathbb{R}$ with L-Lipschitz gradient ∇f is equivalent to finding a fixed point of the nonexpansive operator $T = I - \gamma \nabla f$, where I is the identity, and $0 < \gamma \leq \frac{2}{L}$. The set of fixed points of an operator T is denoted $\mathrm{Fix}(T)$. In this paper, we consider the case where $0 < r < 1$, that is, T is contractive, since this leads to linear convergence[2].

The most common fixed-point algorithm is the Krasnosel'skiĭ-Mann (KM) algorithm. ARock is essentially an asynchronous block-coordinate version of KM iteration.

Definition 1. Krasnosel'skiĭ-Mann algorithm. Let $\epsilon > 0$, and η^k be a series of step lengths in $(\epsilon, 1 - \epsilon)$. Let T be a nonexpansive operator with at least one fixed point, and

$$S := I - T. \tag{1.1}$$

Staring from x^0, the KM Algorithm is defined by the following:

$$x^{k+1} = x^k - \eta^k S(x^k) \tag{1.2}$$

Remark 1. Gradient descent is equivalent to the KM algorithm with $T = I - \gamma \nabla f$. If the fixed-point framework is unfamiliar, it may be helpful to mentally replace S with $\gamma \nabla f$, and view ARock as simply asynchronous block gradient descent.

1.3 The ARock algorithm

ARock is an asynchronous-parallel, block, fixed-point algorithm, in which a shared solution vector x^k is updated by a collection of p computing nodes.

Take a space $\mathbb{H}$ on which to solve an optimization problem. $\mathbb{H}$ can be the real space $\mathbb{R}^N$ or a separable Hilbert space. Break this space into m orthogonal subspaces: $\mathbb{H} = \mathbb{H}_1 \times \ldots \times \mathbb{H}_m$ so that vectors $x \in \mathbb{H}$ can be written as $(x_1, x_2, \ldots, x_m)$ where each x_i is x's component in subspace $\mathbb{H}_i$. Take an operator $T : \mathbb{H} \to \mathbb{H}$ that is r-Lipschitz for $0 < r < 1$. Let $S = I - T$ and $Sx = (S_1 x, \ldots, S_m x)$ where $S_j x$ denotes the j'th block of Sx.

Remark 2. Conventions. *Superscripts* will denote the iteration number of a sequence of points $x^0, x^1, x^2, \ldots$. *Subscripts* will denote different blocks of a vector or operator, e.g., $x = (x_1, x_2, \ldots, x_m)$ and $Sx = (S_1 x, \ldots, S_m x)$. For instance, x_l^k is the lth block of the kth iterate (x^k). $S_l x^k$ is the lth block of $S(x^k)$.

Definition 2. The ARock Algorithm. Let $\eta^k \in \mathbb{R}$ be a series of step lengths and $i(k) \in \{1, \ldots, m\}$ be a series of block indices. Let T be a nonexpansive operator with at least one fixed point x^*, and $S = I - T$. Take a starting point $x^0 \in \mathbb{H}$. Then the ARock algorithm [1] is defined via the iteration:

$$\text{for } i = 1, \ldots, m, \quad x_i^{k+1} \leftarrow \begin{cases} x_i^k - \eta^k S_i(\hat{x}^k), & i = i(k), \\ x_i^k, & i \neq i(k), \end{cases} \tag{1.3}$$

where the **delayed iterate** $\hat{x}^k$ represents a possibly outdated version of the iteration vector x^k used to make an update and the **block index sequence** $i(k)$ specifies which block of x^k is being updated to produce the next iterate x^{k+1}.

The ARock algorithm resembles a block KM iteration. However we use a delayed iterate $\hat{x}^k$ because of asynchronicity. In Section 1.4 we precisely define the block sequence $i(k)$, and the delayed iterate $\hat{x}^k$.

[2] The case of $r = 1$ was considered in [2].

1.4 Setup

We work with a probability measure space: $(\Omega, \Sigma, \mathbb{P})$[3]. We now describe the delayed iterates (which is part of our model of asynchronicity) and the block index.

1.4.1 Delayed iterates

Let $\vec{j} = (j_1, \ldots, j_m) \in \mathbb{N}^m$ be a vector, and $x^0, x^1, x^2, \ldots$ a series of iterates. To model outdated solution data, we find it convenient to define:

$$x^{k-\vec{j}} = \left(x_1^{k-j_1}, x_2^{k-j_2}, \ldots, x_m^{k-j_m}\right). \tag{1.4}$$

Hence we define a series of **delay vectors** $\vec{j}(0), \vec{j}(1), \vec{j}(2), \ldots$ in $\mathbb{N}^m$, corresponding to $x^0, x^1, x^2, \ldots$ respectively. The components of the delay vector $\vec{j}(k) = (j(k,1), j(k,2), \ldots, j(k,m))$ represent the staleness of the components of the solution vector x^k.

Definition 3. Delayed iterate. The delayed iterate $\hat{x}^k$ is defined as[4]:

$$\hat{x}^k = x^{k-\vec{j}(k)}, \text{ or equivalently,} \tag{1.5}$$

$$\hat{x}^k = \left(\hat{x}_1^k, \hat{x}_2^k, \ldots, \hat{x}_m^k\right) = \left(x_1^{k-j(k,1)}, x_2^{k-j(k,2)}, \ldots, x_m^{k-j(k,m)}\right). \tag{1.6}$$

Lastly, we define the current delay $j(k)$ as follows[5]:

$$j(k) = \max_i \{j(k,i)\}. \tag{1.7}$$

These delay vectors depend on the model of asynchronicity chosen. We consider two possibilities in this paper: stochastic and deterministic delays.

1.4.2 Block sequence

We define following filtrations to represent the information that is accumulated as the algorithm runs.

$$\mathcal{F}^k = \sigma\left(x^0, x^1, \ldots, x^k, \vec{j}(0), \vec{j}(1), \ldots, \vec{j}(k)\right) \tag{1.8}$$

$$\mathcal{G}^k = \sigma\left(x^0, x^1, \ldots, x^k\right) \tag{1.9}$$

Here $\sigma(a, b, c, \ldots)$ represents the sigma algebra generated by $a, b, c, \ldots$.

Assumption 1. IID block sequence. The sequence in which blocks of the solution vector are updated, $i(k)$, is a series of uniform[6] IID random variables that takes values $1, 2, \ldots, m$ each with probability $1/m$. $i(k)$ is independent of $\mathcal{F}^k$. That is, $i(k)$ is independent of the sequence of iterates $(x^0, x^1, \ldots, x^k)$ and the sequence of delays $(\vec{j}(0), \vec{j}(1), \ldots, \vec{j}(k))$ jointly[7].

[3] Ω is the probability space, Σ is a sigma algebra, and $\mathbb{P}$ is a corresponding probability measure.

[4] **Stronger asynchronicity:** It is possible to have more general asynchronicity, where different components of the *same block*, $x_l \in \mathbb{H}_l$, have different ages. This leads to similar results, and a similar proof, but the current setup was chosen for simplicity.

[5] Notice the lack of a vector symbol: This distinguishes the current delay from the delay vector.

[6] Nonuniform probabilities are a simple extension. However for simplicity we assume a uniform distribution.

[7] Clearly this makes $i(k)$ independent of $\mathcal{G}^k$ as well.

It is very difficult to remove the assumption that the block sequence is independent of the sequence of delays. Only a few papers that we are aware of make progress in eliminating this assumption [3]–[5]. However obtaining good convergence rates remains elusive.

Also removing the assumption of a random block sequence, and assuming, say, a cyclic choice as in [6], [7] leads to at least an m-times slowdown of the algorithm in the worst case for smooth minimization [6]. The block sequence will be IID if we allow all nodes to randomly update any block chosen in a uniform IID fashion, and computing each block is of equal difficulty. Future work may involve finding an intermediate scenarios between IID and cyclic block choices that still results in adequate rates.

1.5 Lyapunov functions

In this framework, it is easy to generate an example where we have conditional expectation bound:

$$\mathbb{E}\Big[\big\|x^{k+1}-x^*\big\|^2|\sigma\Big(x^0,x^1,\ldots,x^k,\vec{j}(0),\vec{j}(1),\ldots,\vec{j}(k)\Big)\Big] > \big\|x^k-x^*\big\|^2 \tag{1.10}$$

for any nonzero step size. However usually some kind of monotonicity is necessary to prove convergence, especially linear convergence. In [2], following on from [1], the authors propose an **asynchronicity error** term to add to the classical error:

$$\underbrace{\xi^k}_{\text{Total error}} = \underbrace{\big\|x^k-x^*\big\|^2}_{\text{Classical error}} + \underbrace{\frac{1}{m}\sum_{i=1}^{\infty} c_i\big\|x^{k+1-i}-x^{k-i}\big\|^2}_{\text{Asynchronicity error}} \tag{1.11}$$

for positive decreasing coefficients $(c_1, c_2, \ldots)$. Using carefully chosen coefficients and step size, they are able to prove convergence of ARock under unbounded delay for T with Lipschitz constant $r = 1$. This Lyapunov function appears naturally in the proof, and much like a well chosen basis in linear algebra, it seems to be the most natural error to consider when proving convergence. Refer to [2] for further motivation, and the general strategy for generating useful Lyapunov functions.

We use the same Lyapunov function in this paper to derive the main results. However our choice of coefficients is drastically different. Choosing the coefficients that yield strong results is a very involved process and is part of the technical innovation of this paper.

1.6 Structure of the paper

In Section 2, we systematically work through the points of the main argument in Section 1.1, which bolsters our main thesis that asynchronous algorithms are drastically faster at scale, and discuss related work in Section 2.5. This culminates in our most important contribution: Theorem 1, which essentially proves that asynchronous ARock has the same iteration complexity as its synchronous counterpart. This result is proven in Section 3, and Section 4 introduces and proves a similar result for deterministic unbounded delays.

2 New results

In this section, we present our main results, that justify the proposition in our main argument in Section 1.1. First we describe the implementation setup in Section 2.1, that is, the kind of context where our main argument applies. Next in Section 2.2, we describe some factors that cause synchronous algorithms to perform

fewer epoch in a given time period. For instance, we prove that as the number of nodes increases, under a reasonable model synchronous algorithms suffer a $\Theta(\ln(p))$ slowdown, whereas asynchronous algorithms suffer no such penalty (more iterations). In Section 2.3, we derive a sharp convergence rate for synchronous-parallel ARock, so that we can compare iteration complexities. This appears to be a new result, with many implications in parallel optimization. In Section 2.4, we present our main results: The iteration complexity of (asynchronous) ARock is essentially the same as its synchronous counterpart. This result is our main theoretical contribution. The other subsections are independent contributions whose role is to justify our argument and bolster the important of the main theorems. Finally in Section 2.5, we discuss related work and compare results.

2.1 Implementation setup

We now describe the implementation setup that we will be considering. The results and analysis may be more general than this setup, but being concrete about the setup allows us to compare synchronous vs. asynchronous implementations of the same algorithm. Our implementation setup will consist of:

1. **Central Parameter Server:** This is a central node that will maintain the solution vector x, and apply updates that are supplied to it[8]: $x \leftarrow x - \eta S_i \hat{x}$.
2. **Computing Nodes:** There are p nodes that read the solution vector $\hat{x}$ into local memory, calculate $S_i\hat{x}$, and send this update back to the central server.

We now compare and contrast how a synchronous and asynchronous version of ARock would function:

- **Synchronous-parallel:** All computing nodes read the same solution vector x from the server. All computing nodes will compute an update $S_i x$, where i depends on the node, and send this update to the server. The server will save all received updates and, only after x has been read by every node, apply the received updates to the solution vector. After this, all computing nodes will then read the same updated solution vector again, and the process repeats.
- **Asynchronous-parallel:** All computing nodes will continually read the central solution vector x into their local memory. This yields $\hat{x}$, a potentially inconsistently read solution vector. Each node will then calculate an update $S_i\hat{x}$ and send it back to the server. The server will apply any updates to the solution vector as they arrive.

Notice that in the synchronous implementation, a single slow node can hold the entire system up. In the asynchronous implementation, all nodes function independently, and never wait for any other node. The central server does not wait to receive an update from every node, but simply applies them as they come. We now analyze the synchronization penalty of these two implementation setups.

2.2 Synchronization penalty

In this section we discuess point 2 of the main argument in Section 1.1. We consider a simple and well-justified model of our implementation setup to investigate synchronization penalty. Even under perfect load balancing, this model will imply a slowdown for synchronous algorithms that increases as $\Theta(\ln(p))$, where

[8] We could have considered a shared memory architecture, however we chose this setting since we are more focused on algorithm performance at scale.

p is the number of processors. This may be even worse in a real implementation, where other factors may further disadvantage synchronous algorithms (such as the overhead associated with message passing and read/write locks). For the corresponding asynchronous algorithm, we show these problems do not occur. Hence asynchronous algorithms may compute far more iterations per second.

Following [8] (pp. 43) and adding modifications, we model the time taken for node l's update as follows:

$$P_l = R_l + C_l(i_l, m) + S_l. \tag{2.1}$$

Here i_l is the block of the solution vector that node l updates, and $C_l(i, m)$ represents the "predictable" portion of the update time (it is merely a function, not a random variable). This includes computation time, and the delay because of the limited bandwidth of the network. $C_l(i, m)$ is a function of i because different blocks may have different sizes and difficulties. R_l is the random delay involved in receiving the solution vector from the central server, and S_l is the random delay involved in sending an update back to the server. R_l and S_l are assumed IID with exponential distribution of mean λ_l. This exponential model for the random portion of the delay has extensive theoretical and empirical justifications (see [8], pp. 44-45 for a discussion of the evidence).

2.2.1 Synchronous algorithms and random delays

Let's first consider the effect of random delays on the synchronization penalty. For simplicity, assume that $C_l(i, m)$ is constant over i and l, and hence we can write this function as $C(m)$. Also assume we have $\lambda_l = \lambda$ for all l. This situation would occur if all blocks were of equal difficulty to update, and all nodes had the same computational power and network delay distribution. This is the ideal scenario, and yet we will observe a growing synchronization penalty with scale.

Because all nodes must finish updating for the next iteration to start, the iteration time P is given by:

$$P = C(m) + \max_{l=1,2,\ldots,p} \{R_l + S_l\}. \tag{2.2}$$

Hence we have (using [9]):

$$\mathbb{E}P - C(m) \geq \mathbb{E}\left(\max_{l=1,\ldots,p} \{R_l\}\right) = \lambda \sum_{l=1}^{p} \frac{1}{l} \geq \lambda \ln(p).$$

Let's now look at the time $\mathcal{T}(K)$ required for K epochs, which corresponds to Km/p iterations. Let[9] $P^1, P^2, \ldots \sim P$. Then:

$$\begin{aligned} \mathcal{T}(K) &= \sum_{k=1}^{\lceil Km/p \rceil} P^k, \\ \mathbb{E}\mathcal{T}(K) &\geq (Km/p)\mathbb{E}P \\ &\geq (Km/p)(C(m) + \lambda \ln(p)). \end{aligned}$$

Hence for small values of p, the expected time to reach K epochs will decrease linearly with the number of nodes p. However as p becomes larger, there is at least a $\Theta(\ln(p))$ penalty in how long this will take compared to a linear speedup.

[9] We write $A \sim B$ for random variables A and B if these variables have the same distribution.

2.2.2 Asynchronous algorithms

Using the same model, we now show that asynchronous algorithms have no such $\Theta(\ln(p))$ scaling penalty. The time taken for node l to complete k iterations is given by:

$$S_l^k = \sum_{j=1}^{k} P_l^j \tag{2.3}$$

where $P_l^k \sim P$. This is actually a **renewal process** with **interarrival time** P_l (see [10], [11]). However if you consider the total number of iterations completed by all nodes together, then the time of the k'th iteration S^k is known as a **superposition of renewal processes**. From [11] 1.4, as $k \to \infty$ we have:

$$\frac{\mathbb{E}S^k}{k} \to \frac{\mathbb{E}P}{p}, \tag{2.4}$$

$$\text{(by convergence in the previus step) } \mathbb{E}S^k = k\frac{(C(m)+2\lambda)}{p}(1+o_{m,\lambda,p}(1)). \tag{2.5}$$

Remark 3. Notation. The subscripts in $o_{m,\lambda,p}(1)$ denote that this term converges to 0 as $k \to \infty$ in a way that depends on m, p, and λ.

Hence the expected time to complete K epochs is given by:

$$\mathbb{E}\mathcal{T}(K) = \frac{Km}{p}(C(m)+2\lambda)(1+o_{m,\lambda,p}(1)) \tag{2.6}$$

as $K \to \infty$. Hence it can be seen that asynchronous algorithms do not have a $\ln(p)$ penalty as p becomes larger, when K is sufficiently large. Hence for large K, asynchronous algorithms will compute at least $\Theta(\ln(p))$ more epochs per second than synchronous algorithms.

2.2.3 Heterogeneity and synchronous algorithms

Sometimes a parallel problem *cannot* be split into m blocks in a way that updating each block is of equal difficulty (as was previously assumed in this subsection). This can cause significant synchronization penalty in the synchronous case, but has no such effect on asynchronous algorithms because computing nodes do not have to wait for slower nodes or blocks to complete.

Let us assume for the moment that there is no random component of the update time for a single node, and that all nodes have the same computational power. This means that the update time for node l at iteration k is simply:

$$P_l^k = C(i_l, m) \tag{2.7}$$

where i_l is the block that node l updates at iteration k. Assume also that at every iteration, each node l will chose a random block to update, and hence i_l is a uniform random variable on $\{1, 2, \ldots, m\}$. For the synchronous algorithm, we have an update time:

$$P = \max_{l=1,2,\ldots,p} \{C(i_l, m)\} \tag{2.8}$$

Clearly then, as p increases, we have:

$$\mathbb{E}P \to \max_i C(i,m) \tag{2.9}$$

That is, the update time is determined by the most difficult block to update. Hence the the expected time for K epochs is:

$$\mathbb{E}\mathcal{T}(K) = \frac{Km}{p}\Big(\max_i C(i,m) + o_m(1)\Big) \tag{2.10}$$

as $p \to \infty$.

2.2.4 Heterogeneity and asynchronous algorithms

Now consider an asynchronous algorithm. The update time of a single node is:

$$\mathbb{E}P = \mathbb{E}C(i_l,m) = \frac{1}{m}\sum_{i=1}^{m} C(i,m) \tag{2.11}$$

Yet again we have a superposition of renewal processes, and hence from [11], we have as $k \to \infty$:

$$\frac{\mathbb{E}S^k}{k} \to \frac{\mathbb{E}P}{p} \tag{2.12}$$

$$\mathbb{E}S^k = \frac{k}{p}\left(\frac{1}{m}\sum_{i=1}^{m} C(i,m)\right)(1 + o_{m,p}(1)) \tag{2.13}$$

Hence the expected time for K epochs is given by:

$$\mathbb{E}\mathcal{T}(K) = \frac{Km}{p}\left(\frac{1}{m}\sum_{i=1}^{m} C(i,m)\right)(1 + o_{m,p}(1)) \tag{2.14}$$

as $K \to \infty$. Notice that the time taken for an asynchronous algorithm is determined by the average difficulty of updating a block. Compare this to synchronous algorithms where the most difficult block determines the time complexity. If the difficulty of blocks is highly heterogeneous, asynchronous algorithms may complete far more iterations per second, even without considering network effects.

2.2.5 Additional factors

In the previous, we merely gave an analysis of a couple factors that decrease the number of epochs that synchronous solvers complete. Another factors is heterogeneous computing power of the computing nodes themselves, which causes disadvantages even if all blocks are the same difficulty. Also there is significant overhead associated with enforcing synchronization, as well as read and write locks.

2.3 Iteration complexity for synchronous Block KM

In this subsection, we start to look at point 2 of the main argument in Section 1.1. In order to prove there is no iteration complexity penalty, we need to obtain tight rates of convergence for synchronous ARock (which

is merely a synchronous block KM iteration). At every step, each of the p processing nodes are given a random block to update with a KM-style iteration. Hence we have:

$$x^{k+1} = x^k - \eta^k P^k S x^k \tag{2.15}$$

Here P^k is a projection onto a random subset of $\{1, 2, \ldots, m\}$ of size p (we assume $p \leq m$). We note that each block has a $\frac{p}{m}$ probability of being updated on a given iteration.

Definition 4. Convergence rate. An algorithm is said to linearly converge if the error $E(k) = \mathcal{O}(R^k)$ for $0 < R < 1$. R is called the **convergence rate**.

Definition 5. Epoch iteration complexity. The epoch iteration complexity $I(\epsilon)$ is the number of **epochs** required to decrease the error below $\epsilon E(0)$, where $E(0)$ is the initial error.

This error could be the distance from the solution $\left\|x^k - x^*\right\|^2$ or the gap between the function value and its optimal value $f(x^k) - f^*$.

Proposition 6. Convergence rate of block KM iterations. *Let T be an r-Lipschitz operator for $0 < r < 1$. Consider the random subset KM iteration defined in Equation (2.15) for $1 \leq p \leq m$. A step size of $\eta^k = 1$ optimizes the convergence rate. When this optimal step size is chosen, we have convergence rate:*

$$R = 1 - \frac{p}{m}(1 - r^2)$$

and corresponding epoch iteration complexity:

$$I(\epsilon) = \left(\frac{1}{1 - r^2} - \theta\frac{p}{m}\right)\ln(1/\epsilon) \tag{2.16}$$

for some $\theta \in \left[\frac{1}{2}, 1\right]$.

For problems of interest, the first term $(1/(1 - r^2))$ will dominate the second. We are interested in huge-scale problems, which will usually have $m \gg p$ or $r \approx 1$.

This is proven in Appendix A.1. Hence if we have either $r \to 1$ or $p/m \to 0$, then:

$$I = (1 + o(1))\frac{1}{1 - r^2}\ln(1/\epsilon) \tag{2.17}$$

We will eventually prove that ARock has essentially the same iteration complexity.

This result allows us to obtain a sharp convergence rate and epoch iteration complexity for synchronous-parallel block gradient descent (of which block gradient descent is a special case of $p = 1$). This appears to be a new result that extends recent work in [12] to the case of random-subset block gradient descent (which corresponds to the special case $m = 1$, $p = 1$).

Corollary 7. Sharp Convergence Rate of Synchronous-Parallel Block Gradient Descent. *Let f be μ-strongly convex function with L-Lipcshitz gradient $\nabla f(x)$. Let $\kappa = L/\mu$ be the condition number. The operator $T = I - \frac{2}{\mu + L}\nabla f$ is $r = \left(1 - \frac{2}{\kappa + 1}\right)$ Lipschitz. The corresponding block KM iteration Equation (2.15) with optimal step size $\eta^k = 1$ is equivalent to synchronous-parallel block gradient descent with step size*

$2/(\mu+L)$. *The linear convergence rate* R *and epoch iteration complexity* $I(\epsilon)$ *with respect to the error* $\left\|x^k - x^*\right\|^2$ *are given by the following:*

$$R = 1 - 4\frac{p}{m}\frac{\kappa}{(\kappa+1)^2} \qquad = 1 - 4\frac{p}{m\kappa}(1+\mathcal{O}(1/\kappa)) \tag{2.18}$$

$$I(\epsilon) = \frac{1}{4}(\kappa + \mathcal{O}(1))\ln(1/\epsilon) \tag{2.19}$$

as $\kappa \to \infty$. *Lastly, this convergence rate is sharp.*

This is proven in Appendix A.2. Among other things, our main results will show that asynchronous-parallel block-gradient descent has epoch iteration complexity that is asymptotically equal to $\frac{1}{4}\kappa\ln(1/\epsilon)$, which is the complexity of synchronous-parallel block coordinate descent.

Remark 4. Composite objectives. Let $g(x) = \sum_{i=1}^n g(x_i)$ be separable, with each component convex, and subdifferentiable. The exact same convergence rate and complexity clearly holds for block proximal gradient descent. This is because the corresponding nonexpansive operator $T = (I+\partial g)^{-1} \circ \left(I - \frac{2}{\mu+L}\nabla f\right)$ is also $1 - \frac{2}{\kappa+1}$-Lipschitz, and hence all preceding theory applies.

2.4 Iteration complexity for ARock

In this subsection we present our theoretical results for ARock, which completes part 2 of the main argument in Section 1.1. That is, ARock (and hence all its special cases) has essentially the same iteration complexity as its synchronous counterpart. However we present the results in simplified form so as to more effectively communication the main message. Full versions of these results that contain more technical details are given and proven in the proof sections.

2.4.1 Stochastic delays

The first result is convergence under a stochastic unbounded delays from a fixed distribution (though this can be weakened to a changing distribution).

Assumption 2. Stochastic unbounded delays. The sequence of delay vectors $\vec{j}(0), \vec{j}(1), \vec{j}(2), \ldots$ is IID, and $\vec{j}(k)$ is independent of $\mathcal{G}^k$. That is, $\vec{j}(k)$ is indepdendent of the iterates $\left(x^0, x^1, x^2, \ldots, x^k\right)$.

We can imagine a large optimization problem being solved on a busy network. Nodes are continually sending and receiving their updates. Traffic is chaotic and there is some kind of distribution of how long the information take to get from one node to the rest.

The delay vectors $\vec{j}(k)$ have a fixed distribution (though this can be relaxed). Define

$$P_l = \mathbb{P}[j(k) \geq l] \tag{2.20}$$

We let ρ be defined by:

$$\rho = 1 - \frac{1}{m}\left(1 - r^2\right) \tag{2.21}$$

which is the linear convergence rate of the corresponding synchronous KM algorithm (Equation (2.15)) with $p = 1$, and ideal step size. We also define probability moments:

$$M_1 = \sum_{l=1}^{\infty} P_l \rho^{-l/2}, \qquad M_2 = \sum_{l=1}^{\infty} P_l^{1/2} \rho^{-l/2} \tag{2.22}$$

and step size:

$$\eta^k = \eta_1 \triangleq \left(1 + m^{-1/2}\left(\left(1 - r^2\right)^{1/2} M_1 + 2M_2\right)\right) \tag{2.23}$$

Notice that these moments are a function of m. This is immediately clear because ρ is a function of m. Also the probability distribution of the delays may depend on m in a way that depends on the network and how you decide to scale up the computation to a higher number of nodes or blocks.

Theorem 1. Linear convergence for stochastic delays. *Let Assumption 1 and Assumption 2 hold. Let M_1, and M_2 be finite and $\mathcal{O}(m^q)$ for $0 \leq q < 1/2$. Let $\eta^k = \eta_1$. Then there exist positive coefficients $(c_i)_{i=1}^{\infty}$ of the Lyapunov function ξ^k such that we have the following linear convergence rate and iteration complexity respectively:*

$$\mathbb{E}\left[\xi^{k+1} \middle| \mathcal{G}^k\right] \leq \underbrace{\left(1 - \frac{1}{m}\left(1 - r^2\right)\right)}_{\text{Ideal synchronous rate}} + \underbrace{\mathcal{O}\left(m^{q-3/2}\right)}_{\text{Asynchronicity penalty}} \xi^k, \tag{2.24}$$

$$I(\epsilon) = \left(1 + \mathcal{O}\left(m^{-1/2+q}\right)\right)\left(\frac{1}{1 - r^2}\right)\ln(1/\epsilon), \tag{2.25}$$

as $m \to \infty$.

The full version of this is Theorem 2, which is proven in Section 3.5 after a series of results built up in Section 3.

Comparing this to Proposition 6, we can see that as $m \to \infty$, the iteration complexities of asynchronous and synchronous ARock approach the same value. Thus at scale there is no iteration complexity penalty for using outdated information, so long as that information does not become too old as you scale up the problem (i.e. M_1 and M_2 don't grow too fast as m increases.).

Remark 5. Reduction to synchronous case. We will see that when there is no asynchronicity, the Lyapunov function will reduce to the classical error $\xi^k = \left\|x^k - x^*\right\|^2$, and $\eta_1 = 1$. Also the convergence rate and iteration complexity will exactly equal the values obtained in Proposition 6 (again for $p = 1$).

2.4.2 Deterministic delays

We also prove a similar convergence result for deterministic unbounded delays. However we present the section in Section 4, because the specifics of the theorem are a little subtle. The proof of the stochastic delay result can be seen as a warmup to the deterministic result.

2.4.3 Completing the main argument

Hence our convergence rate results complete point 2 of the main argument. We have *more epochs* per given time period, and the epochs make the same progress because iterations are of the *same quality*. Hence asynchronous algorithms may drastically outperform synchronous ones in this setting.

2.5 Related Work

Though asynchronous algorithms were invented long ago, there has been a lot of recent interest, especially for random-block-coordinate-type algorithms, such as "Hogwild!" [13]. In [14], the authors prove linear convergence for an asynchronous stochastic linear solver. In [15], the authors prove function-value convergence for asynchronous stochastic coordinate descent. They prove $\mathcal{O}(1/k)$ convergence for f convex with ∇f Lipschitz, and linear convergence when f is also strongly convex. This was extended in [16] to composite objective functions.

For condition number κ, they report a per-iteration linear convergence rate of

$$1 - \frac{1}{2m\kappa} \tag{2.26}$$

This implies an iteration complexity approximately 8 times higher than our result (where our result matches asymptotically to the complexity of the corresponding synchronous algorithm). For asynchronicity to be useful, the reduced penalty would need to compensate for this 8-fold increase in iterations. Like almost all recent work except for [2], [17], they assume a bounded delay τ. For linear speedup, they require $\tau = \mathcal{O}(m^{1/2})$ and $\tau = \mathcal{O}(m^{1/4})$ for composite objectives. We do not require bounded delays for linear speedup, only sufficiently slowly growing moments of delay. For bounded delay our condition is $\tau = \mathcal{O}(m^q)$ for $0 \leq q < \frac{1}{2}$ for composite and non-composite objectives.

Our work is also more general, since we use the operator setting. So not only do the main results apply to gradient descent, and proximal gradient, but any other algorithm that can be written in a block fixed-point form.

In [18], authors achieve a linear speedup but with a far higher iteration complexity of $\Theta\big(\kappa^2 \ln(1/\epsilon)\big)$. When the iteration complexity is increased by a factor of κ, the linear speedup may be of limited utility. We also note that our conditions for linear speedup are weaker than theirs (which is $\tau = \mathcal{O}\big(m^{1/6}\big)$), and they need to assume that x^k remains bounded, which is unjustified.

In [17] prove function-value linear convergence of an asynchronous block proximal gradient algorithm under unbounded delays. However it is unclear how the iteration complexity they obtain compares to the corresponding synchronous algorithm. Also our result applies to a KM iteration, of which block proximal gradient is a special case.

In [19], the authors review a number of asynchronous algorithm analyses and collect conditions necessary for linear speedup on a fixed problem. In light of potentially increased complexity, as seen in [18], a linear speedup does not imply that asynchronous algorithms will run faster in time. It only implies that the potential slowdown factor for using asynchronicity is bounded for a given problem, but this bound may be as large as κ. What we prove in this paper is much stronger, that the slowdown factor (in terms of iterations) for using asynchronicity is asymptotically 1, i.e. that the slowdown is negligible.

2.6 Unbounded delays

Almost all work on asynchronous algorithms except for [2], [17] assumed bounded delays. That is, there was a limit τ such that $j(k) \leq \tau$ for all k. However there may be no bound on the delay in practice: There is always the possibility of an arbitrarily large network delay. Also this τ needs to be known in advance in order to set the step size, which is impractical. A large τ will hinder the convergence rate by limiting the step size, even if a delay of τ is extremely unlikely. In this work we assume no bound on the delay. In Theorem 1, we are able to obtain fast convergence if the delay distribution is not too spread out. In Theorem 3 we are able

to determine a convergence rate that depends only on the current delay conditions, not on the worst case behavior of delay on the entire time that the algorithm runs.

3 Analysis of ARock under Stochastic Delays

We now give the full version of Theorem 1. We let the coefficients in ξ^k be given by:

$$c_i = m^{1/2}\Big(1+\big(1-r^2\big)^{1/2}\Big)\sum_{l=i}^{\infty} P_l^{-1/2}\rho^{-(l/2-i+1)} \tag{3.1}$$

and define the constant:

$$\eta_2 = m^{1/2}\big(1-r^2\big)^{-1/2}M_1^{-1} \tag{3.2}$$

Finally, define the following **convergence rate function**:

$$R(\eta,\gamma) = \Big(1-\frac{\eta}{m}\big(1-r^2\big)(1-\eta/\gamma)\Big) \tag{3.3}$$

Note that we have $R<1$ when $0<\eta<\gamma$, and the rate is optimized when $\eta=(1/2)\gamma$. Also $\rho\le R$ for $\eta\le 1$, where $\rho = 1-(1/m)(1-r^2)$ is the optimal rate that we wish to prove R is close to.

Theorem 2. Linear convergence for stochastic delays. *Let Assumption 1 and Assumption 2 hold. Let the step size η^k be $\mathcal{F}^k$-measurable, and satisfy $\eta^k\le\eta_1$. Let the probability moments M_1 and M_2 defined in Equation (2.22) be finite. Consider the Lyapunov function defined in Equation (1.11) with constants given by Equation (3.1). Then we have the following linear convergence rate:*

$$\mathbb{E}\big[\xi^{k+1}\big|\mathcal{G}^k\big] \le R\big(\eta^k,\eta_2\big)\xi^k \tag{3.4}$$

Additionally, let $\eta^k=\eta_1$ and assume that $M_1, M_2 = \mathcal{O}(m^q)$ for $0\le q<1/2$. As the number of blocks m approaches ∞, we have the following linear convergence rate and iteration complexity respectively:

$$R(\eta_1,\eta_2) = \underbrace{\Big(1-\frac{1}{m}\big(1-r^2\big)\Big)}_{\textit{Ideal synchronous rate}} + \underbrace{\frac{2}{m^{3/2}}\big(1-r^2\big)^{3/2}\Big(\big(1-r^2\big)^{1/2}M_1+M_2\Big)}_{\textit{Asynchronous rate penalty}}(1+o(1)) \tag{3.5}$$

$$I(\epsilon) = \left(1+\underbrace{2m^{-1/2}\Big(\big(1-r^2\big)^{1/2}M_1+M_2\Big)}_{\textit{Highest order penalty term}}+o\Big(m^{-1/2+q}\Big)\right)\left(\frac{1}{1-r^2}\right)\ln(1/\epsilon) \tag{3.6}$$

We chose to include the exact form of the highest order iteration complexity penalty. This allows us to calculate approximately how large m has to be in order for asynchronicity to cause negligible penalty.

We now prove Theorem 2 in a way that emphasizes the reasons and intuition behind our approach – especially the strategic way in which the coefficients are chosen.

3.1 Preliminary results

Let x^* be any solution, and set $x^* = 0$ with no loss in generality, to make the notation more compact. This can be achieved by translating the origin of the coordinate system to x^*. Hence $\|x^k\|$ is the distance from the solution. The starting point of our analysis is the following[10]:

$$\begin{aligned}\mathbb{E}\Big[\|x^{k+1}\|^2|\mathcal{F}^k\Big] &= \mathbb{E}\Big[\|x^k - \eta^k S_{i(k)}\hat{x}^k\|^2|\mathcal{F}^k\Big] \\ &= \|x^k\|^2 + \mathbb{E}\Big[-2\eta^k\langle x^k, S_{i(k)}\hat{x}^k\rangle + (\eta^k)^2\|S_{i(k)}\hat{x}^k\|^2|\mathcal{F}^k\Big].\end{aligned}$$

Here the expectation is taken over only the block index $i(k)$ (Recall Assumption 1). We let the step size η^k be $\mathcal{G}^k$-measurable (and hence $\mathcal{F}^k$-measurable). Essentially this means that the step size η^k can depend only on the sequence $(x^0, x^1, \ldots, x^k)$, and not the block index or delay. Hence

$$\mathbb{E}\Big[\|x^{k+1}\|^2|\mathcal{F}^k\Big] = \|x^k\|^2 \underbrace{-2\frac{\eta^k}{m}\langle x^k, S\hat{x}^k\rangle}_{\text{cross term}} + \frac{(\eta^k)^2}{m}\|S\hat{x}^k\|^2. \tag{3.7}$$

We now present a simple lemma on the operator S that will be used in the convergence proof (This is the operator version of Theorem 2.1.12 in [20]).

Lemma 8. *Let $S = I - T$, where T is an r-Lipschitz operator. Then for all $x, y \in \mathbb{H}$ we have:*

$$\langle Sy - Sx, y - x\rangle \geq \frac{1}{2}\|Sy - Sx\|^2 + \frac{1}{2}(1 - r^2)\|y - x\|^2 \tag{3.8}$$

Proof.

$$\begin{aligned}(T \text{ is } r\text{-Lipschitz})\ r^2\|y - x\|^2 &\geq \|Ty - Tx\|^2 \\ &= \|(I - S)y - (I - S)x\|^2 \\ &= \|Sy - Sx\|^2 - 2\langle Sy - Sx, y - x\rangle + \|y - x\|^2 \\ (\text{rearrange})\ \langle Sy - Sx, y - x\rangle &\geq \frac{1}{2}\|Sy - Sx\|^2 + \frac{1}{2}(1 - r^2)\|y - x\|^2\end{aligned}$$

□

3.2 The cross term

Remark 6. Strategy. Consider Equation (3.7) again. The $\|S\hat{x}^k\|^2$ term in Equation (3.7) can be thought of as a "waste" term that has to be negated. In light of Lemma 8, a $-\langle S\hat{x}^k, \hat{x}^k\rangle$ term can be used to generate a $-\|S\hat{x}^k\|^2$ term to clean this waste. In addition to cleaning this waste, ideally we would have a $-\|x^k\|^2$ to help prove linear convergence, but instead Lemma 8 produces $-\|\hat{x}^k\|^2$.

The strategy we pursue is as follows: The cross term $-\langle S\hat{x}^k, x^k\rangle$ is approximately equal to $-\langle S\hat{x}^k, \hat{x}^k\rangle$, which allows us to clean the $\|S\hat{x}^k\|^2$ term. The $-\|\hat{x}^k\|^2$ that is also generated is approximately equal to $-\|x^k\|^2$, which helps prove linear convergence. However, there is an error associated with this "conversion". Finally, this conversion error is negated by the use of as Lyapunov function (see eq. (1.11)).

[10] We will use an abuse of notation in this paper. We equate $S_i(x) \in \mathbb{H}_i$ (the components of $S(x)$ in the ith block) and $(0, \ldots, 0, S_i(x), 0, \ldots, 0) \in \mathbb{H}_1 \times \ldots \times \mathbb{H}_m$ (the projection of $S(x)$ to the i'th subspace). Hence we can write the ARock iteration more compactly as $x^{k+1} = x^k - \eta^k S_{i(k)}\hat{x}^k$.

Lemma 9 will eventually allow us to quantify the error associated with converting $-\langle S\hat{x}^k, x^k\rangle$ to $-\langle S\hat{x}^k, \hat{x}^k\rangle$, and the error associated with converting $-\|\hat{x}^k\|^2$ to $-\|x^k\|^2$ mentioned in Remark 6.

Lemma 9. *Let $a > 0$, $j(k)$ be the current delay, η^k be the current step size, and $\epsilon_1, \epsilon_2, \ldots > 0$ be a series of parameters. Then we have:*

$$a\|x^k - \hat{x}^k\| \leq \frac{1}{2}a^2\eta^k\left(\sum_{i=1}^{j(k)}\frac{1}{\epsilon_i}\right) + \frac{1}{2}\frac{1}{\eta^k}\sum_{i=1}^{j(k)}\left(\epsilon_i\|x^{k+1-i} - x^{k-i}\|^2\right) \tag{3.9}$$

Proof. See [1], [2]. □

Remark 7. Free parameters. Lemma 10 generates some positive parameters $\epsilon_1, \epsilon_2, \ldots > 0$ and $\delta_1, \delta_2, \ldots > 0$. It's not immediately clear what these parameters should be set to. However we will see in Section 3.4 if they are properly chosen, they can be used to construct a Lyapunov function that will allow us to prove linear convergence.

We will make use of Lemma 9 twice with parameter sets $(\epsilon_1, \epsilon_2, \ldots)$ and $(\delta_1, \delta_2, \ldots)$ respectively. To simplify notation, we define:

$$E_j = \sum_{i=1}^{j}\frac{1}{\epsilon_i} \qquad D_j = \sum_{i=1}^{j}\frac{1}{\delta_i} \tag{3.10}$$

Lemma 10. *Let Assumption 1 hold. Let $\epsilon_1, \epsilon_2, \ldots > 0$ and $\delta_1, \delta_2, \ldots > 0$ be a sequence of parameters, and E_j, D_j defined as above. Let η^k be $\mathcal{G}^k$-measurable. ARock yields the following inequality:*

$$\begin{aligned}\mathbb{E}\left[\|x^{k+1}\|^2|\mathcal{F}^k\right] \leq{}& \left(1 - \frac{\eta^k}{m}(1-r^2)(1-\eta^k D_{j(k)})\right)\|x^k\|^2 + \frac{1}{m}\sum_{i=1}^{j(k)}(\delta_i(1-r^2)+\epsilon_i)\|x^{k+1-i} - x^{k-i}\|^2 \\ &- \frac{\eta^k}{m}\|S\hat{x}^k\|^2(1-\eta^k(1+E_{j(k)}))\end{aligned}$$

Proof. We make use of Lemma 9 twice in this proof, with parameter sets $(\epsilon_1, \epsilon_2, \ldots)$ and $(\delta_1, \delta_2, \ldots)$ respectively.

$$\begin{aligned}&-2\frac{\eta^k}{m}\langle x^k, S\hat{x}^k\rangle \\ &= -2\frac{\eta^k}{m}\langle \hat{x}^k, S\hat{x}^k\rangle - 2\frac{\eta^k}{m}\langle x^k - \hat{x}^k, S\hat{x}^k\rangle \\ &\leq -\frac{\eta^k}{m}\left(\|S\hat{x}^k\|^2 + (1-r^2)\|\hat{x}\|^2\right) + 2\frac{\eta^k}{m}\|x^k - \hat{x}^k\|\cdot\|S\hat{x}^k\| \\ &\leq -\frac{\eta^k}{m}\left(\|S\hat{x}^k\|^2 + (1-r^2)\|\hat{x}^k\|^2\right) + 2\frac{\eta^k}{m}\left(\frac{1}{2}\|S\hat{x}^k\|^2\eta^k E_{j(k)} + \frac{1}{2}\frac{1}{\eta^k}\sum_{i=1}^{j(k)}\left(\epsilon_i\|x^{k+1-i} - x^{k-i}\|^2\right)\right) \\ &= -\frac{\eta^k}{m}(1-r^2)\|\hat{x}^k\|^2 + \frac{1}{m}\sum_{i=1}^{j(k)}\epsilon_i\|x^{k+1-i} - x^{k-i}\|^2 - \frac{\eta^k}{m}\|S\hat{x}^k\|^2(1-\eta^k D_{j(k)})\end{aligned} \tag{3.11}$$

For sufficiently small step size, this inequality allows us to negate the $\|S\hat{x}^k\|^2$ terms. Now let's examine $-\|\hat{x}^k\|^2$, which we convert to a $-\|x^k\|^2$ term (and some error) for linear convergence.

$$\begin{aligned}
-\|\hat{x}^k\|^2 &= -\|x^k\|^2 - 2\langle \hat{x}^k - x^k, x^k\rangle - \|x^k - \hat{x}^k\|^2 \\
&\leq -\|x^k\|^2 + 2\|\hat{x}^k - x^k\|\|x^k\| \\
\text{(Lemma 9)} &\leq -\|x^k\|^2 + \|x^k\|^2\eta^k D_{j(k)} + \frac{1}{\eta^k}\sum_{i=1}^{j(k)}\Big(\delta_i\|x^{k+1-i} - x^{k-i}\|^2\Big) \\
&= -\big(1-\eta^k D_{j(k)}\big)\|x^k\|^2 + \frac{1}{\eta^k}\sum_{i=1}^{j(k)}\Big(\delta_i\|x^{k+1-i} - x^{k-i}\|^2\Big)
\end{aligned}$$

Hence substituting into (3.11), we have

$$\begin{aligned}
-2\frac{\eta^k}{m}\langle x^k, S\hat{x}^k\rangle \leq{}& -\frac{\eta^k}{m}(1-r^2)\big(1-\eta^k D_{j(k)}\big)\|x^k\|^2 + \frac{\eta^k}{m}(1-r^2)\frac{1}{\eta^k}\sum_{i=1}^{j(k)}\Big(\delta_i\|x^{k+1-i} - x^{k-i}\|^2\Big) \\
&+ \frac{1}{m}\sum_{i=1}^{j(k)}\epsilon_i\|x^{k+1-i} - x^{k-i}\|^2 - \frac{\eta^k}{m}\|S\hat{x}^k\|^2\big(1-\eta^k D_{j(k)}\big) \\
={}& -\frac{\eta^k}{m}(1-r^2)\big(1-\eta^k D_{j(k)}\big)\|x^k\|^2 + \frac{1}{m}\sum_{i=1}^{j(k)}\big(\delta_i(1-r^2)+\epsilon_i\big)\|x^{k+1-i} - x^{k-i}\|^2 \\
&- \frac{\eta^k}{m}\|S\hat{x}^k\|^2\big(1-\eta^k D_{j(k)}\big)
\end{aligned}$$

Using (3.7) immediately yields the result. □

3.3 The Lyapunov function

We now consider how the Lyapunov function defined in Equation (1.11) changes in size from step to step. The reason that a Lyapunov function is needed is to deal with the $\|x^{k+1-i} - x^{k-i}\|^2$ terms. They cannot be negated like $\|S\hat{x}^k\|^2$ terms, and so must be incorporated into the error by using a Lyapunov function. Let $P_l = \mathbb{P}[j(k) \geq l]$.

Lemma 11. *Let the conditions of Lemma 10 and Assumption 2 hold. Define*

$$\eta_1 = \left(1 + \frac{c_1}{m} + \left\|\frac{1}{\epsilon_i}\right\|_{\ell^1}\right)^{-1} \tag{3.12}$$

$$\eta_2 = \left(\sum_{i=1}^{\infty}\frac{P_i}{\delta_i}\right)^{-1} \tag{3.13}$$

$$R(\eta,\gamma) = \Big(1 - \frac{\eta}{m}(1-r^2)(1-\eta/\gamma)\Big) \tag{3.14}$$

Let η^k be $\mathcal{G}^k$-measurable, and $\eta^k \leq \eta_1$. Then ARock satisfies:

$$\mathbb{E}\left[\xi^{k+1}\middle|\mathcal{G}^k\right] \leq \left\|x^k\right\|^2 R(\eta^k,\eta_2) + \frac{1}{m}\sum_{i=1}^{\infty}\left(\left(\epsilon_i + \left(1-r^2\right)\delta_i\right)P_i + c_{i+1}\right)\left\|x^{k+1-i} - x^{k-i}\right\|^2$$

Notice that we have defined η_1 and η_2 in terms of the unspecified parameters $(\epsilon_1, \epsilon_2, \ldots)$ and $(\delta_1, \delta_2, \ldots)$. Eventually, we will set $\epsilon_i = m^{1/2}P_i^{-1/2}\rho^{i/2}$ and $\delta_i = m^{1/2}(1-r^2)^{-1/2}\rho^{i/2}$ for reasons that will be explained in Section 3.5. With this parameter choice, the definitions of η_1 and η_2 will match eq. (2.23) and eq. (3.2) respectively.

Proof.

$$\mathbb{E}\left[\xi^{k+1}\middle|\mathcal{F}^k\right] = \underbrace{\mathbb{E}\left[\left\|x^{k+1}\right\|^2\middle|\mathcal{F}^k\right]}_{A} + \underbrace{\frac{c_1}{m}\mathbb{E}\left[\left\|x^{k+1}-x^k\right\|^2\middle|\mathcal{F}^k\right]}_{B} + \underbrace{\frac{1}{m}\sum_{i=1}^{\infty}c_{i+1}\left\|x^{k+1-i}-x^{k-i}\right\|^2}_{C} \tag{3.15}$$

We obtain a bound on A from Lemma 10. B follows by the definition of ARock:

$$B = \frac{c_1}{m}\frac{\left(\eta^k\right)^2}{m}\left\|S\hat{x}^k\right\|^2.$$

C contains no expectation because it is $\mathcal{F}^k$ measurable. Hence we have:

$$\begin{aligned}\mathbb{E}\left[\xi^{k+1}\middle|\mathcal{F}^k\right] \leq{}& \left\|x^k\right\|^2\left(1 - \frac{\eta^k}{m}\left(1-r^2\right)\left(1-\eta^k\left(D_{j(k)}\right)\right)\right) - \frac{\eta^k}{m}\left\|S\hat{x}^k\right\|^2\left(1-\eta^k\left(1+\frac{c_1}{m}+E_{j(k)}\right)\right)\\ &+ \frac{1}{m}\sum_{i=1}^{j(k)}\left(\epsilon_i + \left(1-r^2\right)\delta_i\right)\left\|x^{k+1-i}-x^{k-i}\right\|^2 + \frac{1}{m}\sum_{i=1}^{\infty}c_{i+1}\left\|x^{k+1-i}-x^{k-i}\right\|^2\end{aligned} \tag{3.16}$$

Notice that $E_j = \sum_{i=1}^{j} 1/\epsilon_i \leq \|1/\epsilon_i\|_{\ell^1}$ for all j, and that therefore the step size condition eliminates the $\left\|S\hat{x}^k\right\|^2$ term.

Now it becomes necessary to take expectations over the delay distribution (by taking the expectation with respect to $\mathcal{G}^k$ instead of $\mathcal{F}^k$). Notice that for a sequence $(\gamma_1, \gamma_2, \ldots)$, we have: $\mathbb{E}\left[\sum_{i=1}^{j(k)}\gamma_i\middle|\mathcal{G}^k\right] = \sum_{i=1}^{\infty}P_i\gamma_i$. This yields:

$$\begin{aligned}\mathbb{E}\left[\xi^{k+1}\middle|G^k\right] \leq{}& \left\|x^k\right\|^2\left(1 - \frac{\eta^k}{m}\left(1-r^2\right)\left(1-\eta^k\left(\sum_{i=1}^{\infty}\frac{P_i}{\delta_i}\right)\right)\right)\\ &+ \frac{1}{m}\sum_{i=1}^{\infty}\left(\epsilon_i + \left(1-r^2\right)\delta_i\right)P_i\left\|x^{k+1-i}-x^{k-i}\right\|^2 + \frac{1}{m}\sum_{i=1}^{\infty}c_{i+1}\left\|x^{k+1-i}-x^{k-i}\right\|^2\end{aligned}$$

which completes the proof. ∎

3.4 Linear convergence

The right-hand side in Lemma 11 closely resembles ξ^k. Ideally, we have:

$$\mathbb{E}\big[\xi^{k+1}\big|\mathcal{G}^k\big] \leq \gamma\xi^k \tag{3.17}$$

for $0 < \gamma < 1$, and some choice of parameters $(\epsilon_1, \epsilon_2, \ldots)$, $(\delta_1, \delta_2, \ldots)$ and coefficients $(c_1, c_2, \ldots)$. In this section we will derive such a result by carefully chosing these parameters. However, in order to derive a coefficient formula, we need the following lemma.

Lemma 12. Coefficient formula. *Let $0 < \rho < 1$ and let $(s_1, s_2, \ldots)$ be a positive sequence. Consider the coefficient formula:*

$$c_i = \sum_{l=i}^{\infty} s_l \rho^{-(l-i+1)}. \tag{3.18}$$

If $c_1 < \infty$, then we have $c_i \downarrow 0$ and:

$$\rho c_i = c_{i+1} + s_i \tag{3.19}$$

Proof.

$$\rho c_i = \sum_{l=i}^{\infty} s_l \rho^{-(l-i)} = \sum_{l=i+1}^{\infty} s_l \rho^{-(l-(i+1)+1)} + s_i = c_{i+1} + s_i$$

Clearly this implies $c_i \downarrow 0$, since $\rho < 1$ and coefficients are nonnegative. □

Recall that ρ is defined in eq. (2.21).

Proposition 13. Linear convergence for stochastic delays. *Let Assumption 1 hold. Let $\eta^k \leq \eta_1$, and let $\epsilon_1, \epsilon_2, \ldots > 0$ and $\delta_1, \delta_2, \ldots > 0$ be a sequence of parameters. Let*

$$\sum_{l=i}^{\infty} \big(\epsilon_l + \big(1 - r^2\big)\delta_l\big) P_l \rho^{-l} < \infty \tag{3.20}$$

With the choice of coefficients[11]*:*

$$c_i = \sum_{l=i}^{\infty} \big(\epsilon_l + \big(1 - r^2\big)\delta_l\big) P_l \rho^{-(l-i+1)} \tag{3.21}$$

We have the following linear convergence result:

$$\mathbb{E}\big[\xi^{k+1}\big|\mathcal{G}^k\big] \leq R\big(\eta^k, \eta_2\big)\xi^k \tag{3.22}$$

[11] This formula will eventually match eq. (3.1) when the parameters $\epsilon_1, \epsilon_2, \ldots > 0$ and $\delta_1, \delta_2, \ldots > 0$ are chosen later.

Proof. By applying Lemma 12 with $s_l = P_l(\epsilon_l + (1-r^2)\delta_l)$, we obtain:

$$\rho c_i = \big(\epsilon_i + (1-r^2)\delta_i\big)P_i + c_{i+1}$$

Hence from Lemma 11, we have:

$$\begin{aligned}\mathbb{E}\big[\xi^{k+1}\big|\mathcal{G}^k\big] &\le \big\|x^k\big\|^2 R\big(\eta^k,\eta_2\big) + \frac{1}{m}\sum_{i=1}^{\infty}\big(\big(\epsilon_i + (1-r^2)\delta_i\big)P_i + c_{i+1}\big)\big\|x^{k+1-i} - x^{k-i}\big\|^2 \\ &\le \big\|x^k\big\|^2 R\big(\eta^k,\eta_2\big) + \frac{1}{m}\sum_{i=1}^{\infty}\rho c_i\big\|x^{k+1-i} - x^{k-i}\big\|^2 \le \max\big(\rho, R\big(\eta^k,\eta_2\big)\big)\xi^k = \rho\xi^k\end{aligned}$$ □

The last line follows, because $0 \le \eta^k \le \eta_1 \le 1$ implies $\rho \le R(\eta^k, \eta_2)$.

3.5 Proof of Theorem 2

Recall we have the following step size restriction $\eta^k \le \eta_1$ (with η_1 defined in eq. (3.12)) coupled with the convergence rate:

$$R\big(\eta^k,\eta_2\big) = \left(1 - \frac{\eta^k}{m}(1-r^2)\big(1-\eta^k/\eta_2\big)\right), \text{ for } \eta_2 = \left(\sum_{i=1}^{\infty}\frac{P_i}{\delta_i}\right)^{-1}$$

We now prove Theorem 2. However we do so in a way that justifies the choice of parameters that we use. In the case of $(\epsilon_1, \epsilon_2, \ldots)$ there is a best choice. For other parameters, we simply pick a sensible though not necessarily optimal choice. First though, a couple of lemmas are needed to look at the asymptotic convergence rate and iteration complexity.

Lemma 14. *Say that as $m \to \infty$, we have $x(m) = \mathcal{O}(1)$, $y(m) = \mathcal{O}(1)$, and:*

$$\begin{aligned}t_1 &= 1 + x(m)m^{-a} + o\big(m^{-a}\big),\ 1 > a > 0 \\ t_2^{-1} &= y(m)m^{-b} + o\big(m^{-b}\big),\ 1 > b > 0\end{aligned}$$

Then

$$R(t_1,t_2) = 1 - \frac{1}{m}(1-r^2)\big(1 - \big(x(m)m^{-a} + y(m)m^{-b}\big) + o\big(m^{-a}+m^{-b}\big)\big)$$

Proof.

$$\begin{aligned}R(t_1,t_2) &= 1 - \frac{1}{m}(1-r^2)t_1\big(1-t_1t_2^{-1}\big) \\ t_1\big(1-t_1t_2^{-1}\big) &= \big(1 - xm^{-a} + o\big(m^{-a}\big)\big)\big(1 - \big(1 - xm^{-a} + o\big(m^{-a}\big)\big)\big(ym^{-b} + o\big(m^{-b}\big)\big)\big) \\ &= \big(1 - xm^{-a} + o\big(m^{-a}\big)\big)\big(1 - \big(1 + \mathcal{O}\big(m^{-a}\big)\big)\big(ym^{-b} + o\big(m^{-b}\big)\big)\big) \\ &= \big(1 - xm^{-a} + o\big(m^{-a}\big)\big)\big(1 - ym^{-b} + o\big(m^{-b}\big)\big) \\ &= 1 - xm^{-a} - ym^{-b} + o\big(m^{-a}+m^{-b}\big) \\ R(t_1,t_2) &= 1 - \frac{1}{m}(1-r^2)\big(1 - \big(xm^{-a} + ym^{-b}\big) + o\big(m^{-a}+m^{-b}\big)\big)\end{aligned}$$ □

Lemma 15. *Let $x(m) = \mathcal{O}(1)$, and $0 \leq a$. If the linear convergence rate R satisfies:*

$$R = 1 - \frac{1}{m}\big(1 - r^2\big)\big(1 - (xm^{-a} + o\big(m^{-a}\big))\big) \tag{3.23}$$

as $m \to \infty$. Then we have:

$$I(\epsilon) = \big(1 + xm^{-a} + o\big(m^{-a}\big)\big)\left(\frac{1}{1 - r^2}\right)\ln(1/\epsilon) \tag{3.24}$$

Proof.

$$\begin{aligned}
\epsilon &= (R(\eta_1, \eta_2))^{I(\epsilon)m} \\
\ln(1/\epsilon) &= -I(\epsilon)m\ln(R(\eta_1, \eta_2)) \\
&= I(\epsilon)m\left(\frac{1}{m}\big(1 - r^2\big)\big(1 - xm^{-a} + o\big(m^{-a}\big)\big) + \mathcal{O}\left(\left(\frac{1}{m}\big(1 - r^2\big)\big(1 - xm^{-a} + o\big(m^{-a}\big)\big)\right)^2\right)\right) \\
&= I(\epsilon)m\left(\frac{1}{m}\big(1 - r^2\big)\big(1 - xm^{-a} + o\big(m^{-a}\big)\big) + \mathcal{O}\left(\left(\frac{1}{m}\big(1 - r^2\big)\mathcal{O}(1)\right)^2\right)\right) \\
&= I(\epsilon)m\left(\frac{1}{m}\big(1 - r^2\big)\big(1 - xm^{-a} + o\big(m^{-a}\big)\big)\right) \\
I(\epsilon) &= \big(1 + xm^{-a} + o\big(m^{-a}\big)\big)\left(\frac{1}{1 - r^2}\right)\ln(1/\epsilon)
\end{aligned}$$

□

Proof of Theorem 2. We start with the conditions of Proposition 13. M_1 and M_2 being finite corresponds to Equation (3.20) in Proposition 13.

It is immediately possible to maximize η_1 over the sequence ϵ_i, by letting $\epsilon_i = \sqrt{m}P_i^{-1/2}\rho^{i/2}$. All thing being equal, increasing η_1 allows for a better convergence rate by increasing the range of possible step sizes. This leads to:

$$\eta_1 = \left(1 + m^{-1}\big(1 - r^2\big)\sum_{l=1}^{\infty}\delta_l P_l \rho^{-l} + 2m^{-1/2}\sum_{l=1}^{\infty}P_l^{1/2}\rho^{-l/2}\right)^{-1} \tag{3.25}$$

and leaves η_2 unchanged. We also let $\delta_l = d\rho^{l/2}$, with d to be determined later. This yields:

$$\eta_1 = \Big(1 + m^{-1}\big(1 - r^2\big)dM_1 + 2m^{-1/2}M_2\Big)^{-1}, \qquad \eta_2^{-1} = d^{-1}M_1$$

for $M_1 = \sum_{l=1}^{\infty}P_l\rho^{-l/2}$, and $M_2 = \sum_{l=1}^{\infty}P_l^{1/2}\rho^{-l/2}$. Now we set $d = am^{1/2}\big(1 - r^2\big)^{-1/2}$, for a to be determined later. We make this choice so that that η_1 and η_2 are both $1 + \mathcal{O}\big(m^{q-1/2}\big)$ for large m, which optimizes the asymptotic rate at which η_1 converges to 1. Recall that the moments M_1 and M_2 vary with m, and satisfy $M_1, M_2 = \mathcal{O}(m^q)$ for $0 \leq q < 1/2$. This yields:

$$\begin{aligned}
\eta_1 &= \Big(1 + am^{-1/2}\big(1 - r^2\big)^{1/2}M_1 + 2m^{-1/2}M_2\Big)^{-1} \\
&= 1 - m^{-1/2}\Big(a\big(1 - r^2\big)^{1/2}M_1 + 2M_2\Big) + \mathcal{O}\big(m^{2q-1}\big)
\end{aligned}$$

$$\eta_2 = a^{-1}m^{-1/2}\left(1-r^2\right)^{1/2}M_1$$

It is clear that M_1, M_2 and c_i match the formulas given in the Theorem 2 (see eq. (2.22), and eq. (3.1)). We now use Lemma 14 with $t_1 = \eta_1$, $t_2 = \eta_2$, $a = b = q - \frac{1}{2}$, etc.

$$R(\eta_1,\eta_2) = 1 - \frac{1}{m}\left(1-r^2\right)\left(1 - m^{-1/2}\left(\left(a+a^{-1}\right)\left(1-r^2\right)^{1/2}M_1 + 2M_2\right) + o\left(m^{-1/2+q}\right)\right)$$

Clearly to minimize the lowest order asynchronicity penalty, we should let $a = 1$. With this choice, we have proven Equation (3.4). Then using this in conjunction with Lemma 15 completes the proof of Theorem 2. □

4 Deterministic Unbounded Delays

We now present a convergence result for deterministic unbounded delays.

Assumption 3. Deterministic unbounded delays. The sequence of delay vectors $\vec{j}(0), \vec{j}(1), \vec{j}(2), \ldots$ is an arbitrary sequence in $\mathbb{N}^m$.

If there can be no assumption made on the distribution of the delays, we a have a slightly weaker result. Whereas before it was possible to use a constant step size, here the step size and convergence rate at step k depend on the current delay $j(k)$. Results that assume bounded delay τ need to know τ in advance to set the step size. If τ is very large, this will decrease the allowable timestep, which will slow convergence – even if a delay of τ is very rare. Here we make no such assumption and the step size and convergence rate are adaptive to whatever delay conditions exist in the system. The step size η^k is a function of the current delay $j(k)$, that must be measured (or upper bounded) at each iteration k.

The delay is allowed to be unbounded. The larger the delay, the less progress is made, but *some progress is always made at every step.* We define a **good behavior boundary**

$$T(m) = bm^q + d \tag{4.1}$$

for $0 \leq q < 1/2$, for arbitrary[12] parameters $b, d > 0$. Any delay less than this will not create a noticeable penalty in the progress of an algorithm in a large system (i.e. as $m \to \infty$) as compared to synchronous ARock. However if the delay is much larger than $T(m)$, the progress can eventually become vanishingly small. $T(m)$ can become very large since it grows as $\Theta(m^q)$.

We let ρ again be defined as in Equation (2.21). For arbitrary[13] $c > 0$, let

$$\gamma = \rho - cm^{-q} \tag{4.2}$$

We define the coefficients in the Lyapunov function from Equation (1.11):

$$c_i = m^{1/2}\left(1 + 2\left(1-r^2\right)\right)\frac{(\gamma/\rho)^i}{1-\gamma/\rho} \tag{4.3}$$

[12] These parameters are arbitrary, but setting them involves a trade-off as we will later see. The larger b is, the larger the asynchronicity penalty, and larger m has to be to ensure negligible penalty as will be seen in Theorem 3.

[13] Again, c can be set freely, but larger c is, the more the convergence rate is penalized for exceeding $T(m)$. However the smaller c is, the larger the asynchronicity penalty will be, and the large m needs to be to ensure negligible penalty (much like b).

We also define the following step size functions:

$$H_1(j) = \left(1 + c^{-1}m^{q-1/2}\left(3 + \gamma^{-j}\right)\right)^{-1}, \qquad H_2(j) = 2cm^{\frac{1}{2}-q}\gamma^j \tag{4.4}$$

These functions play similar roles to η_1 and η_2 from Theorem 2. We again use the same convergence-rate function R as defined in Equation (3.3). Let η^k be $\mathcal{F}^k$ measurable.

Theorem 3. *Let Assumption 1 and Assumption 3 hold. Consider the Lyapunov function defined in Equation (1.11) with coefficients given by Equation (4.3). Let $\eta^k \leq H_1(j)$. Then we have:*

$$\mathbb{E}\left[\xi^{k+1}\middle|\mathcal{G}^k\right] \leq R\left(\eta^k, H_2(j(k))\right)\xi^k \tag{4.5}$$

Additionally, let $\eta^k = H_1(j(k))$. If we have $j(k) \leq T(m) = bm^q + d$ for some $q \in [0, \frac{1}{2})$ and $b, d > 0$, then the convergence rate satisfies:

$$R \leq 1 - \frac{1}{m}(1-r^2)\left(1 - \frac{1}{c}m^{q-1/2}\left(3 + \frac{3}{2}\exp(bc)\right) + o\left(m^{q-\frac{1}{2}}\right)\right)$$

which corresponds to iteration complexity:

$$I(\epsilon) = \left(1 + \frac{1}{c}m^{q-1/2}\left(3 + \frac{3}{2}\exp(bc)\right) + o\left(m^{q-\frac{1}{2}}\right)\right)\left(\frac{1}{1-r^2}\right)\ln(1/\epsilon)$$

Remark 8. Much like in the stochastic delay case (see Remark 7), we prove a more general result where the Lyapunov function, and step size depend on a series of parameters $\epsilon_1, \epsilon_2, \ldots > 0$ and $\delta_1, \delta_2, \ldots > 0$. We make a sensible choice of these parameters to make the result more simple and interpretable.

4.1 Starting point

The starting point is Equation (3.16). However in this result, we assume that η^k is actually $\mathcal{F}^k$ measurable (that is, it is now allowed to also depend on the delays $\left(\vec{j}(0), \vec{j}(1), \vec{j}(2), \ldots, \vec{j}(k)\right)$, whereas before it could only depend on the iterates $(x^0, x^1, x^2, \ldots)$). Recall the definitions of E_j and D_j given in eq. (3.10).

Lemma 16. *Let Assumption 1 and Assumption 3 hold. Let $\epsilon_1, \epsilon_2, \ldots > 0$ and $\delta_1, \delta_2, \ldots > 0$ be a sequence of parameters. Let η^k be $\mathcal{F}^k$-measurable. Define the step size functions:*

$$h_1(j) = \left(1 + \frac{c_1}{m} + E_j\right)^{-1}, \qquad h_2(j) = D_j^{-1}$$

Let $\eta^k \leq h_1(j(k))$. Then ARock yields the following inequality:

$$\mathbb{E}\left[\xi^{k+1}\middle|\mathcal{F}^k\right] \leq \left\|x^k\right\|^2 R\left(\eta^k, h_2(j(k))\right) + \frac{1}{m}\sum_{i=1}^{\infty}\left(\left(\epsilon_i + \left(1-r^2\right)\delta_i\right) + c_{i+1}\right)\left\|x^{k+1-i} - x^{k-i}\right\|^2$$

The step size function h_1 and h_2 are complex expressions. When we eventually set the free parameters $\epsilon_1, \epsilon_2, \ldots > 0$ and $\delta_1, \delta_2, \ldots > 0$, we simplify these functions with inequalities to yield H_1 and H_2, which are much easier to interpret.

Proof. From Equation (3.16) we have:

$$\begin{aligned}\mathbb{E}\big[\xi^{k+1}\big|\mathcal{F}^k\big] &\leq \big\|x^k\big\|^2\Big(1-\frac{\eta^k}{m}\big(1-r^2\big)\big(1-\eta^k D_{j(k)}\big)\Big)-\frac{\eta^k}{m}\big\|S\hat{x}^k\big\|^2\Big(1-\eta^k\Big(1+\frac{c_1}{m}+E_{j(k)}\Big)\Big)\\ &+\frac{1}{m}\sum_{i=1}^{j(k)}\big(\epsilon_i+\big(1-r^2\big)\delta_i\big)\big\|x^{k+1-i}-x^{k-i}\big\|^2+\frac{1}{m}\sum_{i=1}^{\infty}c_{i+1}\big\|x^{k+1-i}-x^{k-i}\big\|^2\\ &\leq \big\|x^k\big\|^2 R\big(\eta^k,h_2(j(k))\big)-\frac{\eta^k}{m}\big\|S\hat{x}^k\big\|^2\big(1-\eta^k/h_1(j(k))\big)\\ &+\frac{1}{m}\sum_{i=1}^{\infty}\big(\big(\epsilon_i+\big(1-r^2\big)\delta_i\big)+c_{i+1}\big)\big\|x^{k+1-i}-x^{k-i}\big\|^2\end{aligned}$$

Setting $\eta^k \leq h_1(j(k))$ eliminates the $\big\|S\hat{x}^k\big\|^2$ term, and yields the desired result. □

4.2 Linear convergence

Proposition 17. Linear convergence for deterministic delays. *Let the conditions of Lemma 16 hold. Also assume:*

$$\sum_{i=1}^{\infty}\big(\epsilon_i+\big(1-r^2\big)\delta_i\big)\rho^{-i}<\infty \tag{4.6}$$

Let the coefficients of the Lyapunov function be given by:

$$c_i=\sum_{l=i}^{\infty}\big(\epsilon_i+\big(1-r^2\big)\delta_i\big)\rho^{l-i+1} \tag{4.7}$$

Then we have the following linear convergence rate at step k:

$$\mathbb{E}\big[\xi^{k+1}\big|\mathcal{F}^k\big]\leq R\big(\eta^k,h_2(j(k))\big)\xi^k \tag{4.8}$$

Proof. We apply Lemma 12 to Lemma 16 with $s_i=\epsilon_i+\big(1-r^2\big)\delta_i$, and we obtain:

$$\epsilon_i+\big(1-r^2\big)\delta_i+c_{i+1}\leq\rho c_i \tag{4.9}$$

Hence we have:

$$\begin{aligned}\mathbb{E}\big[\xi^{k+1}\big|\mathcal{F}^k\big] &\leq \big\|x^k\big\|^2 R\big(\eta^k,h_2(j(k))\big)+\frac{1}{m}\sum_{i=1}^{\infty}\big(\big(\epsilon_i+\big(1-r^2\big)\delta_i\big)+c_{i+1}\big)\big\|x^{k+1-i}-x^{k-i}\big\|^2 &(4.10)\\ &\leq \big\|x^k\big\|^2 R\big(\eta^k,h_2(j(k))\big)+\frac{1}{m}\sum_{i=1}^{\infty}\rho c_i\big\|x^{k+1-i}-x^{k-i}\big\|^2 &(4.11)\\ &\leq \max\big\{R\big(\eta^k,h_2(j(k))\big),\rho\big\}\xi^k &(4.12)\\ &= R\big(\eta^k,h_2(j(k))\big)\xi^k &(4.13)\end{aligned}$$

The last line follows, because $0\leq\eta^k\leq h_1(j)\leq 1$ implies $\rho\leq R(\eta^k,\eta_2)$. □

4.3 Proof of Theorem 3

Proof of Theorem 3. For the first part of the theorem, we simply set

$$\epsilon_i = m^{1/2}\gamma^i$$
$$\delta_i = 2m^{1/2}\gamma^i$$

This choice automatically satisfies that conditions of Proposition 17 (i.e. Equation (4.6)), and the coefficient formula that arises matches Equation (4.3).

We have step size functions h_1 and h_2 that are given by complicated expressions. Notice that we may overestimate h_1 and underestimate h_2, and the linear convergence result still holds. Hence we will simplify these step size expressions to give a more concise step size rule. We have:

$$\sum_{i=1}^{j}\gamma^{-j} \leq \gamma^{-j}\sum_{i=0}^{\infty}\gamma^i = \gamma^{-j}\frac{1}{1-\gamma} \leq c^{-1}m^q\gamma^{-j}$$

which yields:

$$(h_2(j))^{-1} = \sum_{i=1}^{j}\frac{1}{\delta_i} \leq \frac{1}{2c}m^{q-\frac{1}{2}}\gamma^{-j} = H_2(j)$$

which matches Equation (4.4). We also have:

$$c_1 = \sum_{l=1}^{\infty}\big(\epsilon_l + \big(1-r^2\big)\delta_l\big)\rho^{-l} = m^{1/2}\big(1+2\big(1-r^2\big)\big)\sum_{l=1}^{\infty}(\gamma/\rho)^l = m^{1/2}\big(1+2\big(1-r^2\big)\big)\frac{\gamma}{\rho-\gamma} \leq 3c^{-1}m^{1/2+q}$$

And hence:

$$h_1(j) = \left(1+\frac{c_1}{m}+\sum_{i=1}^{j}\frac{1}{\epsilon_i}\right)^{-1} \geq \left(1+\frac{3}{c}m^{q-1/2}+c^{-1}m^{q-1/2}\gamma^{-j}\right)^{-1} = \left(1+\frac{1}{c}m^{q-1/2}\big(3+\gamma^{-j}\big)\right)^{-1} = H_1(j)$$

which matches Equation (4.4). Hence Equation (4.5) is proven.

For the second part of the theorem, we need asymptotic expressions for $H_1(T)$ and $H_2(T)$ to determine an asymptotic convergence rate and iteration complexity. We first bound γ^{-T}

$$\gamma^{-T} = (1-(1-\gamma))^{-T} = \exp\Big(T(1-\gamma)+\mathcal{O}\Big(T(1-\gamma)^2\Big)\Big) = \exp\big(bc+\mathcal{O}\big(m^{-q}\big)\big) = \exp(bc)+\mathcal{O}\big(m^{-q}\big)$$

Hence this yields:

$$(h_2(T))^{-1} \leq \frac{1}{2c}m^{q-\frac{1}{2}}\big(\exp(bc)+\mathcal{O}\big(m^{-q}\big)\big)$$
$$h_1(T) \geq \left(1+\frac{1}{c}m^{q-1/2}\big(3+\exp(bc)+\mathcal{O}\big(m^{-q}\big)\big)\right)^{-1} = 1-\frac{1}{c}m^{q-1/2}(3+\exp(bc))+o\Big(m^{q-\frac{1}{2}}\Big)$$

Therefore, if $j(k) \leq T$, we have convergence rate:

$$R(H_1(j(k)), h_2(j(k))) \leq R(H_1(j(T)), H_2(j(T)))$$

$$= 1 - \frac{1}{m}(1-r^2)\left(1 - \frac{1}{c}m^{q-\frac{1}{2}}\left(3 + \frac{3}{2}\exp(bc)\right) + o\left(m^{q-\frac{1}{2}}\right)\right)$$
$$= 1 - \frac{1}{m}(1-r^2)\left(1 - \mathcal{O}\left(m^{q-\frac{1}{2}}\right)\right)$$

by Lemma 14. By Lemma 15 this corresponds to iteration complexity:

$$I(\epsilon) = \left(1 + \frac{1}{c}m^{q-1/2}\left(3 + \frac{3}{2}\exp(bc)\right) + o\left(m^{q-\frac{1}{2}}\right)\right)\left(\frac{1}{1-r^2}\right)\ln(1/\epsilon)$$
$$= \left(1 + \mathcal{O}\left(m^{q-\frac{1}{2}}\right)\right)\left(\frac{1}{1-r^2}\right)\ln(1/\epsilon)$$

which completes the proof. □

References

[1] Z. Peng, Y. Xu, M. Yan, and W. Yin, "ARock: An Algorithmic Framework for Asynchronous Parallel Coordinate Updates," *SIAM Journal on Scientific Computing*, vol. 38, no. 5, A2851–A2879, Jan. 1, 2016.

[2] R. Hannah and W. Yin, "On Unbounded Delays in Asynchronous Parallel Fixed-Point Algorithms," Sep. 15, 2016. arXiv: 1609.04746 [math].

[3] T. Sun, R. Hannah, and W. Yin, "Asynchronous Coordinate Descent under More Realistic Assumptions," May 19, 2017. arXiv: 1705.08494 [math].

[4] R. Leblond, F. Pedregosa, and S. Lacoste-Julien, "ASAGA: Asynchronous Parallel SAGA," in *PMLR*, Apr. 10, 2017, pp. 46–54.

[5] L. Cannelli, F. Facchinei, V. Kungurtsev, and G. Scutari, "Asynchronous Parallel Algorithms for Nonconvex Big-Data Optimization. Part II: Complexity and Numerical Results," Jan. 17, 2017. arXiv: 1701.04900 [math].

[6] R. Sun and Y. Ye, "Worst-case Complexity of Cyclic Coordinate Descent: $O(n2̂)$ Gap with Randomized Version," Apr. 25, 2016. arXiv: 1604.07130 [cs, math].

[7] Y. Chow, T. Wu, and W. Yin, "Cyclic Coordinate-Update Algorithms for Fixed-Point Problems: Analysis and Applications," *SIAM Journal on Scientific Computing*, A1280–A1300, Jan. 1, 2017.

[8] E. Serpedin and Q. M. Chaudhari, *Synchronization in Wireless Sensor Networks: Parameter Estimation, Peformance Benchmarks, and Protocols*, 1st. New York, NY, USA: Cambridge University Press, 2009.

[9] B. Eisenberg, "On the expectation of the maximum of IID geometric random variables," *Statistics & Probability Letters*, vol. 78, no. 2, pp. 135–143, Feb. 1, 2008.

[10] K. V. Mitov and E. Omey, "Renewal Processes," in *Renewal Processes*, ser. SpringerBriefs in Statistics, Springer International Publishing, 2014, pp. 1–51.

[11] O. Kella and W. Stadje, "Superposition of renewal processes and an application to multi-server queues," *Statistics & Probability Letters*, vol. 76, no. 17, pp. 1914–1924, Nov. 2006.

[12] A. B. Taylor, J. M. Hendrickx, and F. Glineur, "Exact worst-case convergence rates of the proximal gradient method for composite convex minimization," May 11, 2017. arXiv: 1705.04398 [math].

[13] B. Recht, C. Re, S. Wright, and F. Niu, "Hogwild!: A lock-free approach to parallelizing stochastic gradient descent," in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2011, pp. 693–701.

[14] H. Avron, A. Druinsky, and A. Gupta, "Revisiting asynchronous linear solvers: Provable convergence rate through randomization," in *Parallel and Distributed Processing Symposium, 2014 IEEE 28th International*, May 2014, pp. 198–207.

[15] J. Liu, S. J. Wright, C. Ré, V. Bittorf, and S. Sridhar, "An asynchronous parallel stochastic coordinate descent algorithm," *J. Mach. Learn. Res.*, vol. 16, no. 1, pp. 285–322, Jan. 2015.

[16] J. Liu and S. Wright, "Asynchronous stochastic coordinate descent: Parallelism and convergence properties," *SIAM Journal on Optimization*, vol. 25, no. 1, pp. 351–376, Jan. 1, 2015.

[17] Z. Peng, Y. Xu, M. Yan, and W. Yin, "On the Convergence of Asynchronous Parallel Iteration with Arbitrary Delays," Dec. 13, 2016. arXiv: 1612.04425 [cs, math, stat].

[18] H. Mania, X. Pan, D. Papailiopoulos, B. Recht, K. Ramchandran, and M. I. Jordan, "Perturbed iterate analysis for asynchronous stochastic optimization," Jul. 24, 2015. arXiv: 1507.06970 [cs, math, stat].

[19] X. Lian, H. Zhang, C.-J. Hsieh, Y. Huang, and J. Liu, "A Comprehensive Linear Speedup Analysis for Asynchronous Stochastic Parallel Optimization from Zeroth-Order to First-Order," Jun. 1, 2016. arXiv: 1606.00498 [math].

[20] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*. Springer Science & Business Media, Dec. 1, 2013, 253 pp.

A Auxillary Results

A.1 Block KM Iterations

Proof of Proposition 6. Taking conditional expectation on

$$\|x^{k+1}\|^2 = \|x^k\|^2 - 2\eta^k\langle x^k, P^k x^k\rangle + (\eta^k)^2\|P^k x^k\|^2$$

with respect to $i(k)$ yields

$$\begin{aligned}
\mathbb{E}\Big[\|x^{k+1}\|^2 | x^k\Big] &= \|x^k\|^2 - 2\eta^k\frac{p}{m}\langle x^k, Sx^k\rangle + (\eta^k)^2\frac{p}{m}\|Sx^k\|^2 \\
\text{(By Lemma 8)} &\le \|x^k\|^2 - \eta^k\frac{p}{m}\Big(\|Sx^k\|^2 + (1-r^2)\|x^k\|^2\Big) + (\eta^k)^2\frac{p}{m}\|Sx^k\|^2 \\
&= \Big(1 - \eta^k\frac{p}{m}(1-r^2)\Big)\|x^k\|^2 - \eta^k\frac{p}{m}(1-\eta^k)\|Sx^k\|^2, \qquad \text{(A.1)}
\end{aligned}$$

Let $\eta^k \le 1$, and we follow on from (A.1):

$$\text{(By } (1-r)\text{-strong monotonicity of } S) \le \Big(1 - \eta^k\frac{p}{m}(1-r^2)\Big)\|x^k\|^2 - (1-r)^2\eta^k\frac{p}{m}(1-\eta^k)\|x^k\|^2$$

$$= \left(1 - \frac{p}{m} + \frac{p}{m}\left(1 - \eta^k(1-r)\right)^2\right)\left\|x^k\right\|^2.$$

Now let $\eta^k \geq 1$, and we follow on again from A.1.

$$\begin{aligned}
&\mathbb{E}\left[\left\|x^{k+1}\right\|^2 | x^k\right] \\
\text{(Since } S \text{ is } (1+r)\text{-Lipschitz)} &\leq \left(1 - \eta^k \frac{p}{m}\left(1-r^2\right)\right)\left\|x^k\right\|^2 - \eta^k \frac{p}{m}(1-\eta^k)(1+r)^2\left\|x^k\right\|^2 \\
&= \left(1 - \frac{p}{m} + \frac{p}{m}\left(1 - \eta^k(1+r)\right)^2\right)\left\|x^k\right\|^2.
\end{aligned}$$

It can be verified that every single inequality for $\eta^k \leq 1$ is an equality for $T = rI$, and every single inequality for $\eta^k \geq 1$ is an equality for $T = -rI$. Therefore the inequalities give a sharp rate of convergence. This rate is optimized when $\eta^k = 1$, and matches the rate given in Equation (2.21).

Let's now look at the corresponding iteration complexity:

$$\left(1 - \frac{p}{m}\left(1-r^2\right)\right)^{I(\epsilon)\frac{m}{p}} = \epsilon$$

$$I(\epsilon) = \frac{p}{m}\left(\frac{\ln(1/\epsilon)}{-\ln\left(1 - \frac{p}{m}(1-r^2)\right)}\right)$$

Note that:

$$1 - x \leq \frac{-x}{\ln(1-x)} \leq 1 - \frac{1}{2}x$$

and hence

$$I(\epsilon) = \frac{1}{1-r^2}\left(1 - \theta\frac{p}{m}\left(1-r^2\right)\right)\ln(1/\epsilon)$$

where $\theta \in \left[\frac{1}{2}, 1\right]$. This matches Equation (2.16), hence the proof is complete. □

A.2 Random-Subset Block Gradient Descent

Proof of Corollary 7. First we consider the operator $T = I - \frac{2}{L+\mu}\nabla f$:

$$\begin{aligned}
\|T(y) - T(x)\|^2 &= \|y-x\|^2 - \frac{4}{\mu+L}\langle \nabla f(y) - \nabla f(x), y - x\rangle + \left(\frac{2}{L+\mu}\right)^2 \|\nabla f(y) - \nabla f(x)\|^2 \\
\text{(By Theorem 2.1.12 of [20])} &\leq \|y-x\|^2 - \frac{4}{\mu+L}\left(\frac{\mu L}{\mu+L}\|x-y\|^2 + \frac{1}{\mu+L}\|\nabla f(y) - \nabla f(x)\|^2\right) \\
&\quad + \left(\frac{2}{L+\mu}\right)^2 \|\nabla f(y) - \nabla f(x)\|^2 \\
&= \|y-x\|^2\left(1 - \frac{4\mu L}{(\mu+L)^2}\right) = \|y-x\|^2\left(1 - \frac{4\kappa}{(1+\kappa)^2}\right) = \|y-x\|^2\left(1 - \frac{2}{\kappa+1}\right)^2
\end{aligned}$$

Hence T is $\left(1-\frac{2}{\kappa+1}\right)$-Lipschitz.

We now determine the linear convergence rate:

$$r = 1 - \frac{2}{\kappa+1}$$
$$1 - r^2 = \frac{4}{\kappa+1}\left(\frac{\kappa}{\kappa+1}\right) = \frac{4\kappa}{(\kappa+1)^2}$$

And hence using Proposition 6 we have:

$$R = 1 - \frac{p}{m}(1-r^2) = 1 - \frac{p}{m}\frac{4\kappa}{(\kappa+1)^2} = 1 - 4\frac{p}{m\kappa}(1+\mathcal{O}(1/\kappa))$$

which matches Equation (2.18). Next we determine the iteration complexity using Proposition 6 again:

$$I(\epsilon) = \left(\frac{1}{1-r^2} - \theta\frac{p}{m}\right)\ln(1/\epsilon) = \left(\frac{(\kappa+1)^2}{4\kappa} - \theta\frac{p}{m}\right)\ln(1/\epsilon) = \frac{1}{4}\left(\kappa + 2 + \frac{1}{\kappa} - 4\theta\frac{p}{m}\right)\ln(1/\epsilon)$$
$$= \frac{1}{4}(\kappa + \mathcal{O}(1))\ln(1/\epsilon)$$

This matches Equation (2.19).

Lastly, we prove that the convergence rate is sharp. Recall the proof of Proposition 6. Now consider the function $f(x) = \frac{1}{2}\mu\|x\|^2$. This leads to $T = I\left(1-\frac{2}{\kappa+1}\right)$, which is equal to the worst-case example for $\eta \leq 1$. Additionally, consider $f(x) = \frac{1}{2}L\|x\|^2$, which leads to $T = -I\left(1-\frac{2}{\kappa+1}\right)$. This is the worst-case example for $\eta \geq 1$. Therefore, since the worst case examples in the previous are attained by μ-strongly convex functions with L-Lipschitz gradients, we can see that the convergence rate for error $\left\|x^k - x^*\right\|$is sharp. □