LINEARLY CONVERGENT STOCHASTIC ALGORITHMS FOR OPTIMIZATION AND LINEAR SYSTEMS

by

Ji Liu

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN–MADISON

2014

Date of final oral examination: 06/02/2014

The dissertation is approved by the following members of the Final Oral Committee:
Stephen J. Wright, Professor, Computer Sciences
Michael Ferris, Professor, Computer Sciences
Xiaojin Zhu, Professor, Computer Sciences
Vikas Singh, Professor, Biostatisics
Ming Yuan, Professor, Statistics

I dedicate this thesis to my mother and my wife.

ACKNOWLEDGMENTS

First and foremost I wish to thank my advisor, professor Stephen J. Wright of Computer Sciences department at University of Wisconsin-Madison. I have worked with him since the summer three years ago. I really enjoyed every day of learning from him. In the past three years, I learned the correct way to do research: how to find valuable and meaningful research topics and how to tackle a challenging problem step by step, which will undoubtedly benefit to my following academic career. As an advisor, he has never pushed me or sent me any pressure but gave me plenty of freedom to choose research topics. As an native speaker, he has never complained my poor English but always patiently revised my papers word by word and taught me the elegant way to write the academic paper. He offered me tons of help in a silent way: He never told me that he helped me until I found it. It is my luck to meet such a descent and respectable person in my life. Thank you, Steve!

Next, I would like to thank my other thesis-committee members, Michael Ferris, Xiaojin Zhu, Vikas Singh, and Ming Yuan, for their insights, questions, and advice for my research. I would like to thank all my committee members for taking their time to help me improve this dissertation and adjusting their schedule to attend my final defense.

I would also like to extend my gratitude to several other faculty members, who have helped me greatly during my research journey, Eric Bach, Jeffrey Linderoth, Robert Nowak, Benjamin Recht, Christopher Ré, Stephen Robinson, and Sijian Wang. My special thank goes to Benjamin Recht, who has patiently given me a lot of valuable advice on my research work.

I am very blessed to have the support my friends have given me. I would first like to thank my friends, Ming Gao, Heng Guo, Guoliang Jin, Jiexing Li, Jie Liu, Lanyue Lu, Zhigeng Geng, Xiaoming Shi, Linhai Song, Xi Wu, Wentao Wu, Jia Xu, Junming Xu, Luwan Zhang, and Yupu Zhang. I would also like to thank my officemates Eric Anderson, Okan Akalin, Aniruddha Bhargava, Victor Bittorf, James Foster, Jesse Holzer, Taedong Kim, Youngdae Kim, Cong Han Lim, Yanchao Liu, Jagdish Ramakrishnan, Nikhil Rao, Hongbo Dong, Herman Stampfli, Srikrishna Sridhar, Gongguo Tang, Lisa Tang, and Ce Zhang.

Finally, I would like to thank my mother and my wife, without whose unconditional love and support this Ph.D. would have been meaningless. Even though they have their own doubts and concerns, they have always had faith in me and supported me in pursuing my dream. They cheered with me even for my tiniest success and encouraged me when I felt low. The pride and joy in their eyes when seeing me graduate made it all worthwhile.

No words can express my gratitude and love to them. I dedicate this whole dissertation to these two most important women in my life.

— Ji Liu (2014)

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Stochastic first-order methods are fundamental methods that have shown renewed popularity in a variety of settings, especially machine learning. This dissertation considers several methods of this type, for optimization and linear algebra tasks, with a special focus on methods that are well suited to efficient implementation on modern multicore computers. Novel convergence results are presented for this setting, along with estimated on the number of cores that can be used in the computation without significant degradation of efficiency (the "potential parallelism"). Chapter 2 describes an asynchronous parallel stochastic coordinate descent method for convex smooth minimization. In the unconstrained case, sublinear convergence rate is proved for the weakly convex case, linear convergence for the essentially strongly convex case, and potential parallelism of $O(n^{1/2})$ for both cases where n is the problem dimension. For the case of bound constraints, similar complexity is proved, but with potential parallelism of $O(n^{1/4})$. Chapter 3 extends the parallel mechanism in Chapter 2 to solve the composite objective function consisting of a smooth function plus a separable nonsmooth function. In contrast to previous analysis in Chapter 2, the mode of asynchronous computation accounts for the "inconsistent" read situation. Despite the complications arising from "inconsistency", the method achieves a linear convergence rate on functions that satisfy an optimal strong convexity property and a sublinear rate on weakly convex functions with potential parallelism of $O(n^{1/4})$. Chapter 4 considers the linear algebra problem $Ax = b$, under the assumption of feasibility. The randomized Kaczmarz algorithm is parallelized in the same asynchronous parallel mechanism as Chapter 2. We prove a linear convergence rate and show that the potential parallelism is $O(m)$, where m is the number of rows in A. Chapter 5 considers the same problem as Chapter 4 - $Ax = b$ - but discusses an acceleration of the Kaczmarz approach (for serial implementation) using Nesterov-type acceleration scheme. In all chapters, computational experience is reported to back up our claims concerning convergence rate and parallel efficiency.

1 INTRODUCTION AND BACKGROUND

Stochastic first-order methods are fundamental methods that have shown renewed popularity in a variety of settings, especially machine learning. This dissertation considers several methods of this type, for optimization and linear algebra tasks. Specifically, in this dissertation we are interested in minimizing a smooth convex optimization function with a separable constraint or a separable nonsmooth convex regularizer, and finding a feasible point for a linear system $Ax = b$. We particularly focus on methods that are well suited to efficient implementation on modern multicore computers. Stochastic coordinate descent (SCD) is a computation and memory efficient approach to solve convex optimization problem. We apply the asynchronous parallel scheme used in Niu et al. (2011) to speedup the method. For the linear system problem, randomized Kaczmarz algorithm (RK) owns similar advantages to SCD. The same asynchronous parallel scheme is applied to speedup the approach. Novel convergence results are presented for this setting, along with estimated on the number of cores that can be used in the computation without significant degradation of efficiency (the "potential parallelism").

Chapter 2 describes an asynchronous parallel stochastic coordinate descent method (AsySCD) for convex smooth minimization. While parallelization does not essentially improve the convergence rate of stochastic coordinate descent method, it shortens the overall computational time because multiple processors share the computational cost. The traditional parallelization strategy follows the optimization steps as if the algorithm were running on a single processor and then distributes the workload at each iteration to multiple processors. In practice, however, the synchronization is quite expensive especially when the number of processors are substantial. It turns out that linear speedup is achievable only up to a small number of processors. This chapter considers a new asynchronous (communication and lock free) parallelization strategy that is proposed recently in Niu et al. (2011): all processors share the same memory storing the current optimization variable, run the same algorithm, and update the current variable simultaneously. We focus our discussion on the analysis of convergence and speedup properties for AsySCD. In the unconstrained case, sublinear convergence with rate $1/K$ is proved for the weakly convex case, linear convergence for the essentially strongly convex (weaker than strongly convex) case, and potential parallelism of $O(n^{1/2})$ where n is the dimension of the optimization variable. For the case of bound constraints, similar complexity is proved, but with potential parallelism of $O(n^{1/4})$.

Chapter 3 extends AsySCD to minimize a composite objective, consisting of a convex smooth function plus a separable convex nonsmooth function. In contrast to the analysis in Chapter 2 with assuming the "consistent read" model, this chapter plays with a more realistic situation: components of the unknown optimization vector may be written by some cores simultaneously with being read by others (namely "inconsistent read"). Despite the complications arising from the possibility, the method achieves a linear convergence rate for optimal strongly convex functions and a sublinear

rate on general convex functions, with potential parallelism of $O(n^{1/4})$. This result is consistent with the analysis for constrained AsySCD in Chapter 2.

Chapter 4 considers to solve a linear algebra problem $Ax = b$ on multicore computers. We apply the same asynchronous parallelization strategy used in Chapters 2 and 3 to parallelizing the randomized Kaczmarz algorithm (RK is the equivalent to applying stochastic gradient method to the least-squares problem $\min_x \frac{1}{2}\|Ax - b\|^2$). We prove the linear convergence for the asynchronous parallel randomize Kaczmarz (AsyRK) algorithm and also show that the potential parallelism is $O(m)$ where m is the number of rows in A.

Chapter 5 considers the same feasibility problem as Chapter 4, but in the context of a single core and a different speedup scheme — Nesterov's accelerated scheme. We propose an accelerated randomized Kaczmarz (ARK) algorithm: apply Nesterov's accelerated scheme to the RK algorithm. Although Nesterov's accelerated scheme cannot improve the convergence rate of the stochastic gradient method generally, by taking the advantage of the special structure in this feasibility problem, this chapter proves the linear convergence rate for ARK which is better than the standard RK algorithm on ill conditioned problems. The per-iteration cost of RK and ARK are similar if A is dense, but RK is much more able to exploit sparsity in A than is ARK. To deal with the sparse case, an efficient implementation for ARK, called SARK, is proposed. A comparison of convergence rates and average per-iteration complexities among RK, ARK, and SARK is provided, taking into account different levels of sparseness and conditioning. Comparisons with the leading deterministic algorithm — conjugate gradient applied to the normal equations — are also provided. Finally, the analysis is validated via computational testing.

In all chapters, computational experience is reported to back up our claims concerning convergence rate and parallel efficiency.

We review coordinate descent algorithm, Kaczmarz algorithm, and parallel optimization in the following.

1.1 (Stochastic) Coordinate Descent Algorithm

This section reviews some related work on coordinate relaxation and stochastic gradient algorithms.

Consider to solve a convex smooth optimization problem

$$\min_{x \in \mathbb{R}^n} \quad f(x). \tag{1.1}$$

The coordinate descent algorithm selects a coordinate of x in $\{1, 2, c \ldots, n\}$ to decrease the objective function $f(x)$ iteratively. Specifically, it runs the following step iteratively:

$$x_{k+1} = x_k - \gamma_k \nabla_i f(x_k) e_i \tag{1.2}$$

where i is the selected coordinate at iteration k, e_i is the ith natural basis, x_k denotes the value of x at iteration k, and $\gamma_k(>0)$ is the selected steplength satisfying $f(x_{k+1}) \leqslant f(x_k)$. Note that x_{k+1} only updates the ith coordinate of x_k. There are two manners to choose i per iteration: cyclic and stochastic. The cyclic manner updates all coordinates $\{1, \cdots, n\}$ in a deterministic order, while the stochastic manner randomly selects a coordinate to update per iteration. (Stochastic) coordinate descent can be extended to solve problem (1.1) with an additional separable constraint or separable nonsmooth convex regularizer by slightly modifying the step in (1.2).

Among *cyclic coordinate descent* algorithms, Tseng (2001) proved the convergence of a block coordinate descent method for nonsmooth functions with certain conditions. Local and global linear convergence were established under additional assumptions, by Luo and Tseng (1992) and Wang and Lin (2013), respectively. Global linear (sublinear) convergence rate for strongly (weakly) convex optimization was proved in (Beck and Tetruashvili, 2013). Block-coordinate approaches based on proximal-linear subproblems are described in Tseng and Yun (2009, 2010). Wright (2012) uses acceleration on reduced spaces (corresponding to the optimal manifold) to improve the local convergence properties of this approach.

Stochastic coordinate descent is almost identical to cyclic coordinate descent except selecting coordinates in a random manner. Nesterov (2012) studied the convergence rate for a stochastic block coordinate descent method for unconstrained and separably constrained convex smooth optimization, proving linear convergence for the strongly convex case and a sublinear $1/K$ rate for the convex case. Extensions to minimization of composite functions are described by Richtárik and Takáč (2011) and Lu and Xiao (2013). *Stochastic coordinate descent* can be viewed as a special case of stochastic gradient, so analysis of the latter approach can be applied, to obtain for example a sublinear $1/k$ rate of convergence in expectation for strongly convex functions; see, for example Nemirovski et al. (2009). However, stochastic coordinate descent is "special" in that it is possible to guarantee improvement in the objective at every step. Nesterov (2012) studied the convergence rate for a stochastic block coordinate descent method for unconstrained and separably constrained convex smooth optimization, proving linear convergence for the strongly convex case and a sublinear $1/k$ rate for the convex case. Richtárik and Takáč (2011) and Lu and Xiao (2013) extended this work to composite minimization, in which the objective is the sum of a smooth convex function and a separable nonsmooth convex function, and obtained similar (slightly stronger) convergence results. Stochastic coordinate descent is extended by Necoara and Patrascu (2013) to convex optimization with a single linear constraint, randomly updating *two* coordinates at a time to maintain feasibility.

1.2 (Randomized) Kaczmarz Algorithm

This section reviews some related work on (randomized) Kaczmarz algorithm.

Consider the problem of finding a solution to a consistent linear system

$$Ax = b,$$

where A has m rows $\{a_1, \cdots, a_m\}$ and b is a vector in $\mathbb{R}^m$. The Kaczmarz algorithm selects a row of A and a corresponding component in b (say, a_i and b_i), and projects the current x onto the hyperplane defined by $a_i^T x = b_i$ per iteration:

$$x_{k+1} = x_k - \frac{a_i^T x_k - b_i}{\|a_i\|^2} a_i. \tag{1.3}$$

Similar to the coordinate descent algorithm, there are two manners to select i per iteration: cyclic and stochastic (or randomized).

The original Kaczmarz algorithm (Kaczmarz, 1937) used a cyclic projection procedure to solve consistent linear systems $Ax = b$. Kaczmarz proved convergence to the unique solution when A is a square nonsingular matrix. The cyclic ordering of the iterates made it difficult to obtain iteration-based convergence results, but Galantai (2005) proved a linear convergence rate in terms of cycles. Since the 1980s, the Kaczmarz algorithm has found an important application area in Algebraic Reconstruction Techniques (ART) for image reconstruction; see for example Herman (1980, 2009). It is sometimes referred to in this literature as the "sequential row-action ART algorithm."

Strohmer and Vershynin (2009) studied the behavior of RK in the case of a consistent system $Ax = b$ in which A has full column rank (making the solution unique). They proved linear convergence rate for RK in expectation. Needell (2010) also assumed full column rank, but dropped the assumption of consistency, showing that the RK algorithm converges linearly to a ball of fixed radius centered at the solution, where the radius is proportional to the distance of b from the image space of A. Eldar and Needell (2011) presented a modified version of the randomized Kaczmarz method which selects the optimal projection from a randomly chosen set at each iteration. This technique improves the convergence rate, but requires more computation per iteration.

Leventhal and Lewis (2010) extended the RK algorithm for consistent linear equalities $Ax = b$ to the more general setting of consistent linear inequalities and equalities: $A_I x \geqslant b_I$, $A_E x = b_E$. The basic idea is quite similar to the RK algorithm: iteratively update x_{k+1} by projecting x_k onto the hyperplane or half space for a randomly selected equality or inequality constraint. The linear convergence rate was proven to be $1 - 1/(L^2\|A\|_F^2)$, where L is the Hoffman constant (Hoffman, 1952) for the full system.

Zouzias and Freris (2012) considered the case of possibly inconsistent (1.3). They proposed a randomized extended Kaczmarz algorithm by first projecting b orthogonally onto the image space of A to obtain $b_\perp$, then orthogonally projecting the initial point x_0 onto the hyperplane $Ax = b_\perp$. Essentially, the RK algorithm is applied twice. The convergence rate is proven to be $1 - \lambda_{\min}/\|A\|_F^2$, which is the same as the RK algorithm for consistent linear systems. This method can be considered

as a randomized variant of the extended Kaczmarz method proposed by Popa (1999).

1.3 Synchronous / Asynchronous Parallel Optimization Methods

We review parallel optimization approaches in the section, including general synchronous parallel methods, synchronous parallel coordinate descent, asynchronous parallel methods, and parallel methods for the Kaczmarz algorithm. In addition, the asynchronous parallel scheme used in Niu et al. (2011) (also in our work, see Chapters 2, 3, and 4) is detailed in the end.

Synchronous parallel methods distribute the workload and data among multiple processors, and coordinate the computation among processors. Ferris and Mangasarian (1994) proposed to distribute variables among multiple processors and optimize concurrently over each subset. The synchronization step searches the affine hull formed by the current iterate and the points found by each processor. Similar ideas appeared in (Mangasarian, 1995), with a different synchronization step. Goldfarb and Ma (2012) considered a multiple splitting algorithm for functions of the form $f(x) = \sum_{k=1}^{N} f_k(x)$ in which N models are optimized separately and concurrently, then combined in an synchronization step. The alternating direction method-of-multiplier (ADMM) framework (Boyd et al., 2011) can also be implemented in paralle. It dissects the problem into multiple subproblems (possibly after replication of primal variables) and optimizes concurrently, then synchronizes to update multiplier estimates. Duchi et al. (2012) described a subgradient dual-averaging algorithm for partially separable objectives, with subgradient evaluations distributed between cores and combined in ways that reflect the structure of the objective. Parallel stochastic gradient approaches have received broad attention; see Agarwal and Duchi (2012) for an approach that allows delays between evaluation and update, and (Cotter et al., 2011) for a minibatch stochastic gradient approach with Nesterov acceleration. Shalev-Shwartz and Zhang (2013) proposed an accelerated stochastic dual coordinate ascent method.

In the class of *synchronous parallel methods* for coordinate descent, Richtárik and Takáč (2012) studied a synchronized parallel block (or minibatch) coordinate descent algorithm for composite optimization problems of the form (3.1), with a block separable regularizer g. At each iteration, processors update the randomly selected coordinates concurrently and synchronously. Speedup depends on the sparsity of the data matrix that defines the loss functions. A similar synchronous parallel method was studied in Necoara and Clipici (2013) and Bradley et al. (2011); the latter focuses on the case of $g(x) = \|x\|_1$. Scherrer et al. (2012) make greedy choices of multiple blocks of variables to update in parallel. Another greedy way of selecting coordinates was considered by Peng et al. (2013), who also describe a parallel implementation of FISTA, an accelerated first-order algorithm due to Beck and Teboulle (2009). Fercoq and Richtárik (2013b) consider a variant of (3.1) in which f is allowed to be nonsmooth. They apply Nesterov's smoothing scheme to obtain a smoothed version and update multiple blocks of coordinates using block coordinate descent in parallel. Sublinear convergence rate is established for both strongly convex and weakly convex cases.

Fercoq and Richtárik (2013a) applies Nesterov's accelerated scheme to the synchronous parallel block coordinate algorithm of Richtárik and Takáč (2012), and provides an improved sublinear convergence rate, whose efficiency again depends on sparsity in the data matrix. Yang (2013) studied the parallel stochastic dual coordinate ascent method, emphasizing the balance between computation and communication.

We turn now to *asynchronous parallel methods*. Bertsekas and Tsitsiklis (1989) introduced an asynchronous parallel implementation for general fixed point problems $x = q(x)$ over a separable convex closed feasible region. (The optimization problem (2.1) (or (3.1)) considered in Chapter 2 (or Chapter 3) can be formulated in this way by defining $q(x) := \mathcal{P}_\Omega[(I - \gamma\nabla f)(x)]$ (or $q(x) := \mathcal{P}_{\gamma g}[(I - \gamma\nabla f)(x)]$) for some fixed $\gamma > 0$. See the definitions for $\mathcal{P}_\Omega(\cdot)$ and $\mathcal{P}_{\gamma g}(\cdot)$ therein.) Their analysis allows inconsistent reads for x, that is, the coordinates of the read x have different "ages." Linear convergence is established if all ages are bounded and $\nabla^2 f(x)$ satisfies a diagonal dominance condition guaranteeing that the iteration $x = q(x)$ is a maximum-norm contraction mapping for sufficient small γ. However, this condition is strong — stronger, in fact, than the strong convexity condition. For convex quadratic optimization $f(x) = \frac{1}{2}x^T Ax + bx$, the contraction condition requires diagonal dominance of the Hessian: $A_{ii} > \sum_{i \neq j} |A_{ij}|$ for all $i = 1, 2, \ldots, n$. Elsner et al. (1992) considered the same fixed point problem and architecture as Bertsekas and Tsitsiklis (1989), and describe a similar scheme. Their scheme appears to require locking of the shared-memory data structure for x to ensure consistent reading and writing. Frommer and Szyld (2000) give a comprehensive survey of asynchronous methods for solving fixed-point problems.

Hogwild! (Niu et al., 2011) is a lock-free, asynchronous parallel implementation of a stochastic-gradient method, targeted to a multicore computational model similar to the one considered here. Its analysis assumes consistent reading of x, and it is implemented without locking or coordination between processors. Under certain conditions, convergence of Hogwild! approximately matches the sublinear $1/K$ rate of its serial counterpart, which is the constant-steplength stochastic gradient method analyzed in Nemirovski et al. (2009). Chapters 2, 3, and 4 also adopt this asynchronous parallel scheme. We particularly detail this scheme in the end of this section.

We also note recent work by Avron et al. (2014), who proposed an asynchronous linear solver to solve $Ax = b$ where A is a symmetric positive definite matrix, proving a linear convergence rate. Both inconsistent- and consistent-read cases are analyzed in this chapter, with the convergence result for inconsistent read being slightly weaker.

Finally, we turn to *parallel approaches for Kaczmarz algorithm and its variants*. Among synchronous parallel methods, Censor et al. (2001) proposed a parallel component averaging method to solve (4.1). This approach parallel-projects the current x onto all (or multiple) hyperplanes, then applies an averaging scheme to the projections to obtain the next iterate. This method is essentially a gradient descent method for solving $\frac{1}{2}\|Ax - b\|^2$, so is able to handle inconsistent problems. This paper also notes (Censor et al., 2001, Section 5.1) that for sparse problems, parallelism can be obtained by simultaneously projecting the current iterate onto a set of mutually orthogonal

hyperplanes, obtained by considering equations whose nonzero components appear in disjoint locations. When obtained from image reconstruction problems, such sets of equations can be obtained by considering parallel rays that are sufficiently far apart so as to pass through disjoint sets of pixels. This type of parallelism has small granularity, and the amount of communication required between processors may make it unattractive in practice.

A related approach is Block-Cimmino (or Block-AMS) algorithm (Aharoni and Censor, 1989), which can be considered as the block version of Kaczmarz algorithm. Other variants of Block-Cimmino algorithm are described in Elfving and Nikazad (2009); Nikazad (2008).

Elsner et al. (1990) proposed an asynchronous parallel RK algorithm, again for a situation in which all processors have access to x stored in commonly accessible memory. Each processor iteratively runs the following procedures, where x denotes a globally shared version of the variable vector and x' and x'' denote copies stored locally on each processor: (a) read the current global x into the local x'; (b) write the convex combination of the local variables x' and x'' into x'', and also into the shared memory as a new x; (c) project the local x'' onto the selected hyperplane to get a new x''. Note that the algorithm requires locking the shared memory in step (b) because it does not allow two processors to access shared memory at the same time. Our computational experiences with related algorithms (e.g., AsySCD (Liu et al., 2013) (also see Chapter 2) and Hogwild! (Niu et al., 2011)) indicate that memory locking of this type degrades computational performance seriously. Moreover, the convergence analysis establishes convergence, but does not prove a linear convergence rate.

Asynchronous Parallel Scheme in Niu et al. (2011)

This section generalizes the asynchronous scheme used in Niu et al. (2011) (also in Chapters 2, 3, and 4). It assumes a centralized architecture / network shown in Figure 1.1. All cores can access the same memory and work in the following rules:

- All processors / cores share the same memory which saves the current optimization variable x;
- All processors / cores run the same optimization algorithm independently;
- All processors / cores update x concurrently *without* any software locking.

This procedure totally avoids the synchronization cost and even allows data distributed in different locations (cores). Particularly, in our case all cores *simultaneously* run a SCD process for minimizing a convex function (see Chapters 2 and 3) and a RK process for finding a feasible point to $Ax = b$ (see Chapter 4), updating x in an asynchronous fashion.

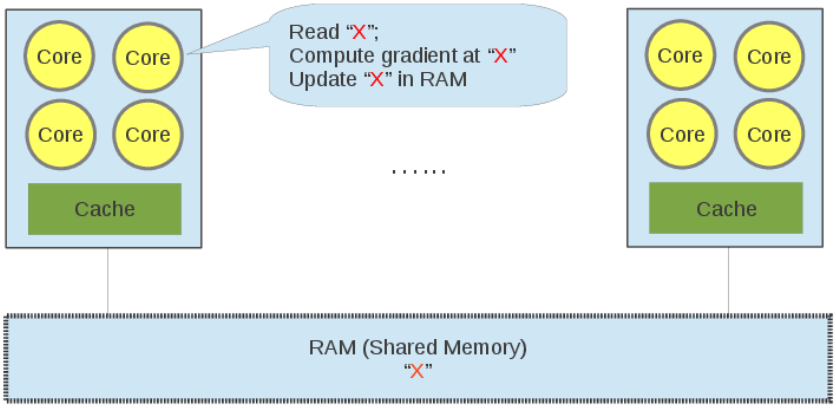

Figure 1.1: How does the *asynchronous* parallel procedure work?

2 AN ASYNCHRONOUS PARALLEL STOCHASTIC COORDINATE DESCENT ALGORITHM FOR CONSTRAINED / UNCONSTRAINED FUNCTIONS

2.1 Overview

We describe an asynchronous parallel stochastic coordinate descent algorithm for minimizing smooth unconstrained or separably constrained functions. The method achieves a linear convergence rate on functions that satisfy an essential strong convexity property and a sublinear rate ($1/K$) on general convex functions. Near-linear speedup on a multicore system can be expected if the number of processors is $O(n^{1/2})$ in unconstrained optimization and $O(n^{1/4})$ in the separable-constrained case, where n is the number of variables. We describe results from implementation on 40-core processors. Please also refer to the original paper in Liu et al. (2013).

2.2 Introduction

Consider the convex optimization problem

$$\min_{x \in \Omega} \quad f(x), \tag{2.1}$$

where $\Omega \subset \mathbb{R}^n$ is a closed convex set and f is a smooth convex mapping from an open neighborhood of Ω to $\mathbb{R}$. We consider two particular cases of Ω in this chapter: the unconstrained case $\Omega = \mathbb{R}^n$, and the separable case

$$\Omega = \Omega_1 \times \Omega_2 \times \ldots \times \Omega_n, \tag{2.2}$$

where each Ω_i, $i = 1, 2, \ldots, n$ is a closed subinterval of the real line.

Formulations of the type (2.1,2.2) arise in many data analysis and machine learning problems, for example, support vector machines (linear or nonlinear dual formulation) (Cortes and Vapnik, 1995), LASSO (after decomposing x into positive and negative parts) (Tibshirani, 1996), and logistic regression. Algorithms based on gradient and approximate or partial gradient information have proved effective in these settings. We mention in particular gradient projection and its accelerated variants (Nesterov, 2004), accelerated proximal gradient methods for regularized objectives (Beck and Teboulle, 2009), and stochastic gradient methods (Nemirovski et al., 2009; Shamir and Zhang, 2013). These methods are inherently serial, in that each iteration depends on the result of the previous iteration. Recently, parallel multicore versions of stochastic gradient and stochastic coordinate descent have been described for problems involving large data sets; see for example (Niu et al., 2011; Richtárik and Takáč, 2012; Avron et al., 2014).

This chapter proposes an asynchronous stochastic coordinate descent (AsySCD) algorithm for convex optimization. Each step of AsySCD chooses an index $i \in \{1, 2, \ldots, n\}$ and subtracts a short,

constant, positive multiple of the partial gradient $\partial f/\partial x_i$ from component i of x. When separable constraints (2.2) are present, the update is "clipped" to maintain feasibility with respect to Ω_i. Updates take place in parallel across the cores of a multicore system.

We use a simple model of computation that matches well to modern multicore architectures. We assume that each core makes coordinate-descent modifications to a centrally stored vector x in an asynchronous, uncoordinated fashion. A consequence of this model is that the version of x that is read by a core in order to evaluate its gradient is usually not the same as the version to which the update is made later, as it is updated in the interim by other cores. We assume that there is a bound τ on the age of the updates, that is, no more than τ updates to x occur between the time at which a processor reads x (and uses it to evaluate one element of the gradient) and the time at which this processor makes its update to a single element of x. (A similar model of parallel asynchronous computation was used in Hogwild! (Niu et al., 2011).) Our implementation, described in Section 2.6, is a little more complex than this simple model would suggest, as it is tailored to the architecture of the Intel Xeon machine that we use for experiments.

We show that linear convergence can be attained if an "essential strong convexity" property (2.3) holds, while sublinear convergence at a "$1/K$" rate can be proved for general convex functions. Our analysis also defines a sufficient condition for near-linear speedup in the number of cores used. This condition relates the value of delay parameter τ (which relates to the number of cores / threads used in the computation) to the problem dimension n. A parameter that quantifies the cross-coordinate interactions in ∇f also appears in this relationship. When the Hessian of f is nearly diagonal, the minimization problem can almost be separated along the coordinate axes, so higher degrees of parallelism are possible.

Section 2.3 specifies the proposed algorithm. Convergence results for unconstrained and constrained cases are described in Sections 2.4 and 2.5, respectively, with proofs given in the appendix. Computational experience is reported in Section 2.6. We discuss several variants of AsySCD in Section 2.7. Some conclusions are given in Section 2.8.

Notation and Assumption

We use the following notation.

- $e_i \in \mathbb{R}^n$ denotes the ith natural basis vector.
- $\|\cdot\|$ denotes the Euclidean norm $\|\cdot\|_2$.
- $S \subset \Omega$ denotes the set on which f attains its optimal value, which is denoted by f^*.
- $\mathcal{P}_S(\cdot)$ and $\mathcal{P}_\Omega(\cdot)$ denote Euclidean projection onto S and Ω, respectively.
- We use x_i for the ith element of x, and $\nabla_i f(x)$, $(\nabla f(x))_i$, or $\partial f/\partial x_i$ for the ith element of $\nabla f(x)$.

- We define the following *essential strong convexity* condition for a convex function f with respect to the optimal set S, with parameter $l > 0$:

$$f(x) - f(y) \geqslant \langle \nabla f(y), x - y \rangle + \frac{l}{2}\|x - y\|^2$$
$$\forall\, x, y \in \Omega \text{ with } \mathcal{P}_S(x) = \mathcal{P}_S(y). \tag{2.3}$$

  This condition is significantly weaker than the usual strong convexity condition, which requires the inequality to hold for *all* $x, y \in \Omega$. In particular, it allows for non-singleton solution sets S, provided that f increases at a uniformly quadratic rate with distance from S. (This property is noted for convex quadratic f in which the Hessian is rank deficient.) Other examples of essentially strongly convex funcions that are not strongly convex include:

  - $f(Ax)$ with arbitrary linear transformation A, where $f(\cdot)$ is strongly convex;
  - $f(x) = \max(a^T x - b, 0)^2$, for $a \neq 0$.

- Define L_{res} as the *restricted Lipschitz constant* for ∇f, where the "restriction" is to the coordinate directions: We have

$$\|\nabla f(x) - \nabla f(x + te_i)\| \leqslant L_{\text{res}}|t|, \quad \text{for all } i = 1, 2, \ldots, n \text{ and } t \in \mathbb{R}, \text{ with } x, x + te_i \in \Omega.$$

- Define L_i as the *coordinate Lipschitz constant* for ∇f in the ith coordinate direction: We have

$$f(x + te_i) - f(x) \leqslant \langle \nabla_i f(x),\ t \rangle + \frac{L_i}{2}t^2, \quad \text{for } i \in \{1, 2, \ldots, n\}, \text{ and } x, x + te_i \in \Omega,$$

  or equivalently

$$|\nabla_i f(x) - \nabla_i f(x + te_i)| \leqslant L_i|t|.$$

- $L_{\max} := \max_{i=1,2,\ldots,n} L_i$. Note that $L_{\text{res}} \geqslant L_{\max}$.

- $R_0 := \|x_0 - \mathcal{P}_S(x_0)\|$, where x_0 is the initial point.

Lipschitz Constants

The nonstandard Lipschitz constants L_{res}, $L_{\max}$, and L_i, $i = 1, 2, \ldots, n$ defined above are crucial in the analysis of our method. Besides bounding the nonlinearity of f along various directions, these quantities capture the interactions between the various components in the gradient ∇f, as quantified in the off-diagonal terms of the Hessian $\nabla^2 f(x)$ (when this matrix exists).

We have noted already that $L_{\text{res}}/L_{\max} \geqslant 1$. Let us consider upper bounds on this ratio under certain conditions. When f is twice continuously differentiable, we have

$$L_i = \sup_{x \in \Omega} \max_{i=1,2,\ldots,n} [\nabla^2 f(x)]_{ii}.$$

Since $\nabla^2 f(x) \succeq 0$ for $x \in \Omega$, we have that

$$|[\nabla^2 f(x)]_{ij}| \leqslant \sqrt{L_i L_j} \leqslant L_{\max}, \quad \forall\, i, j = 1, 2, \ldots, n.$$

Thus L_{res}, which is a bound on the largest column norm for $\nabla^2 f(x)$ over all $x \in \Omega$, is bounded by $\sqrt{n} L_{\max}$, so that

$$\frac{L_{\text{res}}}{L_{\max}} \leqslant \sqrt{n}.$$

If the Hessian is structurally sparse, having at most p nonzeros per row/column, the same argument leads to $L_{\text{res}}/L_{\max} \leqslant \sqrt{p}$.

If $f(x)$ is a convex quadratic with Hessian Q, We have

$$L_{\max} = \max_i \, Q_{ii}, \quad L_{\text{res}} = \max_i \|Q_{\cdot i}\|_2,$$

where $Q_{\cdot i}$ denotes the ith column of Q. If Q is diagonally dominant, we have for any column i that

$$\|Q_{\cdot i}\|_2 \leqslant Q_{ii} + \|[Q_{ji}]_{j \neq i}\|_2 \leqslant Q_{ii} + \sum_{j \neq i} |Q_{ji}| \leqslant 2 Q_{ii},$$

which, by taking the maximum of both sides, implies that $L_{\text{res}}/L_{\max} \leqslant 2$ in this case.

Finally, consider the objective $f(x) = \frac{1}{2}\|Ax - b\|^2$ and assume that $A \in \mathbb{R}^{m \times n}$ is a random matrix whose entries are i.i.d from $\mathcal{N}(0, 1)$. The diagonals of the Hessian are $A_{\cdot i}^T A_{\cdot i}$ (where $A_{\cdot i}$ is the ith column of A), which have expected value m, so we can expect $L_{\max}$ to be not less than m. Recalling that L_{res} is the maximum column norm of $A^T A$, we have

$$\begin{aligned}
\mathbb{E}(\|A^T A_{\cdot i}\|) &\leqslant \mathbb{E}(|A_{\cdot i}^T A_{\cdot i}|) + \mathbb{E}(\|[A_{\cdot j}^T A_{\cdot i}]_{j \neq i}\|) \\
&= m + \mathbb{E} \sqrt{\sum_{j \neq i} |A_{\cdot j}^T A_{\cdot i}|^2} \\
&\leqslant m + \sqrt{\sum_{j \neq i} \mathbb{E} |A_{\cdot j}^T A_{\cdot i}|^2} \\
&= m + \sqrt{(n-1)m},
\end{aligned}$$

where the second inequality uses Jensen's inequality and the final equality uses

$$\begin{aligned}\mathbb{E}(|A_{\cdot j}^T A_{\cdot i}|^2) &= \mathbb{E}(A_{\cdot j}^T \mathbb{E}(A_{\cdot i} A_{\cdot i}^T) A_{\cdot j}) \\ &= \mathbb{E}(A_{\cdot j}^T I A_{\cdot j}) = \mathbb{E}(A_{\cdot j}^T A_{\cdot j}) = m.\end{aligned}$$

We can thus estimate the upper bound on L_{res}/L_{max} roughly by $1 + \sqrt{n/m}$ for this case.

2.3 Algorithm

In AsySCD, multiple processors have access to a shared data structure for the vector x, and each processor is able to compute a randomly chosen element of the gradient vector $\nabla f(x)$. Each processor repeatedly runs the following coordinate descent process (the steplength parameter γ is discussed further in the next section):

R: Choose an index $i \in \{1, 2, \ldots, n\}$ at random, read x, and evaluate $\nabla_i f(x)$;

U: Update component i of the shared x by taking a step of length γ/L_{max} in the direction $-\nabla_i f(x)$.

Algorithm 1 Asynchronous Stochastic Coordinate Descent Algorithm $x_{K+1} = \text{AsySCD}(x_0, \gamma, K)$

Require: $x_0 \in \Omega$, γ, and K
Ensure: x_{K+1}
1: Initialize $j \leftarrow 0$;
2: **while** $j \leqslant K$ **do**
3: Choose $i(j)$ from $\{1, \ldots, n\}$ with equal probability;
4: $x_{j+1} \leftarrow \mathcal{P}_\Omega \left(x_j - \frac{\gamma}{L_{max}} e_{i(j)} \nabla_{i(j)} f(x_{k(j)}) \right)$;
5: $j \leftarrow j + 1$;
6: **end while**

Since these processors are being run concurrently and without synchronization, x may change between the time at which it is read (in step R) and the time at which it is upatted (step U). We capture the system-wide behavior of AsySCD in Algorithm 1. There is a global counter j for the total number of updates; x_j denotes the state of x after j updates. The index $i(j) \in \{1, 2, \ldots, n\}$ denotes the component updated at step j. $k(j)$ denotes the x-iterate at which the update applied at iteration j was calculated. Obviously, we have $k(j) \leqslant j$, but we assume that the delay between the time of evaluation and updating is bounded uniformly by a positive integer τ, that is, $j - k(j) \leqslant \tau$ for all j. The value of τ captures the essential parallelism in the method, as it indicates the number of processors that are involved in the computation.

The projection operation P_Ω onto the feasible set is not needed in the case of unconstrained optimization. For separable constraints (2.2), it requires a simple clipping operation on the $i(j)$ component of x.

We note several differences with earlier asynchronous approaches. Unlike the asynchronous scheme in Bertsekas and Tsitsiklis (1989, Section 6.1), the *latest* value of x is updated at each step, not an earlier iterate. Although our model of computation is similar to Hogwild! (Niu et al., 2011), the algorithm differs in that each iteration of AsySCD evaluates a single component of the gradient exactly, while Hogwild! computes only a (usually crude) estimate of the full gradient. Our analysis of AsySCD below is comprehensively different from that of Niu et al. (2011), and we obtain stronger convergence results.

2.4 Unconstrained Smooth Convex Case

This section presents results about convergence of AsySCD in the unconstrained case $\Omega = \mathbb{R}^n$. The theorem encompasses both the linear rate for essentially strongly convex f and the sublinear rate for general convex f. The result depends strongly on the delay parameter τ. (Proofs of results in this section appear in Appendix.)

A crucial issue in AsySCD is the choice of steplength parameter γ. This choice involves a tradeoff: We would like γ to be long enough that significant progress is made at each step, but not so long that the gradient information computed at step $k(j)$ is stale and irrelevant by the time the update is applied at step j. We enforce this tradeoff by means of a bound on the ratio of expected squared norms on ∇f at successive iterates; specifically,

$$\rho^{-1} \leqslant \frac{\mathbb{E}\|\nabla f(x_{j+1})\|^2}{\mathbb{E}\|\nabla f(x_j)\|^2} \leqslant \rho, \tag{2.4}$$

where $\rho > 1$ is a user defined parameter. The analysis becomes a delicate balancing act in the choice of ρ and steplength γ between aggression and excessive conservatism. We find, however, that these values can be chosen to ensure steady convergence for the asynchronous method at a *linear* rate, with rate constants that are almost consistent with vanilla short-step full-gradient descent.

We use the following assumption in some of the results of this section.

Assumption 2.1. *There is a real number R such that*

$$\|x_j - \mathcal{P}_S(x_j)\| \leqslant R, \quad \text{for all } j = 0, 1, 2, \ldots.$$

Theorem 2.1. *Suppose that $\Omega = \mathbb{R}^n$ in (2.1) and that S is nonempty. For any $\rho > 1$, define the quantity ψ as follows:*

$$\psi := 1 + \frac{2\tau\rho^\tau L_{\text{res}}}{\sqrt{n} L_{\max}}. \tag{2.5}$$

Suppose that the steplength parameter $\gamma > 0$ satisfies the following three upper bounds:

$$\gamma \leqslant \frac{1}{\psi}, \tag{2.6a}$$

$$\gamma \leqslant \frac{(\rho-1)\sqrt{n}L_{\max}}{2\rho^{\tau+1}L_{\mathrm{res}}}, \tag{2.6b}$$

$$\gamma \leqslant \frac{(\rho-1)\sqrt{n}L_{\max}}{L_{\mathrm{res}}\rho^{\tau}(2+\frac{L_{\mathrm{res}}}{\sqrt{n}L_{\max}})}. \tag{2.6c}$$

Then we have that for any $j \geqslant 0$ that

$$\rho^{-1}\mathbb{E}(\|\nabla f(x_j)\|^2) \leqslant \mathbb{E}(\|\nabla f(x_{j+1})\|^2) \leqslant \rho\mathbb{E}(\|\nabla f(x_j)\|^2). \tag{2.7}$$

Moreover, if the essentially strong convexity property (2.3) holds with $l > 0$, we have

$$\mathbb{E}(f(x_j) - f^*) \leqslant \left(1 - \frac{2l\gamma}{nL_{\max}}\left(1 - \frac{\psi}{2}\gamma\right)\right)^j (f(x_0) - f^*). \tag{2.8}$$

For general smooth convex functions f, if, in addition, Assumption 2.1 is satisfied, we have

$$\mathbb{E}(f(x_j) - f^*) \leqslant \frac{1}{(f(x_0) - f^*)^{-1} + \frac{\gamma}{nL_{\max}R^2}(1 - \frac{\psi}{2}\gamma)j}. \tag{2.9}$$

This theorem demonstrates linear convergence (2.8) for AsySCD in the unconstrained essentially strongly convex case. This result is better than that obtained for Hogwild! (Niu et al., 2011), which guarantees only sublinear convergence under the stronger assumption of strict convexity.

The following corollary proposes an interesting particular choice of the parameters for which the convergence expressions become more comprehensible. The result requires a condition on the delay bound τ in terms of n and the ratio $L_{\max}/L_{\mathrm{res}}$.

Corollary 2.2. *Suppose that S is nonempty, and that*

$$\tau + 1 \leqslant \frac{\sqrt{n}L_{\max}}{2eL_{\mathrm{res}}}. \tag{2.10}$$

Then if we choose

$$\rho = 1 + \frac{2eL_{\mathrm{res}}}{\sqrt{n}L_{\max}}, \tag{2.11}$$

define ψ by (2.5), and set $\gamma = 1/\psi$, we have for the essentially strongly convex case (2.3) with $l > 0$ that

$$\mathbb{E}(f(x_j) - f^*) \leqslant \left(1 - \frac{l}{2nL_{\max}}\right)^j (f(x_0) - f^*). \tag{2.12}$$

For the case of general convex f*, if, in addition, Assumption 2.1 is satisfied, we have*

$$\mathbb{E}(f(x_j) - f^*) \leqslant \frac{1}{(f(x_0) - f^*)^{-1} + \frac{j}{4n L_{\max} R^2}}. \tag{2.13}$$

We note that the linear rate (2.12) is broadly consistent with the linear rate for the classical steepest descent method applied to strongly convex functions, which has a rate constant of $(1-2l/L)$, where L is the standard Lipschitz constant for ∇f. If we assume (not unreasonably) that n steps of stochastic coordinate descent cost roughly the same as one step of steepest descent, and note from (2.12) that n steps of stochastic coordinate descent would achieve a reduction factor of about $(1-l/(2L_{\max}))$, a standard argument would suggest that stochastic coordinate descent would require about $4L_{\max}/L$ times more computation. (Note that $L_{\max}/L \in [1/n, 1]$.) The stochastic approach may gain an advantage from the parallel implementation, however. Steepest descent would require synchronization and careful division of evaluation work, whereas the stochastic approach can be implemented in an asynchronous fashion.

For the general convex case, (2.13) defines a sublinear rate, whose relationship with the rate of the steepest descent for general convex optimization is similar to the previous paragraph.

As noted in Section 2.2, the parameter τ is closely related to the number of cores that can be involved in the computation, without degrading the convergence performance of the algorithm. In other words, if the number of cores is small enough such that (2.10) holds, the convergence expressions (2.12), (2.13) do not depend on the number of cores, implying that linear speedup can be expected. A small value for the ratio $L_{\text{res}}/L_{\max}$ (not much greater than 1) implies a greater degree of potential parallelism. As we note at the end of Section 2.2, this ratio tends to be small in some important applications. In these situations, the maximal number of cores could be as high as $O(\sqrt{n})$.

We conclude this section with a high-probability estimate for convergence of $\{f(x_j)\}_{j=1,2,\ldots}$.

Theorem 2.3. *Suppose that the assumptions of Corollary 2.2 hold, including the definitions of* ρ *and* ψ*. Then for any* $\epsilon \in (0, f(x_0) - f^*)$ *and* $\eta \in (0,1)$*, we have that*

$$\mathbb{P}\left(f(x_j) - f^* \leqslant \epsilon\right) \geqslant 1 - \eta, \tag{2.14}$$

provided that either of the following sufficient conditions hold for the index j*. In the essentially strongly convex case* (2.3) *with* $l > 0$*, it suffices to have*

$$j \geqslant \frac{2nL_{\max}}{l} \left|\log \frac{f(x_0) - f^*}{\epsilon\eta}\right|, \tag{2.15}$$

while in the general convex case, a sufficient condition is

$$j \geqslant 4n L_{\max} R^2 \left(\frac{1}{\epsilon \eta} - \frac{1}{f(x_0) - f^*} \right). \tag{2.16}$$

2.5 Constrained Smooth Convex Case

This section considers the case of separable constraints (2.2). We show results about convergence rates and high-probability complexity estimates, analogous to those of the previous section. (Proofs appear in Appendix.)

As in the unconstrained case, the steplength γ should be chosen to ensure steady progress while ensuring that update information does not become too stale. Because constraints are present, the ratio (2.4) is no longer appropriate. We use instead a ratio of squares of expected differences in successive primal iterates:

$$\frac{\mathbb{E}\|x_{j-1} - \bar{x}_j\|^2}{\mathbb{E}\|x_j - \bar{x}_{j+1}\|^2}, \tag{2.17}$$

where $\bar{x}_{j+1}$ is the hypothesized full update obtained by applying the single-component update to *every* component of x_j, that is,

$$\bar{x}_{j+1} := \arg\min_{x \in \Omega} \langle \nabla f(x_{k(j)}), x - x_j \rangle + \frac{L_{\max}}{2\gamma} \|x - x_j\|^2.$$

In the unconstrained case $\Omega = \mathbb{R}^n$, the ratio (2.17) reduces to

$$\frac{\mathbb{E}\|\nabla f(x_{k(j-1)})\|^2}{\mathbb{E}\|\nabla f(x_{k(j)})\|^2},$$

which is evidently related to (2.4), but not identical.

We have the following result concerning convergence of the expected error to zero.

Theorem 2.4. *Suppose that Ω has the form (2.2) and that $n \geqslant 5$. Let ρ be a constant with $\rho > \left(1 - 2/\sqrt{n}\right)^{-1}$, and define the quantity ψ as follows:*

$$\psi := 1 + \frac{L_{\text{res}} \tau \rho^\tau}{\sqrt{n} L_{\max}} \left(2 + \frac{L_{\max}}{\sqrt{n} L_{\text{res}}} + \frac{2\tau}{n} \right). \tag{2.18}$$

Suppose that the steplength parameter $\gamma > 0$ satisfies the following two upper bounds:

$$\gamma \leqslant \frac{1}{\psi}, \quad \gamma \leqslant \left(1 - \frac{1}{\rho} - \frac{2}{\sqrt{n}} \right) \frac{\sqrt{n} L_{\max}}{4 L_{\text{res}} \tau \rho^\tau}. \tag{2.19}$$

Then we have

$$\mathbb{E}\|x_{j-1} - \bar{x}_j\|^2 \leqslant \rho \mathbb{E}\|x_j - \bar{x}_{j+1}\|^2, \quad j = 1, 2, \ldots. \tag{2.20}$$

If the essential strong convexity property (2.3) *holds with* $l > 0$*, we have for* $j = 1, 2, \ldots$ *that*

$$\begin{aligned} &\mathbb{E}\|x_j - \mathcal{P}_S(x_j)\|^2 + \frac{2\gamma}{L_{\max}}(\mathbb{E}f(x_j) - f^*) \\ &\leqslant \left(1 - \frac{l}{n(l+\gamma^{-1}L_{\max})}\right)^j \left(R_0^2 + \frac{2\gamma}{L_{\max}}(f(x_0) - f^*)\right). \end{aligned} \tag{2.21}$$

For general smooth convex function f*, we have*

$$\mathbb{E}f(x_j) - f^* \leqslant \frac{n(R_0^2 L_{\max} + 2\gamma(f(x_0) - f^*))}{2\gamma(n+j)}. \tag{2.22}$$

Similarly to the unconstrained case, the following corollary proposes an interesting particular choice for the parameters for which the convergence expressions become more comprehensible. The result requires a condition on the delay bound τ in terms of n and the ratio $L_{\max}/L_{\text{res}}$.

Corollary 2.5. *Suppose that* S *is nonempty, that* $\tau \geqslant 1$ *and* $n \geqslant 5$*, and that*

$$\tau(\tau+1) \leqslant \frac{\sqrt{n}L_{\max}}{4eL_{\text{res}}}. \tag{2.23}$$

If we choose

$$\rho = 1 + \frac{4e\tau L_{\text{res}}}{\sqrt{n}L_{\max}}, \tag{2.24}$$

then the steplength $\gamma = 1/2$ *will satisfy the bounds* (2.19)*. In addition, for the essentially strongly convex case* (2.3) *with* $l > 0$*, we have for* $j = 1, 2, \ldots$ *that*

$$\mathbb{E}(f(x_j) - f^*) \leqslant \left(1 - \frac{l}{n(l+2L_{\max})}\right)^j (L_{\max}R_0^2 + f(x_0) - f^*), \tag{2.25}$$

while for the case of general convex f*, we have*

$$\mathbb{E}(f(x_j) - f^*) \leqslant \frac{n(L_{\max}R_0^2 + f(x_0) - f^*)}{j+n}. \tag{2.26}$$

Similarly to Section 2.4, and provided τ satisfies (2.23), the convergence rate is not affected appreciably by the delay bound τ, and close-to-linear speedup can be expected for multicore implementations when (2.23) holds. This condition is more restrictive than (2.10) in the unconstrained case, but still holds in many problems for interesting values of τ. When $L_{\text{res}}/L_{\max}$ is bounded independently of dimension, the maximal number of cores allowed is of the the order of $n^{1/4}$, which is slightly smaller than the $O(n^{1/2})$ value obtained for the unconstrained case.

We conclude this section with another high-probability bound, whose proof tracks that of Theorem 2.3.

Theorem 2.6. *Suppose that the conditions of Corollary 2.5 hold, including the definitions of* ρ *and* ψ*. Then for* $\epsilon > 0$ *and* $\eta \in (0,1)$*, we have that*

$$\mathbb{P}\left(f(x_j) - f^* \leqslant \epsilon\right) \geqslant 1 - \eta,$$

provided that one of the following conditions holds: In the essentially strongly convex case (2.3) *with* $l > 0$*, we require*

$$j \geqslant \frac{n(l + 2L_{\max})}{l} \left|\log \frac{L_{\max} R_0^2 + f(x_0) - f^*}{\epsilon\eta}\right|,$$

while in the general convex case, it suffices that

$$j \geqslant \frac{n(L_{\max} R_0^2 + f(x_0) - f^*)}{\epsilon\eta} - n.$$

2.6 Experiments

We illustrate the behavior of two variants of the stochastic coordinate descent approach on test problems constructed from several data sets. Our interests are in the efficiency of multicore implementations (by comparison with a single-threaded implementation) and in performance relative to alternative solvers for the same problems.

All our test problems have the form (2.1), with either $\Omega = \mathbb{R}^n$ or Ω separable as in (2.2). The objective f is quadratic, that is,

$$f(x) = \frac{1}{2} x^T Q x + c^T x,$$

with Q symmetric positive definite.

Our implementation of AsySCD is called DIMM-WITTED (or DW for short). It runs on various numbers of threads, from 1 to 40, each thread assigned to a single core in our 40-core Intel Xeon architecture. Cores on the Xeon architecture are arranged into four sockets — ten cores per socket, with each socket having its own memory. Non-uniform memory access (NUMA) means that memory accesses to local memory (on the same socket as the core) are less expensive than accesses to memory on another socket. In our DW implementation, we assign each socket an equal-sized "slice" of Q, a row submatrix. The components of x are partitioned between cores, each core being responsible for updating its own partition of x (though it can read the components of x from other cores). The components of x assigned to the cores correspond to the rows of Q assigned to that core's socket. Computation is grouped into "epochs," where an epoch is defined to be the period of computation during which each component of x is updated exactly once. We use the parameter p to denote the number of epochs that are executed between reordering (shuffling) of the coordinates of x. We investigate both shuffling after every epoch ($p = 1$) and after every tenth epoch ($p = 10$). Access to x is lock-free, and updates are performed asynchronously. This update scheme does not implement exactly the "sampling with replacement" scheme analyzed in previous sections, but can

be viewed as a high performance, practical adaptation of the AsySCD method.

To do each coordinate descent update, a thread must read the latest value of x. Most components are already in the cache for that core, so that it only needs to fetch those components recently changed. When a thread writes to x_i, the hardware ensures that this x_i is simultaneously removed from other cores, signaling that they must fetch the updated version before proceeding with their respective computations.

Although DW is not a precise implementation of AsySCD, it largely achieves the consistent-read condition that is assumed by the analysis. Inconsistent read happens on a core only if the following three conditions are satisfied simultaneously:

- A core does not finish reading recently changed coordinates of x (note that it needs to read no more than τ coordinates);
- Among these recently changed coordinates, modifications take place both to coordinates that *have been read* and that are *still to be read* by this core;
- Modification of the already-read coordinates happens earlier than the modification of the still-unread coordinates.

Inconsistent read will occur only if at least two coordinates of x are modified twice during a stretch of approximately τ updates to x (that is, iterations of Algorithm 1). For the DW implementation, inconsistent read would require repeated updating of a particular component in a stretch of approximately τ iterations that straddles two epochs. This event would be rare, for typical values of n and τ. Of course, one can avoid the inconsistent read issue altogether by changing the shuffling rule slightly, enforcing the requirement that no coordinate can be modified twice in a span of τ iterations. From the practical perspective, this change does not improve performance, and detracts from the simplicity of the approach. From the theoretical perspective, however, the analysis for the inconsistent-read model would be interesting and meaningful, and we plan to study this topic in future work.

The first test problem `QP` is an unconstrained, regularized least squares problem constructed with synthetic data. It has the form

$$\min_{x \in \mathbb{R}^n} \; f(x) := \frac{1}{2}\|Ax - b\|^2 + \frac{\alpha}{2}\|x\|^2. \tag{2.27}$$

All elements of $A \in \mathbb{R}^{m \times n}$, the true model $\tilde{x} \in \mathbb{R}^n$, and the observation noise vector $\delta \in \mathbb{R}^m$ are generated in i.i.d. fashion from the Gaussian distribution $\mathcal{N}(0,1)$, following which each column in A is scaled to have a Euclidean norm of 1. The observation $b \in \mathbb{R}^m$ is constructed from $A\tilde{x} + \delta\|A\tilde{x}\|/(5m)$. We choose $m = 6000$, $n = 20000$, and $\alpha = 0.5$. We therefore have $L_{\max} = 1 + \alpha = 1.5$ and

$$\frac{L_{\text{res}}}{L_{\max}} \approx \frac{1 + \sqrt{n/m} + \alpha}{1 + \alpha} \approx 2.2.$$

This problem is diagonally dominant, and the condition (2.10) is satisfied when delay parameter τ is less than about 95. In Algorithm 1, we set the steplength parameter γ to 1, and we choose initial iterate to be $x_0 = \mathbf{0}$. We measure convergence of the residual norm $\|\nabla f(x)\|$.

Our second problem `QPc` is a bound-constrained version of (2.27):

$$\min_{x \in \mathbb{R}^n_+} \quad f(x) := \frac{1}{2}(x - \tilde{x})^T (A^T A + \alpha I)(x - \tilde{x}). \tag{2.28}$$

The methodology for generating A and $\tilde{x}$ and for choosing the values of m, n, γ, and x_0 is the same as for (2.27). We measure convergence via the residual $\|x - \mathcal{P}_\Omega(x - \nabla f(x))\|$, where Ω is the nonnegative orthant $\mathbb{R}^n_+$. At the solution of (2.28), about half the components of x are at their lower bound of 0.

Our third and fourth problems are quadratic penalty functions for linear programming relaxations of vertex cover problems on large graphs. The vertex cover problem for an undirected graph with edge set E and vertex set V can be written as a binary linear program:

$$\min_{y \in \{0,1\}^{|V|}} \sum_{v \in V} y_v \quad \text{subject to } \; y_u + y_v \geqslant 0, \quad \forall\, (u,v) \in E. \tag{2.29}$$

By relaxing each binary constraint to the interval $[0,1]$, introducing slack variables for the cover inequalities, we obtain a problem of the form

$$\min_{y_v \in [0,1],\, s_{uv} \in [0,1]} \sum_{v \in V} y_v \quad \text{subject to } \; y_u + y_v - s_{uv} = 0, \quad \forall\, (u,v) \in E. \tag{2.30}$$

This has the form

$$\min_{x \in [0,1]^n} \; c^T x \quad \text{subject to } \; Ax = b,$$

for $n = |V| + |E|$. The test problem (2.31) is a regularized quadratic penalty reformulation of this linear program for some penalty parameter β:

$$\min_{x \in [0,1]^n} \quad c^T x + \frac{\beta}{2}\|Ax - b\|^2 + \frac{1}{2\beta}\|x\|^2, \tag{2.31}$$

with $\beta = 5$. Since the solution to the original problem (2.29) is estimated by rounding the solution for (2.30), we do not need a highly accurate solution to (2.30). Thus, (2.31) can provide a sufficiently accurate solution to (2.30) when β is chosen to be appropriately large. Two test data sets `Amazon` and `DBLP` have dimensions $n = 561050$ and $n = 520891$, respectively.

We tracked the behavior of the residual as a function of the number of epochs, when executed on different numbers of cores. Figure 2.1 shows convergence behavior for each of our four test problems on various numbers of cores with two different shuffling periods: $p = 1$ and $p = 10$. We note the following points.

Problem	1 core	40 cores
QP	98.4	3.03
QPc	59.7	1.82
Amazon	17.1	1.25
DBLP	11.5	.91

Table 2.1: Runtimes (seconds) for the four test problems on 1 and 40 cores.

- The total amount of computation to achieve any level of precision appears to be almost independent of the number of cores, at least up to 40 cores. In this respect, the performance of the algorithm does not change appreciably as the number of cores is increased. Thus, any deviation from linear speedup is due not to degradation of convergence speed in the algorithm but rather to systems issues in the implementation.

- When we reshuffle after every epoch ($\mathsf{p} = 1$), convergence is slightly faster in synthetic unconstrained QP but slightly slower in Amazon and DBLP than when we do occasional reshuffling ($\mathsf{p} = 10$). Overall, the convergence rates with different shuffling periods are comparable in the sense of epochs. However, when the dimension of the variable is large, the shuffling operation becomes expensive, so we would recommend using a large value for p for large-dimensional problems.

Results for speedup on multicore implementations are shown in Figures 2.2 and 2.3 for DW with $\mathsf{p} = 10$. Speedup is defined as follows:

$$\frac{\text{runtime a single core using DW}}{\text{runtime on } \mathsf{P} \text{ cores}}.$$

Near-linear speedup can be observed for the two QP problems with synthetic data. For Problems 3 and 4, speedup is at most 12-14; there are few gains when the number of cores exceeds about 12. We believe that the degradation is due mostly to memory contention. Although these problems have high dimension, the matrix Q is very sparse (in contrast to the dense Q for the synthetic data set). Thus, the ratio of computation to data movement / memory access is much lower for these problems, making memory contention effects more significant.

Figures 2.2 and 2.3 also show results of a global-locking strategy for the parallel stochastic coordinate descent method, in which the whole vector x is locked by a core whenever it performs a read or update. The performance curve for this strategy hugs the horizontal axis; it is not competitive.

Wall clock times required for the four test problems on 1 and 40 cores, to reduce residuals below 10^{-5} are shown in Table 2.1. (Similar speedups are noted when we use a convergence tolerance looser than 10^{-5}.)

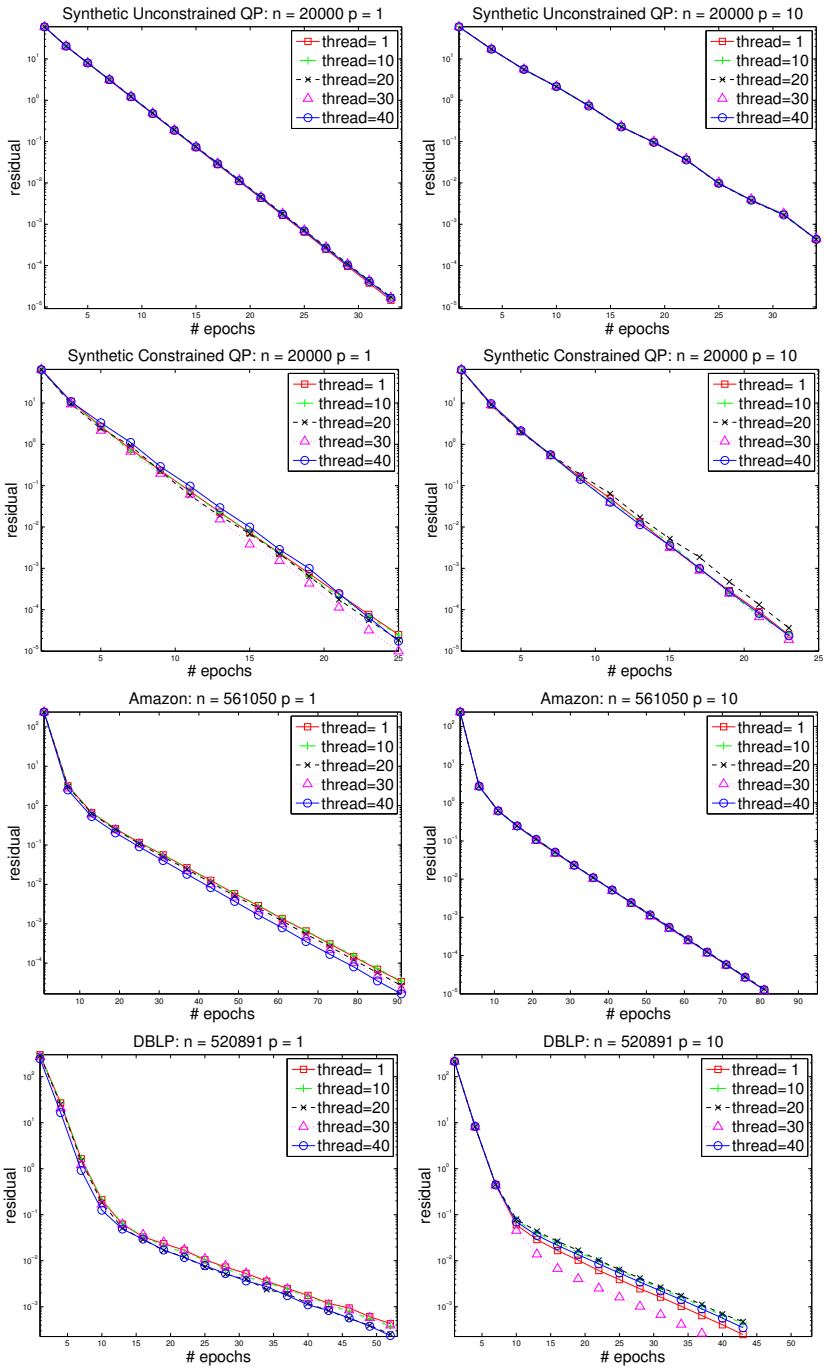

Figure 2.1: Residuals vs epoch number for the four test problems. Results are reported for variants in which indices are reshuffled after every epoch ($\mathtt{p} = 1$) and after every tenth epoch ($\mathtt{p} = 10$).

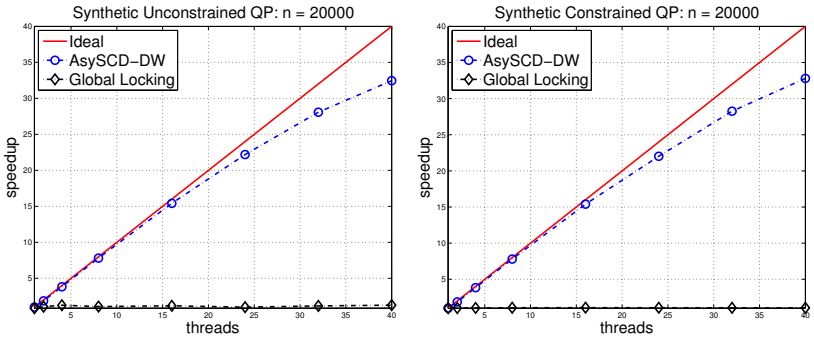

Figure 2.2: Test problems 1 and 2: Speedup of multicore implementations of DW on up to 40 cores of an Intel Xeon architecture. Ideal (linear) speedup curve is shown for reference, along with poor speedups obtained for a global-locking strategy.

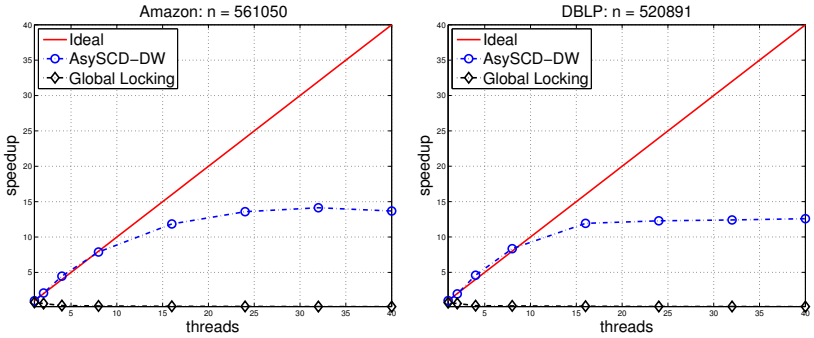

Figure 2.3: Test problems 3 and 4: Speedup of multicore implementations of DW on up to 40 cores of an Intel Xeon architecture. Ideal (linear) speedup curve is shown for reference, along with poor speedups obtained for a global-locking strategy.

#cores	Time(sec)	Speedup
1	55.9	1
10	5.19	10.8
20	2.77	20.2
30	2.06	27.2
40	1.81	30.9

Table 2.2: Runtimes (seconds) and speedup for multicore implementations of DW on different number of cores for the weakly convex `QPc` problem (with $\alpha = 0$) to achieve a residual below 0.06.

All problems reported on above are essentially strongly convex. Similar speedup properties can be obtained in the weakly convex case as well. We show only speedups for the `QPc` problem with $\alpha = 0$. Table 2.2 demonstrates similar speedup to the essentially strongly convex case shown in Figure 2.2.

Turning now to comparisons between AsySCD and alternative algorithms, we start by consid-

#cores	Time(sec) SynGD / AsySCD	Speedup SynGD / AsySCD
1	121. / 27.1	0.22 / 1.00
10	11.4 / 2.57	2.38 / 10.5
20	6.00 / 1.36	4.51 / 19.9
30	4.44 / 1.01	6.10 / 26.8
40	3.91 / 0.88	6.93 / 30.8

Table 2.3: Efficiency comparison between SynGD and AsySCD for the `QP` problem. The running time and speedup are based on the residual achieving a tolerance of 10^{-5}.

Dataset	# of Samples	# of Features	Train time(sec) LIBSVM	AsySCD
adult	32561	123	16.15	1.39
news	19996	1355191	214.48	7.22
rcv	20242	47236	40.33	16.06
reuters	8293	18930	1.63	0.81
w8a	49749	300	33.62	5.86

Table 2.4: Efficiency comparison between LIBSVM and AsySCD for kernel SVM using 40 cores using homogeneous kernels ($K(x_i, x_j) = (x_i^T x_j)^2$). The running time and speedup are calculated based on the "residual" 10^{-3}. Here, to make both algorithms comparable, the "residual" is defined by $\|x - \mathcal{P}_\Omega(x - \nabla f(x))\|_\infty$.

ering the basic gradient descent method. We implement gradient descent in a parallel, synchronous fashion, distributing the gradient computation load on multiple cores and updates the variable x in parallel at each step. The resulting implementation is called SynGD. Table 2.3 reports running time and speedup of both AsySCD over SynGD, showing a clear advantage for AsySCD.

Next we compare AsySCD to LIBSVM (Chang and Lin, 2011) a popular parallel solver for kernel support vector machines (SVM). Both algorithms are run on 40 cores to solve the dual formulation of kernel SVM, without an intercept term. All datasets used in 2.4 except reuters were obtained from the LIBSVM dataset repository[1]. The dataset reuters is a sparse binary text classification dataset constructed as a one-versus-all version of Reuters-2159[2]. Our comparisons, shown in Table 2.4, indicate that AsySCD outperforms LIBSVM on these test sets.

Sridhar et al. (2013) developed an efficient LP solver by relaxing an LP problem into a bound-constrained QP problem, which is then solved by AsySCD. Please refer to their paper for more experiments and comparison.

[1] `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`

[2] `http://www.daviddlewis.com/resources/testcollections/reuters21578/`

2.7 Extension

The AsySCD algorithm can be extended by partitioning the coordinates into blocks, and modifying Algorithm 1 to work with these blocks rather than with single coordinates. If L_i, $L_{\max}$, and L_{res} are defined in the block sense, as follows:

$$\|\nabla f(x) - \nabla f(x + E_i t)\| \leqslant L_{\text{res}} \|t\| \quad \forall x, i, t \in \mathbb{R}^{|i|},$$
$$\|\nabla_i f(x) - \nabla_i f(x + E_i t)\| \leqslant L_i \|t\| \quad \forall x, i, t \in \mathbb{R}^{|i|},$$
$$L_{\max} = \max_i L_i,$$

where E_i is the projection from the ith block to $\mathbb{R}^n$ and $|i|$ denotes the number of components in block i, our analysis can be extended appropriately.

To make the AsySCD algorithm more efficient, one can redefine the steplength in Algorithm 1 to be $\frac{\gamma}{L_{i(j)}}$ rather than $\frac{\gamma}{L_{\max}}$. Our analysis can be applied to this variant by doing a change of variables to $\tilde{x}$, with $x_i = \frac{L_i}{L_{\max}} \tilde{x}_i$ and defining L_i, L_{res}, and $L_{\max}$ in terms of $\tilde{x}$.

2.8 Conclusion

This chapter proposes an asynchronous parallel stochastic coordinate descent algorithm for minimizing convex objectives, in the unconstrained and separable-constrained cases. Sublinear convergence (at rate $1/K$) is proved for general convex functions, with stronger linear convergence results for functions that satisfy an essential strong convexity property. Our analysis indicates the extent to which parallel implementations can be expected to yield near-linear speedup, in terms of a parameter that quantifies the cross-coordinate interactions in the gradient ∇f and a parameter τ that bounds the delay in updating. Our computational experience confirms the theory.

The analysis extends almost immediately to a *block*-coordinate update scheme, provided that the constraints are block-separable.

Appendix

This section containts convergence proofs for AsySCD in the unconstrained case and constrained case.

Proofs for Unconstrained Case

In Algorithm 1, the indices $i(j)$, $j = 0, 1, 2, \ldots$ are random variables. We denote the expectation over all random variables as $\mathbb{E}$, the conditional expectation in term of $i(j)$ given $i(0), i(1), \cdots, i(j-1)$ as $\mathbb{E}_{i(j)}$. We start with a technical result, then move to the proof of Theorem 2.1.

Lemma 2.7. *For any* x*, we have*

$$\|x - \mathcal{P}_S(x)\|^2 \|\nabla f(x)\|^2 \geqslant (f(x) - f^*)^2.$$

If the essential strong convexity property (2.3) *holds, we have*

$$\|\nabla f(x)\|^2 \geqslant 2l(f(x) - f^*).$$

Proof. The first inequality is proved as follows:

$$f(x) - f^* \leqslant \langle \nabla f(x), x - \mathcal{P}_S(x) \rangle \leqslant \|\nabla f(x)\| \|\mathcal{P}_S(x) - x\|.$$

For the second bound, we have from the definition (2.3), setting $y \leftarrow x$ and $x \leftarrow \mathcal{P}_S(x)$, that

$$\begin{aligned} f^* - f(x) &\geqslant \langle \nabla f(x), \mathcal{P}_S(x) - x \rangle + \frac{l}{2}\|x - \mathcal{P}_S(x)\|^2 \\ &= \frac{l}{2}\|\mathcal{P}_S(x) - x + \frac{1}{l}\nabla f(x)\|^2 - \frac{1}{2l}\|\nabla f(x)\|^2 \geqslant -\frac{1}{2l}\|\nabla f(x)\|^2, \end{aligned}$$

as required. □

Proof of Theorem 2.1

Proof. We prove each of the two inequalities in (2.7) by induction. We start with the left-hand inequality. For all values of j, we have

$$\begin{aligned} &\mathbb{E}\left(\|\nabla f(x_j)\|^2 - \|\nabla f(x_{j+1})\|^2\right) \\ &\quad = \mathbb{E}\langle \nabla f(x_j) + \nabla f(x_{j+1}), \nabla f(x_j) - \nabla f(x_{j+1}) \rangle \\ &\quad = \mathbb{E}\langle 2\nabla f(x_j) + \nabla f(x_{j+1}) - \nabla f(x_j), \nabla f(x_j) - \nabla f(x_{j+1}) \rangle \\ &\quad \leqslant 2\mathbb{E}\langle \nabla f(x_j), \nabla f(x_j) - \nabla f(x_{j+1}) \rangle \\ &\quad \leqslant 2\mathbb{E}(\|\nabla f(x_j)\| \|\nabla f(x_j) - \nabla f(x_{j+1})\|) \\ &\quad \leqslant 2L_{\text{res}}\mathbb{E}(\|\nabla f(x_j)\| \|x_j - x_{j+1}\|) \\ &\quad \leqslant \frac{2L_{\text{res}}\gamma}{L_{\max}}\mathbb{E}(\|\nabla f(x_j)\| \|\nabla_{i(j)} f(x_{k(j)})\|) \\ &\quad \leqslant \frac{L_{\text{res}}\gamma}{L_{\max}}\mathbb{E}(n^{-1/2}\|\nabla f(x_j)\|^2 + n^{1/2}\|\nabla_{i(j)} f(x_{k(j)})\|^2) \\ &\quad = \frac{L_{\text{res}}\gamma}{L_{\max}}\mathbb{E}(n^{-1/2}\|\nabla f(x_j)\|^2 + n^{1/2}\mathbb{E}_{i(j)}(\|\nabla_{i(j)} f(x_{k(j)})\|^2)) \\ &\quad = \frac{L_{\text{res}}\gamma}{L_{\max}}\mathbb{E}(n^{-1/2}\|\nabla f(x_j)\|^2 + n^{-1/2}\|\nabla f(x_{k(j)})\|^2) \\ &\quad \leqslant \frac{L_{\text{res}}\gamma}{\sqrt{n}L_{\max}}\mathbb{E}\left(\|\nabla f(x_j)\|^2 + \|\nabla f(x_{k(j)})\|^2\right). \end{aligned} \tag{2.32}$$

We can use this bound to show that the left-hand inequality in (2.7) holds for $j = 0$. By setting $j = 0$ in (2.32) and noting that $k(0) = 0$, we obtain

$$\mathbb{E}\left(\|\nabla f(x_0)\|^2 - \|\nabla f(x_1)\|^2\right) \leqslant \frac{L_{\text{res}}\gamma}{\sqrt{n}L_{\max}} 2\mathbb{E}(\|\nabla f(x_0)\|^2). \tag{2.33}$$

From (2.6b), we have

$$\frac{2L_{\text{res}}\gamma}{\sqrt{n}L_{\max}} \leqslant \frac{\rho - 1}{\rho^\tau} \leqslant \frac{\rho - 1}{\rho} = 1 - \rho^{-1},$$

where the second inequality follows from $\rho > 1$. By substituting into (2.33), we obtain $\rho^{-1}\mathbb{E}(\|\nabla f(x_0)\|^2) \leqslant \mathbb{E}(\|\nabla f(x_1)\|^2)$, establishing the result for $j = 1$. For the inductive step, we use (2.32) again, assuming that the left-hand inequality in (2.7) holds up to stage j, and thus that

$$\mathbb{E}(\|\nabla f(x_{k(j)})\|^2) \leqslant \rho^\tau \mathbb{E}(\|\nabla f(x_j)\|^2),$$

provided that $0 \leqslant j - k(j) \leqslant \tau$, as assumed. By substituting into the right-hand side of (2.32) again, and using $\rho > 1$, we obtain

$$\mathbb{E}\left(\|\nabla f(x_j)\|^2 - \|\nabla f(x_{j+1})\|^2\right) \leqslant \frac{2L_{\text{res}}\gamma\rho^\tau}{\sqrt{n}L_{\max}} \mathbb{E}\left(\|\nabla f(x_j)\|^2\right).$$

By substituting (2.6b) we conclude that the left-hand inequality in (2.7) holds for all j.

We now work on the right-hand inequality in (2.7). For all j, we have the following:

$$\begin{aligned}
&\mathbb{E}\left(\|\nabla f(x_{j+1})\|^2 - \|\nabla f(x_j)\|^2\right) \\
&\quad = \mathbb{E}\langle \nabla f(x_j) + \nabla f(x_{j+1}), \nabla f(x_{j+1}) - \nabla f(x_j)\rangle \\
&\quad \leqslant \mathbb{E}(\|\nabla f(x_j) + \nabla f(x_{j+1})\|\|\nabla f(x_j) - \nabla f(x_{j+1})\|) \\
&\quad \leqslant L_{\mathrm{res}}\mathbb{E}(\|\nabla f(x_j) + \nabla f(x_{j+1})\|\|x_j - x_{j+1}\|) \\
&\quad \leqslant L_{\mathrm{res}}\mathbb{E}((2\|\nabla f(x_j)\| + \|\nabla f(x_{j+1}) - \nabla f(x_j)\|)\|x_j - x_{j+1}\|) \\
&\quad \leqslant L_{\mathrm{res}}\mathbb{E}(2\|\nabla f(x_j)\|\|x_j - x_{j+1}\| + L_{\mathrm{res}}\|x_j - x_{j+1}\|^2) \\
&\quad \leqslant L_{\mathrm{res}}\mathbb{E}\left(\frac{2\gamma}{L_{\max}}\|\nabla f(x_j)\|\|\nabla_{i(j)} f(x_{k(j)})\| + \frac{L_{\mathrm{res}}\gamma^2}{L_{\max}^2}\|\nabla_{i(j)} f(x_{k(j)})\|^2\right) \\
&\quad \leqslant L_{\mathrm{res}}\mathbb{E}\left(\frac{\gamma}{L_{\max}}(n^{-1/2}\|\nabla f(x_j)\|^2 + n^{1/2}\|\nabla_{i(j)} f(x_{k(j)})\|^2 + \frac{L_{\mathrm{res}}\gamma^2}{L_{\max}^2}\|\nabla_{i(j)} f(x_{k(j)})\|^2\right) \\
&\quad = L_{\mathrm{res}}\mathbb{E}\left(\frac{\gamma}{L_{\max}}(n^{-1/2}\|\nabla f(x_j)\|^2 + n^{1/2}\mathbb{E}_{i(j)}(\|\nabla_{i(j)} f(x_{k(j)})\|^2)) + \right. \\
&\qquad \left.\frac{L_{\mathrm{res}}\gamma^2}{L_{\max}^2}\mathbb{E}_{i(j)}(\|\nabla_{i(j)} f(x_{k(j)})\|^2)\right) \\
&\quad = L_{\mathrm{res}}\mathbb{E}\left(\frac{\gamma}{L_{\max}}(n^{-1/2}\|\nabla f(x_j)\|^2 + n^{-1/2}\|\nabla f(x_{k(j)})\|^2) + \frac{L_{\mathrm{res}}\gamma^2}{nL_{\max}^2}\|\nabla f(x_{k(j)})\|^2\right) \\
&\quad = \frac{\gamma L_{\mathrm{res}}}{\sqrt{n}L_{\max}}\mathbb{E}\left(\|\nabla f(x_j)\|^2 + \|\nabla f(x_{k(j)})\|^2\right) + \frac{\gamma^2 L_{\mathrm{res}}^2}{nL_{\max}^2}\mathbb{E}(\|\nabla f(x_{k(j)})\|^2) \\
&\quad \leqslant \frac{\gamma L_{\mathrm{res}}}{\sqrt{n}L_{\max}}\mathbb{E}(\|\nabla f(x_j)\|^2) + \left(\frac{\gamma L_{\mathrm{res}}}{\sqrt{n}L_{\max}} + \frac{\gamma L_{\mathrm{res}}^2}{nL_{\max}^2}\right)\mathbb{E}(\|\nabla f(x_{k(j)})\|^2), \qquad (2.34)
\end{aligned}$$

where the last inequality is from the observation $\gamma \leqslant 1$. By setting $j = 0$ in this bound, and noting that $k(0) = 0$, we obtain

$$\mathbb{E}\left(\|\nabla f(x_1)\|^2 - \|\nabla f(x_0)\|^2\right) \leqslant \left(\frac{2\gamma L_{\mathrm{res}}}{\sqrt{n}L_{\max}} + \frac{\gamma L_{\mathrm{res}}^2}{nL_{\max}^2}\right)\mathbb{E}(\|\nabla f(x_0)\|^2). \qquad (2.35)$$

By using (2.6c), we have

$$\frac{2\gamma L_{\mathrm{res}}}{\sqrt{n}L_{\max}} + \frac{\gamma L_{\mathrm{res}}^2}{nL_{\max}^2} = \frac{L_{\mathrm{res}}\gamma}{\sqrt{n}L_{\max}}\left(2 + \frac{L_{\mathrm{res}}}{\sqrt{n}L_{\max}}\right) \leqslant \frac{\rho - 1}{\rho^\tau} < \rho - 1,$$

where the last inequality follows from $\rho > 1$. By substituting into (2.35), we obtain $\mathbb{E}(\|\nabla f(x_1)\|^2) \leqslant \rho\mathbb{E}(\|\nabla f(x_0)\|^2)$, so the right-hand bound in (2.7) is established for $j = 0$. For the inductive step, we use (2.34) again, assuming that the right-hand inequality in (2.7) holds up to stage j, and thus that

$$\mathbb{E}(\|\nabla f(x_j)\|^2) \leqslant \rho^\tau\mathbb{E}(\|\nabla f(x_{k(j)})\|^2),$$

provided that $0 \leqslant j - k(j) \leqslant \tau$, as assumed. From (2.34) and the left-hand inequality in (2.7), we have by substituting this bound that

$$\mathbb{E}\left(\|\nabla f(x_{j+1})\|^2 - \|\nabla f(x_j)\|^2\right) \leqslant \left(\frac{2\gamma L_{\text{res}}\rho^\tau}{\sqrt{n}L_{\max}} + \frac{\gamma L_{\text{res}}^2\rho^\tau}{nL_{\max}^2}\right)\mathbb{E}(\|\nabla f(x_j)\|^2). \tag{2.36}$$

It follows immediately from (2.6c) that the term in parentheses in (2.36) is bounded above by $\rho - 1$. By substituting this bound into (2.36), we obtain $\mathbb{E}(\|\nabla f(x_{j+1})\|^2) \leqslant \rho\mathbb{E}(\|\nabla f(x_j)\|^2)$, as required.

At this point, we have shown that both inequalities in (2.7) are satisfied for all j.

Next we prove (2.8) and (2.9). Take the expectation of $f(x_{j+1})$ in terms of $i(j)$:

$$\begin{aligned}
\mathbb{E}_{i(j)} f(x_{j+1}) &= \mathbb{E}_{i(j)} f\left(x_j - \frac{\gamma}{L_{\max}} e_{i(j)} \nabla_{i(j)} f(x_{k(j)})\right) \\
&= \frac{1}{n}\sum_{i=1}^{n} f\left(x_j - \frac{\gamma}{L_{\max}} e_i \nabla_i f(x_{k(j)})\right) \\
&\leqslant \frac{1}{n}\sum_{i=1}^{n} f(x_j) - \frac{\gamma}{L_{\max}}\langle \nabla f(x_j), e_i \nabla_i f(x_{k(j)})\rangle + \frac{L_i}{2L_{\max}^2}\gamma^2 \|\nabla_i f(x_{k(j)})\|^2 \\
&\leqslant f(x_j) - \frac{\gamma}{nL_{\max}}\langle \nabla f(x_j), \nabla f(x_{k(j)})\rangle + \frac{\gamma^2}{2nL_{\max}}\|\nabla f(x_{k(j)})\|^2 \\
&= f(x_j) + \frac{\gamma}{nL_{\max}} \underbrace{\langle \nabla f(x_{k(j)}) - \nabla f(x_j), \nabla f(x_{k(j)})\rangle}_{T_1} \\
&\qquad - \left(\frac{\gamma}{nL_{\max}} - \frac{\gamma^2}{2nL_{\max}}\right)\|\nabla f(x_{k(j)})\|^2.
\end{aligned} \tag{2.37}$$

The second term T_1 is caused by delay. If there is no the delay issue, T_1 should be 0 because of $\nabla f(x_j) = \nabla f(x_{k(j)})$. We estimate the upper bound of $\|\nabla f(x_{k(j)}) - \nabla f(x_j)\|$:

$$\begin{aligned}
\|\nabla f(x_{k(j)}) - \nabla f(x_j)\| &\leqslant \sum_{d=k(j)}^{j-1} \|\nabla f(x_{d+1}) - \nabla f(x_d)\| \\
&\leqslant L_{\text{res}} \sum_{d=k(j)}^{j-1} \|x_{d+1} - x_d\| \\
&= \frac{L_{\text{res}}\gamma}{L_{\max}} \sum_{d=k(j)}^{j-1} \left\|\nabla_{i(d)} f(x_{k(d)})\right\|.
\end{aligned} \tag{2.38}$$

Then $\mathbb{E}(|T_1|)$ can be bounded by

$$\begin{aligned}
\mathbb{E}(|T_1|) &\leqslant \mathbb{E}(\|\nabla f(x_{k(j)}) - \nabla f(x_j)\|\|\nabla f(x_{k(j)})\|) \\
&\leqslant \frac{L_{res}\gamma}{L_{max}}\mathbb{E}\left(\sum_{d=k(j)}^{j-1} \|\nabla_{i(d)} f(x_{k(d)})\|\|\nabla f(x_{k(j)})\|\right) \\
&\leqslant \frac{L_{res}\gamma}{2L_{max}}\mathbb{E}\left(\sum_{d=k(j)}^{j-1} n^{1/2}\|\nabla_{i(d)} f(x_{k(d)})\|^2 + n^{-1/2}\|\nabla f(x_{k(j)})\|^2\right) \\
&= \frac{L_{res}\gamma}{2L_{max}}\mathbb{E}\left(\sum_{d=k(j)}^{j-1} n^{1/2}\mathbb{E}_{i(d)}(\|\nabla_{i(d)} f(x_{k(d)})\|^2) + n^{-1/2}\|\nabla f(x_{k(j)})\|^2\right) \\
&= \frac{L_{res}\gamma}{2L_{max}}\mathbb{E}\left(\sum_{d=k(j)}^{j-1} n^{-1/2}\|\nabla f(x_{k(d)})\|^2 + n^{-1/2}\|\nabla f(x_{k(j)})\|^2\right) \\
&= \frac{L_{res}\gamma}{2\sqrt{n}L_{max}} \sum_{d=k(j)}^{j-1} \mathbb{E}(\|\nabla f(x_{k(d)})\|^2 + \|\nabla f(x_{k(j)})\|^2) \\
&\leqslant \frac{\tau\rho^{\tau}L_{res}\gamma}{\sqrt{n}L_{max}}\mathbb{E}(\|\nabla f(x_{k(j)})\|^2) \qquad (2.39)
\end{aligned}$$

where the second line uses (2.38), and the final inequality uses the fact for d between $k(j)$ and $j-1$, $k(d)$ lies in the range $k(j)-\tau$ and $j-1$, so we have $|k(d)-k(j)| \leqslant \tau$ for all d.

Taking expectation on both sides of (2.37) in terms of all random variables, together with (2.39), we obtain

$$\begin{aligned}
&\mathbb{E}(f(x_{j+1}) - f^*) \\
&\leqslant \mathbb{E}(f(x_j) - f^*) + \frac{\gamma}{nL_{max}}\mathbb{E}(|T_1|) - \left(\frac{\gamma}{nL_{max}} - \frac{\gamma^2}{2nL_{max}}\right)\mathbb{E}(\|\nabla f(x_{k(j)})\|^2) \\
&\leqslant \mathbb{E}(f(x_j) - f^*) - \left(\frac{\gamma}{nL_{max}} - \frac{\tau\rho^{\tau}L_{res}\gamma^2}{n^{3/2}L_{max}^2} - \frac{\gamma^2}{2nL_{max}}\right)\mathbb{E}(\|\nabla f(x_{k(j)})\|^2) \\
&= \mathbb{E}(f(x_j) - f^*) - \frac{\gamma}{nL_{max}}\left(1 - \frac{\psi}{2}\gamma\right)\mathbb{E}(\|\nabla f(x_{k(j)})\|^2), \qquad (2.40)
\end{aligned}$$

which (because of (2.6a)) implies that $\mathbb{E}(f(x_j) - f^*)$ is monotonically decreasing.

Assume now that the essential strong convexity property (2.3) holds. From Lemma 2.7 and the fact that $\mathbb{E}(f(x_j) - f^*)$ is monotonically decreasing, we have

$$\|\nabla f(x_{k(j)})\|^2 \geqslant 2l\mathbb{E}(f(x_{k(j)}) - f^*) \geqslant 2l\mathbb{E}(f(x_j) - f^*).$$

So by substituting in (2.40), we obtain

$$\mathbb{E}(f(x_{j+1}) - f^*) \leqslant \left(1 - \frac{2l\gamma}{n L_{\max}} \left(1 - \frac{\psi}{2}\gamma\right)\right) \mathbb{E}(f(x_j) - f^*),$$

from which the linear convergence claim (2.8) follows by an obvious induction.

For the case of general smooth convex f, we have from Lemma 5.2, Assumption 2.1, and the monotonic decreasing property for $\mathbb{E}(f(x_j) - f^*)$ that

$$\|\nabla f(x_{k(j)})\|^2 \geqslant \frac{(f(x_{k(j)}) - f^*)^2}{\|x_{k(j)} - \mathcal{P}_S(x_{k(j)})\|^2} \geqslant \frac{(f(x_{k(j)}) - f^*)^2}{R^2} \geqslant \frac{(f(x_j) - f^*)^2}{R^2}.$$

By substituting into (2.40), we obtain

$$\begin{aligned}
\mathbb{E}(f(x_{j+1}) - f^*) &\leqslant \mathbb{E}(f(x_j) - f^*) - \frac{\gamma}{n L_{\max} R^2} \left(1 - \frac{\psi}{2}\gamma\right) \mathbb{E}((f(x_j) - f^*)^2) \\
&\leqslant \mathbb{E}(f(x_j) - f^*) - \frac{\gamma}{n L_{\max} R^2} \left(1 - \frac{\psi}{2}\gamma\right) (\mathbb{E}(f(x_j) - f^*))^2,
\end{aligned}$$

where the second inequality uses Jensen's inequality $\mathbb{E}(\nu^2) \geqslant (\mathbb{E}(\nu))^2$. Defining

$$C := \frac{\gamma}{n L_{\max} R^2} \left(1 - \frac{\psi}{2}\gamma\right),$$

we have

$$\begin{aligned}
& \mathbb{E}(f(x_{j+1}) - f^*) \leqslant \mathbb{E}(f(x_j) - f^*) - C(\mathbb{E}(f(x_j) - f^*))^2 \\
\Rightarrow\ & \frac{1}{\mathbb{E}(f(x_j) - f^*)} \leqslant \frac{1}{\mathbb{E}(f(x_{j+1}) - f^*)} - C\frac{\mathbb{E}(f(x_j) - f^*)}{\mathbb{E}(f(x_{j+1}) - f^*)} \\
\Rightarrow\ & \frac{1}{\mathbb{E}(f(x_{j+1}) - f^*)} - \frac{1}{\mathbb{E}(f(x_j) - f^*)} \geqslant C\frac{\mathbb{E}(f(x_j) - f^*)}{\mathbb{E}(f(x_{j+1}) - f^*)} \geqslant C \\
\Rightarrow\ & \frac{1}{\mathbb{E}(f(x_{j+1}) - f^*)} \geqslant \frac{1}{f(x_0) - f^*} + C(j+1) \\
\Rightarrow\ & \mathbb{E}(f(x_{j+1}) - f^*) \leqslant \frac{1}{(f(x_0) - f^*)^{-1} + C(j+1)},
\end{aligned}$$

which completes the proof of the sublinear rate (2.9). □

Proof. (Corollary 2.2) Note first that for ρ defined by (2.11), we have

$$\rho^\tau \leqslant \rho^{\tau+1} = \left(\left(1 + \frac{2eL_{\text{res}}}{\sqrt{n}L_{\max}}\right)^{\frac{\sqrt{n}L_{\max}}{2eL_{\text{res}}}}\right)^{\frac{2eL_{\text{res}}(\tau+1)}{\sqrt{n}L_{\max}}} \leqslant e^{\frac{2eL_{\text{res}}(\tau+1)}{\sqrt{n}L_{\max}}} \leqslant e,$$

and thus from the definition of ψ (2.5) that

$$\psi = 1 + \frac{2\tau\rho^{\tau}\mathsf{L}_{\text{res}}}{\sqrt{n}\mathsf{L}_{\max}} \leqslant 1 + \frac{2\tau e\mathsf{L}_{\text{res}}}{\sqrt{n}\mathsf{L}_{\max}} \leqslant 2. \tag{2.41}$$

We show now that the steplength parameter choice $\gamma = 1/\psi$ satisfies all the bounds in (2.6), by showing that the second and third bounds are implied by the first. For the second bound (2.6b), we have

$$\frac{(\rho-1)\sqrt{n}\mathsf{L}_{\max}}{2\rho^{\tau+1}\mathsf{L}_{\text{res}}} \geqslant \frac{(\rho-1)\sqrt{n}\mathsf{L}_{\max}}{2e\mathsf{L}_{\text{res}}} \geqslant 1 \geqslant \frac{1}{\psi},$$

where the second inequality follows from (2.11). For the third bound (2.6c), we have

$$\frac{(\rho-1)\sqrt{n}\mathsf{L}_{\max}}{\mathsf{L}_{\text{res}}\rho^{\tau}(2+\frac{\mathsf{L}_{\text{res}}}{\sqrt{n}\mathsf{L}_{\max}})} = \frac{2e\mathsf{L}_{\text{res}}}{\mathsf{L}_{\text{res}}\rho^{\tau}(2+\frac{\mathsf{L}_{\text{res}}}{\sqrt{n}\mathsf{L}_{\max}})} \geqslant \frac{2e\mathsf{L}_{\text{res}}}{\mathsf{L}_{\text{res}}e(2+\frac{\mathsf{L}_{\text{res}}}{\sqrt{n}\mathsf{L}_{\max}})} = \frac{2}{2+\frac{\mathsf{L}_{\text{res}}}{\sqrt{n}\mathsf{L}_{\max}}} \geqslant \frac{1}{\psi}.$$

We can thus set $\gamma = 1/\psi$, and by substituting this choice into (2.8) and using (2.42), we obtain (2.12). We obtain (2.13) by making the same substitution into (2.9). □

Proof of Corollary 2.2

Proof. Note first that for ρ defined by (2.11), we have

$$\rho^{\tau} \leqslant \rho^{\tau+1} = \left(\left(1+\frac{2e\mathsf{L}_{\text{res}}}{\sqrt{n}\mathsf{L}_{\max}}\right)^{\frac{\sqrt{n}\mathsf{L}_{\max}}{2e\mathsf{L}_{\text{res}}}}\right)^{\frac{2e\mathsf{L}_{\text{res}}(\tau+1)}{\sqrt{n}\mathsf{L}_{\max}}} \leqslant e^{\frac{2e\mathsf{L}_{\text{res}}(\tau+1)}{\sqrt{n}\mathsf{L}_{\max}}} \leqslant e,$$

and thus from the definition of ψ (2.5) that

$$\psi = 1 + \frac{2\tau\rho^{\tau}\mathsf{L}_{\text{res}}}{\sqrt{n}\mathsf{L}_{\max}} \leqslant 1 + \frac{2\tau e\mathsf{L}_{\text{res}}}{\sqrt{n}\mathsf{L}_{\max}} \leqslant 2. \tag{2.42}$$

We show now that the steplength parameter choice $\gamma = 1/\psi$ satisfies all the bounds in (2.6), by showing that the second and third bounds are implied by the first. For the second bound (2.6b), we have

$$\frac{(\rho-1)\sqrt{n}\mathsf{L}_{\max}}{2\rho^{\tau+1}\mathsf{L}_{\text{res}}} \geqslant \frac{(\rho-1)\sqrt{n}\mathsf{L}_{\max}}{2e\mathsf{L}_{\text{res}}} \geqslant 1 \geqslant \frac{1}{\psi},$$

where the second inequality follows from (2.11). For the third bound (2.6c), we have

$$\frac{(\rho-1)\sqrt{n}\mathsf{L}_{\max}}{\mathsf{L}_{\text{res}}\rho^{\tau}(2+\frac{\mathsf{L}_{\text{res}}}{\sqrt{n}\mathsf{L}_{\max}})} = \frac{2e\mathsf{L}_{\text{res}}}{\mathsf{L}_{\text{res}}\rho^{\tau}(2+\frac{\mathsf{L}_{\text{res}}}{\sqrt{n}\mathsf{L}_{\max}})} \geqslant \frac{2e\mathsf{L}_{\text{res}}}{\mathsf{L}_{\text{res}}e(2+\frac{\mathsf{L}_{\text{res}}}{\sqrt{n}\mathsf{L}_{\max}})} = \frac{2}{2+\frac{\mathsf{L}_{\text{res}}}{\sqrt{n}\mathsf{L}_{\max}}} \geqslant \frac{1}{\psi}.$$

We can thus set $\gamma = 1/\psi$, and by substituting this choice into (2.8) and using (2.42), we obtain (2.12). We obtain (2.13) by making the same substitution into (2.9). □

Proof of Theorem 2.3

Proof. From Markov's inequality, we have

$$\begin{aligned}
\mathbb{P}(f(x_j) - f^* \geqslant \epsilon) &\leqslant \epsilon^{-1}\mathbb{E}(f(x_j) - f^*) \\
&\leqslant \epsilon^{-1}\left(1 - \frac{l}{2nL_{\max}}\right)^j (f(x_0) - f^*) \\
&\leqslant \epsilon^{-1}(1-c)^{(1/c)\left|\log \frac{f(x_0)-f^*}{n\epsilon}\right|} (f(x_0) - f^*) \quad \text{with } c = l/(2nL_{\max}) \\
&\leqslant \epsilon^{-1}(f(x_0) - f^*)e^{-\left|\log \frac{f(x_0)-f^*}{n\epsilon}\right|} \\
&= \eta e^{\log \frac{(f(x_0)-f^*)}{n\epsilon}} e^{-\left|\log \frac{f(x_0)-f^*}{n\epsilon}\right|} \\
&\leqslant \eta,
\end{aligned}$$

where the second inequality applies (2.12), the third inequality uses the definition of j (2.15), and the second last inequality uses the inequality $(1-c)^{1/c} \leqslant e^{-1}\ \forall\, c \in (0,1)$, which proves the essentially strongly convex case. Similarly, the general convex case is proven by

$$\mathbb{P}(f(x_j) - f^* \geqslant \epsilon) \leqslant \epsilon^{-1}\mathbb{E}(f(x_j) - f^*) \leqslant \frac{f(x_0) - f^*}{\epsilon\left(1 + j\frac{f(x_0)-f^*}{4nL_{\max}R^2}\right)} \leqslant \eta,$$

where the second inequality uses (2.13) and the last inequality uses the definition of j (2.16). □

Proofs for Constrained Case

This section uses the same definitions for $\mathbb{E}$ and $\mathbb{E}_{i(j)}$ as in Section 2.8. Before proving Theorem 2.4, we introduce notation and several preliminary results. Denote

$$(\Delta_j)_{i(j)} := (x_j - x_{j+1})_{i(j)}, \tag{2.43}$$

and formulate the update in Step 4 of Algorithm 1 in the following way:

$$x_{j+1} = \arg\min_{x\in\Omega} \langle \nabla_{i(j)} f(x_{k(j)}), (x - x_j)_{i(j)} \rangle + \frac{L_{\max}}{2\gamma}\|x - x_j\|^2.$$

(Note that $(x_{j+1})_i = (x_j)_i$ for $i \neq i(j)$.) From the optimality condition for this formulation, we have

$$\left\langle (x - x_{j+1})_{i(j)}, \nabla_{i(j)} f(x_{k(j)}) - \frac{L_{\max}}{\gamma}(\Delta_j)_{i(j)} \right\rangle \geqslant 0, \quad \text{for all } x \in \Omega.$$

This implies in particular that for all $x \in \Omega$, we have

$$\left\langle (\mathcal{P}_S(x) - x_{j+1})_{i(j)}, \nabla_{i(j)} f(x_{k(j)}) \right\rangle \geqslant \frac{L_{\max}}{\gamma} \left\langle (\mathcal{P}_S(x) - x_{j+1})_{i(j)}, (\Delta_j)_{i(j)} \right\rangle. \tag{2.44}$$

From the definition of $L_{\max}$, and using the notation (2.43), we have

$$f(x_{j+1}) \leqslant f(x_j) + \langle \nabla_{i(j)} f(x_j), -(\Delta_j)_{i(j)} \rangle + \frac{L_{\max}}{2} \|(\Delta_j)_{i(j)}\|^2,$$

which indicates that

$$\langle \nabla_{i(j)} f(x_j), (\Delta_j)_{i(j)} \rangle \leqslant f(x_j) - f(x_{j+1}) + \frac{L_{\max}}{2} \|(\Delta_j)_{i(j)}\|^2. \tag{2.45}$$

From optimality conditions for this defintion, we have

$$\left\langle x - \bar{x}_{j+1}, \nabla f(x_{k(j)}) + \frac{L_{\max}}{\gamma} (\bar{x}_{j+1} - x_j) \right\rangle \geqslant 0 \quad \forall x \in \Omega. \tag{2.46}$$

We now define $\Delta_j := x_j - \bar{x}_{j+1}$, and note that this definition is consistent with $(\Delta)_{i(j)}$ defined in (2.43). It can be seen that

$$\mathbb{E}_{i(j)}(\|x_{j+1} - x_j\|^2) = \frac{1}{n} \|\bar{x}_{j+1} - x_j\|^2.$$

Proof of Theorem 2.4

Proof. We prove (2.20) by induction. First, note that for any vectors a and b, we have

$$\|a\|^2 - \|b\|^2 = 2\|a\|^2 - (\|a\|^2 + \|b\|^2) \leqslant 2\|a\|^2 - 2\langle a, b \rangle \leqslant 2\langle a, a - b \rangle \leqslant 2\|a\|\|a - b\|,$$

Thus for all j, we have

$$\|x_{j-1} - \bar{x}_j\|^2 - \|x_j - \bar{x}_{j+1}\|^2 \leqslant 2\|x_{j-1} - \bar{x}_j\| \|x_j - \bar{x}_{j+1} - x_{j-1} + \bar{x}_j\|. \tag{2.47}$$

The second factor in the r.h.s. of (2.47) is bounded as follows:

$$
\begin{aligned}
&\|x_j - \bar{x}_{j+1} - x_{j-1} + \bar{x}_j\| \\
&\quad = \left\| x_j - \mathcal{P}_\Omega(x_j - \frac{\gamma}{L_{\max}}\nabla f(x_{k(j)})) - (x_{j-1} - \mathcal{P}_\Omega(x_{j-1} - \frac{\gamma}{L_{\max}}\nabla f(x_{k(j-1)}))) \right\| \\
&\quad \leqslant \left\| x_j - \frac{\gamma}{L_{\max}}\nabla f(x_{k(j)}) - \mathcal{P}_\Omega(x_j - \frac{\gamma}{L_{\max}}\nabla f(x_{k(j)})) - (x_{j-1} - \frac{\gamma}{L_{\max}}\nabla f(x_{k(j-1)}) \right. \\
&\qquad \left. -\mathcal{P}_\Omega(x_{j-1} - \frac{\gamma}{L_{\max}}\nabla f(x_{k(j-1)}))) \right\| + \frac{\gamma}{L_{\max}} \left\| \nabla f(x_{k(j-1)}) - \nabla f(x_{k(j)}) \right\| \\
&\quad \leqslant \left\| x_j - \frac{\gamma}{L_{\max}}\nabla f(x_{k(j)}) - x_{j-1} + \frac{\gamma}{L_{\max}}\nabla f(x_{k(j-1)}) \right\| \\
&\qquad + \frac{\gamma}{L_{\max}} \left\| \nabla f(x_{k(j-1)}) - \nabla f(x_{k(j)}) \right\| \\
&\quad \leqslant \|x_j - x_{j-1}\| + 2\frac{\gamma}{L_{\max}} \left\| \nabla f(x_{k(j)}) - \nabla f(x_{k(j-1)}) \right\| \\
&\quad \leqslant \|x_j - x_{j-1}\| + 2\frac{\gamma}{L_{\max}} \sum_{d=\min\{k(j-1),k(j)\}}^{\max\{k(j-1),k(j)\}-1} \|\nabla f(x_d) - \nabla f(x_{d+1})\| \\
&\quad \leqslant \|x_j - x_{j-1}\| + 2\frac{\gamma L_{\text{res}}}{L_{\max}} \sum_{d=\min\{k(j-1),k(j)\}}^{\max\{k(j-1),k(j)\}-1} \|x_d - x_{d+1}\|, \qquad (2.48)
\end{aligned}
$$

where the first inequality follows by adding and subtracting a term, and the second inequality uses the nonexpansive property of projection:

$$\|(z - \mathcal{P}_\Omega(z)) - (y - \mathcal{P}_\Omega(y))\| \leqslant \|z - y\|.$$

One can see that $j-1-\tau \leqslant k(j-1) \leqslant j-1$ and $j-\tau \leqslant k(j) \leqslant j$, which implies that $j-1-\tau \leqslant d \leqslant j-1$ for each index d in the summation in (2.48). It also follows that

$$\max\{k(j-1), k(j)\} - 1 - \min\{k(j-1), k(j)\} \leqslant \tau. \qquad (2.49)$$

We set $j = 1$, and note that $k(0) = 0$ and $k(1) \leqslant 1$. Thus, in this case, we have that the lower and upper limits of the summation in (2.48) are 0 and 0, respectively. Thus, this summation is vacuous, and we have

$$\|x_1 - \bar{x}_2 + x_0 - \bar{x}_1\| \leqslant \left(1 + 2\frac{\gamma L_{\text{res}}}{L_{\max}}\right) \|x_1 - x_0\|,$$

By substituting this bound in (2.47) and setting $j = 1$, we obtain

$$\mathbb{E}(\|x_0 - \bar{x}_1\|^2) - \mathbb{E}(\|x_1 - \bar{x}_2\|^2) \leqslant \left(2 + 4\frac{\gamma L_{\text{res}}}{L_{\max}}\right) \mathbb{E}(\|x_1 - x_0\| \|\bar{x}_1 - x_0\|). \qquad (2.50)$$

For any j, we have

$$\begin{aligned}\mathbb{E}(\|x_j - x_{j-1}\|\|\bar{x}_j - x_{j-1}\|) &\leqslant \frac{1}{2}\mathbb{E}(n^{1/2}\|x_j - x_{j-1}\|^2 + n^{-1/2}\|\bar{x}_j - x_{j-1}\|^2) \\ &= \frac{1}{2}\mathbb{E}(n^{1/2}\mathbb{E}_{i(j-1)}(\|x_j - x_{j-1}\|^2) + n^{-1/2}\|\bar{x}_j - x_{j-1}\|^2) \\ &= \frac{1}{2}\mathbb{E}(n^{-1/2}\|\bar{x}_j - x_{j-1}\|^2 + n^{-1/2}\|\bar{x}_j - x_{j-1}\|^2) \\ &= n^{-1/2}\mathbb{E}\|\bar{x}_j - x_{j-1}\|^2. \end{aligned} \tag{2.51}$$

Returning to (2.50), we have

$$\mathbb{E}(\|x_0 - \bar{x}_1\|^2) - \mathbb{E}(\|x_1 - \bar{x}_2\|^2) \leqslant 2n^{-1/2}\mathbb{E}\|\bar{x}_1 - x_0\|^2$$

which implies that

$$\mathbb{E}(\|x_0 - \bar{x}_1\|^2) \leqslant \left(1 - \frac{2}{\sqrt{n}} - \frac{4\gamma L_{\text{res}}}{\sqrt{n}L_{\max}}\right)^{-1} \mathbb{E}(\|x_1 - \bar{x}_2\|^2) \leqslant \rho\mathbb{E}(\|x_1 - \bar{x}_2\|^2).$$

To see the last inequality above, we only need to verify that

$$\gamma \leqslant \left(1 - \rho^{-1} - \frac{2}{\sqrt{n}}\right)\frac{\sqrt{n}L_{\max}}{4L_{\text{res}}}.$$

This proves that (2.20) holds for $j = 1$.

To take the inductive step, we assume that show that (2.20) holds up to index $j-1$. We have for $j - 1 - \tau \leqslant d \leqslant j - 2$ that

$$\begin{aligned}\mathbb{E}(\|x_d - x_{d+1}\|\|\bar{x}_j - x_{j-1}\|) &\leqslant \frac{1}{2}\mathbb{E}(n^{1/2}\|x_d - x_{d+1}\|^2 + n^{-1/2}\|\bar{x}_j - x_{j-1}\|^2) \\ &= \frac{1}{2}\mathbb{E}(n^{1/2}\mathbb{E}_{i(d)}(\|x_d - x_{d+1}\|^2) + n^{-1/2}\|\bar{x}_j - x_{j-1}\|^2) \\ &= \frac{1}{2}\mathbb{E}(n^{-1/2}\|x_d - \bar{x}_{d+1}\|^2 + n^{-1/2}\|\bar{x}_j - x_{j-1}\|^2) \\ &\leqslant \frac{1}{2}\mathbb{E}(n^{-1/2}\rho^{\tau}\|x_{j-1} - \bar{x}_j\|^2 + n^{-1/2}\|\bar{x}_j - x_{j-1}\|^2) \\ &\leqslant \frac{\rho^{\tau}}{n^{1/2}}\mathbb{E}(\|\bar{x}_j - x_{j-1}\|^2), \end{aligned} \tag{2.52}$$

where the second inequality uses the inductive hypothesis. By substituting (2.48) into (2.47) and

taking expectation on both sides of (2.47), we obtain

$$\begin{aligned}
&\mathbb{E}(\|x_{j-1}-\bar{x}_j\|^2)-\mathbb{E}(\|x_j-\bar{x}_{j+1}\|^2)\\
&\quad\leqslant 2\mathbb{E}(\|\bar{x}_j-x_{j-1}\|\|\bar{x}_j-\bar{x}_{j+1}+x_j-x_{j-1}\|)\\
&\quad\leqslant 2\mathbb{E}\left(\|\bar{x}_j-x_{j-1}\|\left(\|x_j-x_{j-1}\|+2\frac{\gamma L_{res}}{L_{max}}\sum_{d=\min\{k(j-1),k(j)\}}^{\max\{k(j-1),k(j)\}-1}\|x_d-x_{d+1}\|\right)\right)\\
&\quad= 2\mathbb{E}(\|\bar{x}_j-x_{j-1}\|\|x_j-x_{j-1}\|)+\\
&\qquad 4\frac{\gamma L_{res}}{L_{max}}\sum_{d=\min\{k(j-1),k(j)\}}^{\max\{k(j-1),k(j)\}-1}\mathbb{E}(\|\bar{x}_j-x_{j-1}\|\|x_d-x_{d+1}\|)\\
&\quad\leqslant n^{-1/2}\left(2+\frac{4\gamma L_{res}\tau\rho^\tau}{L_{max}}\right)\mathbb{E}(\|x_{j-1}-\bar{x}_j\|^2),
\end{aligned}$$

where the last line uses (2.49), (2.51), and (2.52). It follows that

$$\mathbb{E}(\|x_{j-1}-\bar{x}_j\|^2)\leqslant\left(1-n^{-1/2}\left(2+\frac{4\gamma L_{res}\tau\rho^\tau}{L_{max}}\right)\right)^{-1}\mathbb{E}(\|x_j-\bar{x}_{j+1}\|^2)\leqslant\rho\mathbb{E}(\|x_j-\bar{x}_{j+1}\|^2).$$

To see the last inequality, one only needs to verify that

$$\rho^{-1}\leqslant 1-\frac{1}{\sqrt{n}}\left(2+\frac{4\gamma L_{res}\tau\rho^\tau}{L_{max}}\right)\ \Leftrightarrow\ \gamma\leqslant\left(1-\rho^{-1}-\frac{2}{\sqrt{n}}\right)\frac{\sqrt{n}L_{max}}{4L_{res}\tau\rho^\tau},$$

and the last inequality is true because of the upper bound of γ in (2.19). It proves (2.20).

Next we will show the expectation of objective is monotonically decreasing. We have using the definition (2.43) that

$$\begin{aligned}
&\mathbb{E}_{i(j)}(f(x_{j+1}))=n^{-1}\sum_{i=1}^n f(x_j+(\Delta_j)_i)\\
&\quad\leqslant n^{-1}\sum_{i=1}^n\left[f(x_j)+\langle\nabla_i f(x_j),(\bar{x}_{j+1}-x_j)_i\rangle+\frac{L_{max}}{2}\|(x_{j+1}-x_j)_i\|^2\right]\\
&\quad= f(x_j)+n^{-1}\left(\langle\nabla f(x_j),\bar{x}_{j+1}-x_j\rangle+\frac{L_{max}}{2}\|\bar{x}_{j+1}-x_j\|^2\right)\\
&\quad= f(x_j)+\frac{1}{n}\left(\langle\nabla f(x_{k(j)}),\bar{x}_{j+1}-x_j\rangle+\frac{L_{max}}{2}\|\bar{x}_{j+1}-x_j\|^2\right)+\frac{1}{n}\langle\nabla f(x_j)-\nabla f(x_{k(j)}),\bar{x}_{j+1}-x_j\rangle\\
&\quad\leqslant f(x_j)+\frac{1}{n}\left(\frac{L_{max}}{2}\|\bar{x}_{j+1}-x_j\|^2-\frac{L_{max}}{\gamma}\|\bar{x}_{j+1}-x_j\|^2\right)+\frac{1}{n}\langle\nabla f(x_j)-\nabla f(x_{k(j)}),\bar{x}_{j+1}-x_j\rangle\\
&\quad= f(x_j)-\left(\frac{1}{\gamma}-\frac{1}{2}\right)\frac{L_{max}}{n}\|\bar{x}_{j+1}-x_j\|^2+\frac{1}{n}\langle\nabla f(x_j)-\nabla f(x_{k(j)}),\bar{x}_{j+1}-x_j\rangle,
\end{aligned}\tag{2.53}$$

where the second inequality uses (2.46). Consider the expectation of the last term on the right-hand

side of this expression. We have

$$\begin{aligned}
&\mathbb{E}\langle \nabla f(x_j) - \nabla f(x_{k(j)}), \bar{x}_{j+1} - x_j \rangle \\
&\quad \leqslant \mathbb{E}\|\nabla f(x_j) - \nabla f(x_{k(j)})\|\|\bar{x}_{j+1} - x_j\| \\
&\quad \leqslant \mathbb{E} \sum_{d=k(j)}^{j-1} \|\nabla f(x_d) - \nabla f(x_{d+1})\|\|\bar{x}_{j+1} - x_j\| \\
&\quad \leqslant L_{\text{res}}\mathbb{E} \sum_{d=k(j)}^{j-1} \|x_d - x_{d+1}\|\|\bar{x}_{j+1} - x_j\| \\
&\quad \leqslant \frac{L_{\text{res}}}{2}\mathbb{E} \sum_{d=k(j)}^{j-1} (n^{1/2}\|x_d - x_{d+1}\|^2 + n^{-1/2}\|\bar{x}_{j+1} - x_j\|^2) \\
&\quad = \frac{L_{\text{res}}}{2}\mathbb{E} \sum_{d=k(j)}^{j-1} (n^{1/2}\mathbb{E}_{i(d)}(\|x_d - x_{d+1}\|^2) + n^{-1/2}\|\bar{x}_{j+1} - x_j\|^2) \\
&\quad = \frac{L_{\text{res}}}{2}\mathbb{E} \sum_{d=k(j)}^{j-1} (n^{-1/2}\|x_d - \bar{x}_{d+1}\|^2 + n^{-1/2}\|\bar{x}_{j+1} - x_j\|^2) \\
&\quad \leqslant \frac{L_{\text{res}}}{2n^{1/2}}\mathbb{E} \sum_{d=k(j)}^{j-1} (1+\rho^\tau)\|\bar{x}_{j+1} - x_j\|^2 \\
&\quad \leqslant \frac{L_{\text{res}}\tau\rho^\tau}{n^{1/2}}\mathbb{E}\|\bar{x}_{j+1} - x_j\|^2, \qquad (2.54)
\end{aligned}$$

where the fifth inequality uses (2.20). By taking expectation on both sides of (2.53) and sustituting (2.54), we have

$$\mathbb{E}(f(x_{j+1})) \leqslant \mathbb{E}(f(x_j)) - \frac{1}{n}\left(\left(\frac{1}{\gamma} - \frac{1}{2}\right)L_{\max} - \frac{L_{\text{res}}\tau\rho^\tau}{n^{1/2}}\right)\mathbb{E}\|\bar{x}_{j+1} - x_j\|^2.$$

To see $\left(\frac{1}{\gamma} - \frac{1}{2}\right)L_{\max} - \frac{L_{\text{res}}\tau\rho^\tau}{n^{1/2}} \geqslant 0$, we only need to verify

$$\gamma \leqslant \left(\frac{1}{2} + \frac{L_{\text{res}}\tau\rho^\tau}{\sqrt{n}L_{\max}}\right)^{-1}$$

which is implied by the first upper bound of γ (2.19). Therefore, we have proved the monotonicity $\mathbb{E}(f(x_{j+1})) \leqslant \mathbb{E}(f(x_j))$.

Next we prove the sublinear convergence rate for the constrained smooth convex case in (2.22).

We have

$$
\begin{aligned}
&\|x_{j+1} - \mathcal{P}_S(x_{j+1})\|^2 \leqslant \|x_{j+1} - \mathcal{P}_S(x_j)\|^2 \\
&\quad = \|x_j - (\Delta_j)_{i(j)} e_{i(j)} - \mathcal{P}_S(x_j)\|^2 \\
&\quad = \|x_j - \mathcal{P}_S(x_j)\|^2 + |(\Delta_j)_{i(j)}|^2 - 2(x_j - \mathcal{P}_S(x_j))_{i(j)}(\Delta_j)_{i(j)} \\
&\quad = \|x_j - \mathcal{P}_S(x_j)\|^2 - |(\Delta_j)_{i(j)}|^2 - 2\left((x_j - \mathcal{P}_S(x_j))_{i(j)} - (\Delta_j)_{i(j)}\right)(\Delta_j)_{i(j)} \\
&\quad = \|x_j - \mathcal{P}_S(x_j)\|^2 - \|(\Delta_j)_{i(j)}\|^2 + 2(\mathcal{P}_S(x_j) - x_{j+1})_{i(j)}(\Delta_j)_{i(j)} \\
&\quad \leqslant \|x_j - \mathcal{P}_S(x_j)\|^2 - |(\Delta_j)_{i(j)}|^2 + \frac{2\gamma}{L_{\max}}(\mathcal{P}_S(x_j) - x_{j+1})_{i(j)} \nabla_{i(j)} f(x_{k(j)}) \\
&\quad = \|x_j - \mathcal{P}_S(x_j)\|^2 - |(\Delta_j)_{i(j)}|^2 + \frac{2\gamma}{L_{\max}}(\mathcal{P}_S(x_j) - x_j)_{i(j)} \nabla_{i(j)} f(x_{k(j)}) + \\
&\quad \frac{2\gamma}{L_{\max}}\left((\Delta_j)_{i(j)} \nabla_{i(j)} f(x_j) + (\Delta_j)_{i(j)}\left(\nabla_{i(j)} f(x_{k(j)}) - \nabla_{i(j)} f(x_j)\right)\right) \\
&\quad \leqslant \|x_j - \mathcal{P}_S(x_j)\|^2 - |(\Delta_j)_{i(j)}|^2 + \frac{2\gamma}{L_{\max}}(\mathcal{P}_S(x_j) - x_j)_{i(j)} \nabla_{i(j)} f(x_{k(j)}) + \\
&\quad \frac{2\gamma}{L_{\max}}\Bigg(f(x_j) - f(x_{j+1}) + \frac{L_{\max}}{2}|(\Delta_j)_{i(j)}|^2 \\
&\quad + (\Delta_j)_{i(j)}\left(\nabla_{i(j)} f(x_{k(j)}) - \nabla_{i(j)} f(x_j)\right)\Bigg) \\
&\quad = \|x_j - \mathcal{P}_S(x_j)\|^2 - (1-\gamma)|(\Delta_j)_{i(j)}|^2 + \frac{2\gamma}{L_{\max}} \underbrace{(\mathcal{P}_S(x_j) - x_j)_{i(j)} \nabla_{i(j)} f(x_{k(j)})}_{T_1} + \\
&\quad \frac{2\gamma}{L_{\max}}(f(x_j) - f(x_{j+1})) + \frac{2\gamma}{L_{\max}} \underbrace{(\Delta_j)_{i(j)}\left(\nabla_{i(j)} f(x_{k(j)}) - \nabla_{i(j)} f(x_j)\right)}_{T_2},
\end{aligned}
\tag{2.55}
$$

where the second inequality uses (2.44) and the third inequality uses (2.45). We now seek upper

bounds on the quantities T_1 and T_2 in the expectation sense. For T_1, we have

$$\begin{aligned}
\mathbb{E}(T_1) &= n^{-1}\mathbb{E}\langle \mathcal{P}_S(x_j) - x_j, \nabla f(x_{k(j)})\rangle \\
&= n^{-1}\mathbb{E}\langle \mathcal{P}_S(x_j) - x_{k(j)}, \nabla f(x_{k(j)})\rangle + n^{-1}\mathbb{E}\sum_{d=k(j)}^{j-1}\langle x_d - x_{d+1}, \nabla f(x_{k(j)})\rangle \\
&= n^{-1}\mathbb{E}\langle \mathcal{P}_S(x_j) - x_{k(j)}, \nabla f(x_{k(j)})\rangle \\
&\quad + n^{-1}\mathbb{E}\sum_{d=k(j)}^{j-1}\langle x_d - x_{d+1}, \nabla f(x_d)\rangle + \langle x_d - x_{d+1}, \nabla f(x_{k(j)}) - \nabla f(x_d)\rangle \\
&\leqslant n^{-1}\mathbb{E}(f^* - f(x_{k(j)})) + n^{-1}\mathbb{E}\sum_{d=k(j)}^{j-1}\left(f(x_d) - f(x_{d+1}) + \frac{L_{\max}}{2}\|x_d - x_{d+1}\|^2\right) \\
&\quad + n^{-1}\mathbb{E}\sum_{d=k(j)}^{j-1}\langle x_d - x_{d+1}, \nabla f(x_{k(j)}) - \nabla f(x_d)\rangle \\
&= n^{-1}\mathbb{E}(f^* - f(x_j)) + \frac{L_{\max}}{2n}\mathbb{E}\sum_{d=k(j)}^{j-1}\|x_d - x_{d+1}\|^2 \\
&\quad + n^{-1}\mathbb{E}\sum_{d=k(j)}^{j-1}\langle x_d - x_{d+1}, \nabla f(x_{k(j)}) - \nabla f(x_d)\rangle \\
&= n^{-1}\mathbb{E}(f^* - f(x_j)) + \frac{L_{\max}}{2n^2}\mathbb{E}\sum_{d=k(j)}^{j-1}\|x_d - \bar{x}_{d+1}\|^2 \\
&\quad + n^{-1}\mathbb{E}\sum_{d=k(j)}^{j-1}\langle x_d - x_{d+1}, \nabla f(x_{k(j)}) - \nabla f(x_d)\rangle \\
&\leqslant n^{-1}\mathbb{E}(f^* - f(x_j)) + \frac{L_{\max}\tau\rho^\tau}{2n^2}\mathbb{E}\|x_j - \bar{x}_{j+1}\|^2 \\
&\quad + n^{-1}\sum_{d=k(j)}^{j-1}\underbrace{\mathbb{E}\langle x_d - x_{d+1}, \nabla f(x_{k(j)}) - \nabla f(x_d)\rangle}_{T_3},
\end{aligned}$$

where the last inequality uses (2.20). The upper bound of $\mathbb{E}(T_3)$ is estimated by

$$\begin{aligned}
\mathbb{E}(T_3) &= \mathbb{E}\langle x_d - x_{d+1}, \nabla f(x_{k(j)}) - \nabla f(x_d)\rangle \\
&= \mathbb{E}(\mathbb{E}_{i(d)}\langle x_d - x_{d+1}, \nabla f(x_{k(j)}) - \nabla f(x_d)\rangle) \\
&= n^{-1}\mathbb{E}\langle x_d - \bar{x}_{d+1}, \nabla f(x_{k(j)}) - \nabla f(x_d)\rangle \\
&\leqslant n^{-1}\mathbb{E}\|x_d - \bar{x}_{d+1}\|\|\nabla f(x_{k(j)}) - \nabla f(x_d)\| \\
&\leqslant n^{-1}\mathbb{E}(\|x_d - \bar{x}_{d+1}\| \sum_{t=k(j)}^{d-1} \|\nabla f(x_t) - \nabla f(x_{t+1})\|) \\
&\leqslant \frac{L_{res}}{n} \sum_{t=k(j)}^{d-1} \mathbb{E}(\|x_d - \bar{x}_{d+1}\|\|x_t - x_{t+1}\|) \\
&\leqslant \frac{L_{res}}{2n} \sum_{t=k(j)}^{d-1} \mathbb{E}(n^{-1/2}\|x_d - \bar{x}_{d+1}\|^2 + n^{1/2}\|x_t - x_{t+1}\|^2) \\
&\leqslant \frac{L_{res}}{2n} \sum_{t=k(j)}^{d-1} \mathbb{E}(n^{-1/2}\|x_d - \bar{x}_{d+1}\|^2 + n^{-1/2}\|x_t - \bar{x}_{t+1}\|^2) \\
&\leqslant \frac{L_{res}\rho^\tau}{n^{3/2}} \sum_{t=k(j)}^{d-1} \mathbb{E}(\|x_j - \bar{x}_{j+1}\|^2) \\
&\leqslant \frac{L_{res}\tau\rho^\tau}{n^{3/2}}\mathbb{E}(\|x_j - \bar{x}_{j+1}\|^2).
\end{aligned}$$

where the second last inequality uses (2.20). Therefore, $\mathbb{E}(T_1)$ can be bounded by

$$\begin{aligned}
\mathbb{E}(T_1) &= \mathbb{E}\langle(\mathcal{P}_S(x_j) - x_j)_{i(j)}, \nabla_{i(j)} f(x_{k(j)})\rangle \\
&\leqslant \frac{1}{n}\mathbb{E}(f^* - f(x_j)) + \frac{L_{max}\tau\rho^\tau}{2n^2}\mathbb{E}\|x_j - \bar{x}_{j+1}\|^2 + \sum_{d=k(j)}^{j-1} \frac{L_{res}\tau\rho^\tau}{n^{5/2}}\mathbb{E}(\|x_j - \bar{x}_{j+1}\|^2) \\
&= \frac{1}{n}\left(f^* - \mathbb{E}f(x_j) + \left(\frac{L_{max}\tau\rho^\tau}{2n} + \frac{L_{res}\tau^2\rho^\tau}{n^{3/2}}\right)\mathbb{E}(\|x_j - \bar{x}_{j+1}\|^2)\right). \qquad (2.56)
\end{aligned}$$

For T_2, we have

$$\begin{aligned}
\mathbb{E}(T_2) &= \mathbb{E}(\Delta_j)_{i(j)} \left(\nabla_{i(j)} f(x_{k(j)}) - \nabla_{i(j)} f(x_j)\right) \\
&= n^{-1}\mathbb{E}\langle \Delta_j, \nabla f(x_{k(j)}) - \nabla f(x_j)\rangle \\
&\leqslant n^{-1}\mathbb{E}(\|\Delta_j\|\|\nabla f(x_{k(j)}) - \nabla f(x_j)\|) \\
&\leqslant n^{-1}\mathbb{E}\left(\sum_{d=k(j)}^{j-1} \|\Delta_j\|\|\nabla f(x_d) - \nabla f(x_{d+1})\|\right) \\
&\leqslant \frac{L_{res}}{n}\mathbb{E}\left(\sum_{d=k(j)}^{j-1} \|\Delta_j\|\|x_d - x_{d+1}\|\right) \\
&= \frac{L_{res}}{2n}\mathbb{E}\left(\sum_{d=k(j)}^{j-1} n^{-1/2}\|\Delta_j\|^2 + n^{1/2}\|x_d - x_{d+1}\|^2\right) \\
&= \frac{L_{res}}{2n}\mathbb{E}\left(\sum_{d=k(j)}^{j-1} n^{-1/2}\|x_j - \bar{x}_{j+1}\|^2 + n^{1/2}\mathbb{E}_{i(d)}\|x_d - x_{d+1}\|^2\right) \\
&= \frac{L_{res}}{2n}\mathbb{E}\left(\sum_{d=k(j)}^{j-1} n^{-1/2}\|x_j - \bar{x}_{j+1}\|^2 + n^{-1/2}\|x_d - \bar{x}_{d+1}\|^2\right) \\
&= \frac{L_{res}}{2n^{3/2}}\left(\sum_{d=k(j)}^{j-1} \mathbb{E}\|x_j - \bar{x}_{j+1}\|^2 + \mathbb{E}\|x_d - \bar{x}_{d+1}\|^2\right) \\
&\leqslant \frac{L_{res}(1+\rho^\tau)}{2n^{3/2}} \sum_{d=k(j)}^{j-1} \mathbb{E}\|x_j - \bar{x}_{j+1}\|^2 \\
&\leqslant \frac{L_{res}\tau\rho^\tau}{n^{3/2}}\mathbb{E}\|x_j - \bar{x}_{j+1}\|^2, \qquad (2.57)
\end{aligned}$$

where the second last inequality uses (2.20).

By taking the expectation on both sides of (2.55), using $\mathbb{E}_{i(j)}(|(\Delta_j)_{i(j)}|^2) = n^{-1}\|x_j - \bar{x}_{j+1}\|^2$, and substituting the upper bounds from (2.56) and (2.57), we obtain

$$\begin{aligned}
\mathbb{E}\|x_{j+1} - \mathcal{P}_S(x_{j+1})\|^2 &\leqslant \mathbb{E}\|x_j - \mathcal{P}_S(x_j)\|^2 \\
&\quad - \frac{1}{n}\left(1 - \gamma - \frac{2\gamma L_{res}\tau\rho^\tau}{L_{max}n^{1/2}} - \frac{\gamma\tau\rho^\tau}{n} - \frac{2\gamma L_{res}\tau^2\rho^\tau}{L_{max}n^{3/2}}\right)\mathbb{E}\|x_j - \bar{x}_{j+1}\|^2 \\
&\quad + \frac{2\gamma}{L_{max}n}(f^* - \mathbb{E}f(x_j)) + \frac{2\gamma}{L_{max}}(\mathbb{E}f(x_j) - \mathbb{E}f(x_{j+1})) \\
&\leqslant \mathbb{E}\|x_j - \mathcal{P}_S(x_j)\|^2 + \frac{2\gamma}{L_{max}n}(f^* - \mathbb{E}f(x_j)) + \frac{2\gamma}{L_{max}}(\mathbb{E}f(x_j) - \mathbb{E}f(x_{j+1})). \qquad (2.58)
\end{aligned}$$

In the second inequality, we were able to drop the term involving $\mathbb{E}\|x_j - \bar{x}_{j+1}\|^2$ by using the fact

that

$$1-\gamma-\frac{2\gamma L_{\text{res}}\tau\rho^{\tau}}{L_{\max}n^{1/2}}-\frac{\gamma\tau\rho^{\tau}}{n}-\frac{2\gamma L_{\text{res}}\tau^{2}\rho^{\tau}}{L_{\max}n^{3/2}}=1-\gamma\psi\geqslant 0,$$

which follows from the definition (2.18) of ψ and from the first upper bound on γ in (2.19). It follows that

$$\begin{aligned}
&\mathbb{E}\|x_{j+1}-\mathcal{P}_S(x_{j+1})\|^2+\frac{2\gamma}{L_{\max}}(\mathbb{E}f(x_{j+1})-f^*)\\
&\quad\leqslant \mathbb{E}\|x_j-\mathcal{P}_S(x_j)\|^2+\frac{2\gamma}{L_{\max}}(\mathbb{E}f(x_j)-f^*)-\frac{2\gamma}{L_{\max}n}(\mathbb{E}f(x_j)-f^*)\\
&\quad\leqslant \|x_0-\mathcal{P}_S(x_0)\|^2+\frac{2\gamma}{L_{\max}}(f(x_0)-f^*)-\frac{2\gamma}{L_{\max}n}\sum_{t=0}^{j}(\mathbb{E}f(x_t)-f^*)\\
&\quad\leqslant \|x_0-\mathcal{P}_S(x_0)\|^2+\frac{2\gamma}{L_{\max}}(f(x_0)-f^*)-\frac{2\gamma(j+1)}{L_{\max}n}(\mathbb{E}f(x_{j+1})-f^*),
\end{aligned}\tag{2.59}$$

where the second inequality follows by applying induction to the inequality

$$S_{j+1}\leqslant S_j-\frac{2\gamma}{L_{\max}n}\mathbb{E}(f(x_j)-f^*),$$

where

$$S_j:=\mathbb{E}(\|x_j-\mathcal{P}_S(x_j)\|^2)+\frac{2\gamma}{L_{\max}}\mathbb{E}(f(x_j)-\mathcal{P}_S(x_j)),$$

and the last line uses the monotonicity of $\mathbb{E}f(x_j)$ (proved above) and the definition of R_0. It implies that

$$\begin{aligned}
&\mathbb{E}\|x_{j+1}-\mathcal{P}_S(x_{j+1})\|^2+\frac{2\gamma}{L_{\max}}(\mathbb{E}f(x_{j+1})-f^*)+\frac{2\gamma(j+1)}{L_{\max}n}(\mathbb{E}f(x_{j+1})-f^*)\\
&\quad\leqslant R_0^2+\frac{2\gamma}{L_{\max}}(f(x_0)-f^*)\\
&\Rightarrow \frac{2\gamma(n+j+1)}{L_{\max}n}(\mathbb{E}f(x_{j+1})-f^*)\leqslant R_0^2+\frac{2\gamma}{L_{\max}}(f(x_0)-f^*)\\
&\Rightarrow \mathbb{E}f(x_{j+1})-f^*\leqslant\frac{n(R_0^2L_{\max}+2\gamma(f(x_0)-f^*))}{2\gamma(n+j+1)}.
\end{aligned}$$

This completes the proof of the sublinear convergence rate (2.22).

Finally, we prove the linear convergence rate (2.21) for the essentially strongly convex case. All bounds proven above hold, and we make use the following additional property:

$$f(x_j)-f^*\geqslant\langle\nabla f(\mathcal{P}_S(x_j)),x_j-\mathcal{P}_S(x_j)\rangle+\frac{l}{2}\|x_j-\mathcal{P}_S(x_j)\|^2\geqslant\frac{l}{2}\|x_j-\mathcal{P}_S(x_j)\|^2,$$

due to feasibility of x_j and $\langle\nabla f(\mathcal{P}_S(x_j)),x_j-\mathcal{P}_S(x_j)\rangle\geqslant 0$. By using this result together with some

elementary manipulation, we obtain

$$\begin{aligned} f(x_j) - f^* &= \left(1 - \frac{L_{\max}}{l\gamma + L_{\max}}\right)(f(x_j) - f^*) + \frac{L_{\max}}{l\gamma + L_{\max}}(f(x_j) - f^*) \\ &\geqslant \left(1 - \frac{L_{\max}}{l\gamma + L_{\max}}\right)(f(x_j) - f^*) + \frac{L_{\max} l}{2(l\gamma + L_{\max})}\|x_j - \mathcal{P}_S(x_j)\|^2 \\ &= \frac{L_{\max} l}{2(l\gamma + L_{\max})}\left(\|x_j - \mathcal{P}_S(x_j)\|^2 + \frac{2\gamma}{L_{\max}}(f(x_j) - f^*)\right). \end{aligned} \tag{2.60}$$

Recalling (2.59), we have

$$\begin{aligned} &\mathbb{E}\|x_{j+1} - \mathcal{P}_S(x_{j+1})\|^2 + \frac{2\gamma}{L_{\max}}(\mathbb{E}f(x_{j+1}) - f^*) \\ &\quad\leqslant \mathbb{E}\|x_j - \mathcal{P}_S(x_j)\|^2 + \frac{2\gamma}{L_{\max}}(\mathbb{E}f(x_j) - f^*) - \frac{2\gamma}{L_{\max} n}(\mathbb{E}f(x_j) - f^*). \end{aligned} \tag{2.61}$$

By taking the expectation of both sides in (2.60) and substituting in the last term of (2.61), we obtain

$$\begin{aligned} &\mathbb{E}\|x_{j+1} - \mathcal{P}_S(x_{j+1})\|^2 + \frac{2\gamma}{L_{\max}}(\mathbb{E}f(x_{j+1}) - f^*) \\ &\leqslant \mathbb{E}\|x_j - \mathcal{P}_S(x_j)\|^2 + \frac{2\gamma}{L_{\max}}(\mathbb{E}f(x_j) - f^*) \\ &\quad - \frac{2\gamma}{L_{\max} n}\left(\frac{L_{\max} l}{2(l\gamma + L_{\max})}\left(\mathbb{E}\|x_j - \mathcal{P}_S(x_j)\|^2 + \frac{2\gamma}{L_{\max}}(\mathbb{E}f(x_j) - f^*)\right)\right) \\ &= \left(1 - \frac{l}{n(l + \gamma^{-1}L_{\max})}\right)\left(\mathbb{E}\|x_j - \mathcal{P}_S(x_j)\|^2 + \frac{2\gamma}{L_{\max}}(\mathbb{E}f(x_j) - f^*)\right) \\ &\leqslant \left(1 - \frac{l}{n(l + \gamma^{-1}L_{\max})}\right)^{j+1}\left(\|x_0 - \mathcal{P}_S(x_0)\|^2 + \frac{2\gamma}{L_{\max}}(f(x_0) - f^*)\right), \end{aligned}$$

which yields (2.21). □

Proof of Corollary 2.5

Proof. To apply Theorem 2.4, we first show $\rho > \left(1 - \frac{2}{\sqrt{n}}\right)^{-1}$. Using the bound (2.23), together with $L_{\text{res}}/L_{\max} \geqslant 1$, we obtain

$$\begin{aligned} \left(1 - \frac{2}{\sqrt{n}}\right)\left(1 + \frac{4e\tau L_{\text{res}}}{\sqrt{n} L_{\max}}\right) &= \left(1 + \frac{4e\tau L_{\text{res}}}{\sqrt{n} L_{\max}}\right) - \left(1 + \frac{4e\tau L_{\text{res}}}{\sqrt{n} L_{\max}}\right)\frac{2}{\sqrt{n}} \\ &\geqslant \left(1 + \frac{4e\tau}{\sqrt{n}}\right) - \left(1 + \frac{1}{\tau + 1}\right)\frac{2}{\sqrt{n}} = 1 + \left(2e\tau - 1 - \frac{1}{\tau + 1}\right)\frac{2}{\sqrt{n}} > 1, \end{aligned}$$

where the last inequality uses $\tau \geqslant 1$. Note that for ρ defined by (2.24), and using (2.23), we have

$$\rho^\tau \leqslant \rho^{\tau+1} = \left(\left(1 + \frac{4e\tau L_{\text{res}}}{\sqrt{n} L_{\max}} \right)^{\frac{\sqrt{n} L_{\max}}{4e\tau L_{\text{res}}}} \right)^{\frac{4e\tau L_{\text{res}}(\tau+1)}{\sqrt{n} L_{\max}}} \leqslant e^{\frac{4e\tau L_{\text{res}}(\tau+1)}{\sqrt{n} L_{\max}}} \leqslant e.$$

Thus from the definition of ψ (2.18), we have that

$$\begin{aligned} \psi &= 1 + \frac{L_{\text{res}} \tau \rho^\tau}{\sqrt{n} L_{\max}} \left(2 + \frac{L_{\max}}{\sqrt{n} L_{\text{res}}} + \frac{2\tau}{n} \right) \leqslant 1 + \frac{L_{\text{res}} \tau \rho^\tau}{4eL_{\text{res}} \tau(\tau+1)} \left(2 + \frac{1}{\sqrt{n}} + \frac{2\tau}{n} \right) \\ &\leqslant 1 + \frac{1}{4(\tau+1)} \left(2 + \frac{1}{\sqrt{n}} + \frac{2\tau}{n} \right) \leqslant 1 + \left(\frac{1}{4} + \frac{1}{16} + \frac{1}{10} \right) \leqslant 2. \end{aligned} \tag{2.62}$$

(The second last inequality uses $n \geqslant 5$ and $\tau \geqslant 1$.) Thus, the steplength parameter choice $\gamma = 1/2$ satisfies the first bound in (2.19). To show that the second bound in (2.19) holds also, we have

$$\begin{aligned} \left(1 - \frac{1}{\rho} - \frac{2}{\sqrt{n}} \right) \frac{\sqrt{n} L_{\max}}{4L_{\text{res}} \tau \rho^\tau} &= \left(\frac{\rho - 1}{\rho} - \frac{2}{\sqrt{n}} \right) \frac{\sqrt{n} L_{\max}}{4L_{\text{res}} \tau \rho^\tau} \\ &= \frac{4e\tau L_{\text{res}}}{4L_{\text{res}} \tau \rho^{\tau+1}} - \frac{L_{\max}}{2L_{\text{res}} \tau \rho^\tau} \geqslant 1 - \frac{1}{2} = \frac{1}{2}. \end{aligned}$$

We can thus set $\gamma = 1/2$, and by substituting this choice into (2.21), we obtain (2.25). We obtain (2.26) by making the same substitution into (2.22). □

3 AN ASYNCHRONOUS PARALLEL STOCHASTIC PROXIMAL COORDINATE DESCENT ALGORITHM FOR COMPOSITE FUNCTIONS

3.1 Overview

We extend the AsySCD algorithm described in Chapter 2 for minimizing a composite objective function, which consists of a smooth convex function plus a separable convex function. In contrast to previous analysis in Chapter 2, the mode of asynchronous computation accounts for the fact that components of the unknown vector may be written by some cores simultaneously with being read by others (namely "inconsistent read"). Despite the complications arising from the "inconsistent read" model, the method achieves a linear convergence rate on functions that satisfy an optimal strong convexity property and a sublinear rate ($1/k$) on general convex functions. Near-linear speedup on a multicore system can be expected if the number of processors is $O(n^{1/4})$. The results are consistent with the analysis for constrained AsySCD in Chapter 2. We describe results from implementation on ten cores of a multicore processor. Please also refer to the original paper in Liu and Wright (2014).

3.2 Introduction

Consider the convex optimization problem

$$\min_x \quad F(x) := f(x) + g(x), \tag{3.1}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$ is a smooth convex function and $g : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$ is a separable, closed, convex, and extended real-valued function. "Separable" means that $g(x)$ can be expressed as $g(x) = \sum_{i=1}^n g_i((x)_i)$, where $(x)_i$ denotes the ith element of x and each $g_i : \mathbb{R} \mapsto \mathbb{R} \cup \{\infty\}$, $i = 1, 2, \ldots, n$ is a closed, convex, and extended real-valued function.

Formulations of the type (3.1) arise in many data analysis and machine learning problems, for example, the linear primal or nonlinear dual formulation of support vector machines (Cortes and Vapnik, 1995), the LASSO approach to regularized least squares, and regularized logistic regression.

This chapter proposes an asynchronous stochastic proximal coordinate-descent algorithm, called AsySPCD (an extension of AsySCD), for composite objective functions. The basic step of AsySPCD, executed repeatedly by each core of a multicore system, is as follows: Choose an index $i \in \{1, 2, \ldots, n\}$; read x from shared memory and evaluate the ith element of ∇f; subtract a short, constant, positive of this partial gradient from $(x)_i$; and perform a proximal operation on $(x)_i$ to account for the regularization term $g_i(\cdot)$. (Generally, we denote by $\hat{x}$ the version of x that is used by a core to evaluate its component of $\nabla f(\hat{x})$.) We assume, however, that indefinite delays do not occur between reading and updating: There is a bound τ such no more than τ component-wise

updates to x are missed by a core, between the time at which it reads the vector $\hat{x}$ and the time at which it makes its update to the chosen element of x. A similar model of parallel asynchronous computation was used in HOGWILD! (Niu et al., 2011) and ASYSCD in Chapter 2. However, there is a key difference in this chapter: *We do* not *assume that the evaluation vector* $\hat{x}$ *is a version of* x *that actually existed in the shared memory at some point in time. Recall that we assume that* $\hat{x}$ *is some early iterate of* x *in* ASYSCD. Rather, we account for the fact that the components of x may be updated by multiple cores while in the process of being read by another core, so that $\hat{x}$ may be a "hybrid" version that never actually existed in memory. Our new model, which we call an "inconsistent read" model, is significantly closer to the reality of asynchronous computation, and dispenses with the somewhat unsatisfying "consistent read" assumption of previous work. It also requires a quite distinct style of analysis; our proofs differ substantially from those in previous related works.

We show that, for suitable choices of steplength, our algorithm converges at a linear rate if an "optimal strong convexity" property (3.2) holds. It attains sublinear convergence at a "$1/k$" rate for general convex functions. Our analysis also defines a sufficient condition for near-linear speedup in the number of cores used. This condition relates the value of delay parameter τ (which corresponds closely to the number of cores / threads used in the computation) to the problem dimension n. (Here τ plays a similar role as the τ in Chapter 2. They can be roughly treated as the same quantity.) A parameter that quantifies the cross-coordinate interactions in ∇f also appears in this relationship. When the Hessian of f is nearly diagonal, the minimization problem is almost separable, so higher degrees of parallelism are possible.

Section 3.3 specifies the proposed algorithm. Convergence results are described in Section 3.4, with proofs given in the appendix. Computational experience is reported in Section 3.5. A summary and conclusions appear in Section 3.6.

Notation and Assumption

We use the same definitions for e_i, $\|\cdot\|$, $(x)_i$, $\nabla_i f(x)$, $(\nabla f(x))_i$, and $\partial f/\partial x_i$ as Section 2.2 and define the additional notations used in the remainder of the chapter below.

- Ω denotes the intersection of $\mathrm{dom}(f)$ and $\mathrm{dom}(g)$
- S denotes the set on which F attains its optimal value, which is denoted by F^*.
- $\mathcal{P}_S(\cdot)$ denotes Euclidean-norm projection onto S.
- Given a matrix A, we use $A_{\cdot j}$ to denote its jth column and $A_{i\cdot}$ to denote its ith row.
- $f_j^* := f(\mathcal{P}_S(x_j))$ and $g_j^* := f(\mathcal{P}_S(x_j))$.
- $F^* := F(\mathcal{P}_S(x))$ denotes the optimal objective value. (Note that $F^* = f_j^* + g_j^*$ for any j.)

- Given a scalar function $h: \mathbb{R} \to \mathbb{R}$, define the proximal operator

$$\mathcal{P}_{i,h}(y) := \arg\min_x \frac{1}{2}\|x - y\|^2 + h((x)_i).$$

  Similarly, for the vector function g, we denote

$$\mathcal{P}_g(y) := \arg\min_x \frac{1}{2}\|x - y\|^2 + g(x).$$

  Note that the proximal operator is a nonexpansive operator, that is, $\|\mathcal{P}_g(x) - \mathcal{P}_g(y)\| \leqslant \|x - y\|$.

We define the following *optimal strong convexity* condition for a convex function f with respect to the optimal set S, with parameter $l > 0$:

$$F(x) - F(\mathcal{P}_S(x)) \geqslant \frac{l}{2}\|x - \mathcal{P}_S(x)\|^2 \quad \forall x \in \Omega. \tag{3.2}$$

This condition is significantly weaker than the usual strong convexity condition. A strongly convex function $F(.)$ is an optimally strongly convex function, but the inverse is not true in general. We provide several examples of optimally strongly convex functions that are not strongly convex:

- $F(x) =$ constant.
- $F(x) = f(Ax)$, where f is a strongly convex function and A is any matrix, possibly one with a nontrivial kernel.
- $F(x) = f(Ax) + \mathbf{1}_X(x)$ with strongly convex f, and arbitrary A, where $\mathbf{1}_X(x)$ is an indicator function defined on a closed convex set X. Note first that $y^* := Ax^*$ is unique for any $x^* \in S$, from the strong convexity of f. The optimal solution set S is defined by

$$Ax = y^*, \quad x \in X.$$

  The inequality (3.2) clearly holds for $x \notin X$, since the left-hand side is infinite in this case. For $x \in X$, we have by the famous theorem of Hoffman (1952) (when specialized to the case of equalities only, and squaring both sides) that there exists $c > 0$ such that

$$\|Ax - y^*\|^2 = \|A(x - \mathcal{P}_S(x))\|^2 \geqslant c\|x - \mathcal{P}_S(x)\|^2.$$

  Then from the strong convexity of $f(x)$, we have that there exist a positive number l such

that for any $x \in X$

$$\begin{aligned} F(x) - F(\mathcal{P}_S(x)) &= f(Ax) - f(A\mathcal{P}_S(x)) \\ &\geqslant \langle \partial f(A\mathcal{P}_S(x)), A(x - \mathcal{P}_S(x))\rangle + \frac{l}{2}\|A(x - \mathcal{P}_S(x))\|^2 \\ &\geqslant \frac{l}{2}\|A(x - \mathcal{P}_S(x))\|^2 \geqslant \frac{lc}{2}\|x - \mathcal{P}_S(x)\|^2; \end{aligned}$$

- Squared hinge loss $F(x) = \sum_i \max(0, a_i^T x - y_i)^2$. To verify optimal strong convexity, we reformulate this problem as

$$\min_{t,x} \|t\|^2 \text{ subject to } t \geqslant 0,\ t_i \geqslant a_i^T x - y_i \ \ \forall_i,$$

  and apply the result just derived.

Note that optimal strong convexity (3.2) is a weaker version of the "essential strong convexity" condition used in Chapter 2. A concept called "restricted strong convexity" proposed in Lai and Yin (2013) (See Lemma 4.6) is similar in that it requires a certain quantity to increase quadratically with distance from the solution set, but different in that the objective is assumed to be differentiable. Anitescu (2000) defines a "quadratic growth condition" for (smooth) nonlinear programming in which the objective is assumed to grow at least quadratically with distance to a local solution in some feasible neighborhood of that solution. Since our setting (unconstrained, nonsmooth, convex) is quite different, we believe the use of a different term is warranted here.

Throughout this chapter, we make the following assumption.

Assumption 3.1. *The optimal solution set* S *of* (3.1) *is nonempty.*

Lipschitz Constants

We define two different Lipschitz constants L_{res} and L_{max} that are critical to the analysis, as follows. Their definitions are essentially the same as our definitions in Chapter 2, but expressed in a slightly different way for the convenience of proofs. L_{res} is the *restricted Lipschitz constant* for ∇f along the coordinate directions: For any $x \in \Omega$, for any $i = 1, 2, \ldots, n$, and any $t \in \mathbb{R}$ such that $x + te_i \in \Omega$, we have

$$\|\nabla f(x) - \nabla f(x + te_i)\| \leqslant L_{res}|t|.$$

The *coordinate Lipschitz constant* L_{max} is defined for x, i, t satisfying the same conditions as above:

$$\|\nabla f(x) - \nabla f(x + te_i)\|_\infty \leqslant L_{max}|t|.$$

Note that

$$f(x + te_i) - f(x) \leqslant \langle \nabla_i f(x),\ t\rangle + \frac{L_{max}}{2}t^2. \tag{3.3}$$

We denote the ratio between these two quantities by Λ:

$$\Lambda := L_{\text{res}}/L_{\max}. \tag{3.4}$$

Making the implicit assumption that L_{res} and $L_{\max}$ are chosen to be the smallest values that satisfy their respective definitions, we have from standard relationships between the ℓ_2 and ℓ_∞ norms that

$$1 \leqslant \Lambda \leqslant \sqrt{n}.$$

Besides bounding the nonlinearity of f along various directions, the quantities L_{res} and $L_{\max}$ capture the interactions between the various components in the gradient ∇f. In the case of twice continuously differentiable f, we can understand these interactions by observing the diagonal and off-diagonal terms of the Hessian $\nabla^2 f(x)$.

3.3 Algorithm

In our algorithm AsySPCD, multiple processors have access to a shared data structure for the vector x, and each processor is able to compute a randomly chosen element of the gradient vector $\nabla f(x)$. Each processor repeatedly runs the following proximal coordinate descent process. (Choice of the steplength parameter γ is discussed further in the next section.)

R: Choose an index $i \in \{1, 2, \ldots, n\}$ at random, read x into the local storage location $\hat{x}$, and evaluate $\nabla_i f(\hat{x})$;

U: Update component i of the shared x by taking a step of length $\gamma/L_{\max}$ in the direction $-\nabla_i f(\hat{x})$, follows by a proximal operation:

$$x \leftarrow \mathcal{P}_{i, \frac{\gamma}{L_{\max}} g_i}\left(x - \frac{\gamma}{L_{\max}} e_i \nabla_i f(\hat{x})\right).$$

Notice that each step changes just a single element of x, that is, the ith element. Unlike standard proximal coordinate descent, the value $\hat{x}$ at which the coordinate gradient is calculated usually differs from the value of x to which the update is applied, because while the processor is evaluating its gradient, other processors may repeatedly update the value of x stored in memory. As mentioned above, we used an "inconsistent read" model of asynchronous computation here, in contrast to the "consistent read" models of AsySCD in Chapter 2 and Hogwild! (Niu et al., 2011). Figure 3.1 shows how inconsistent reading can occur, as a result of updating of components of x while it is being read. Consistent reading can be guaranteed only by means of a software lock, but such a mechanism degrades parallel performance significantly. In fact, the implementations of Hogwild!

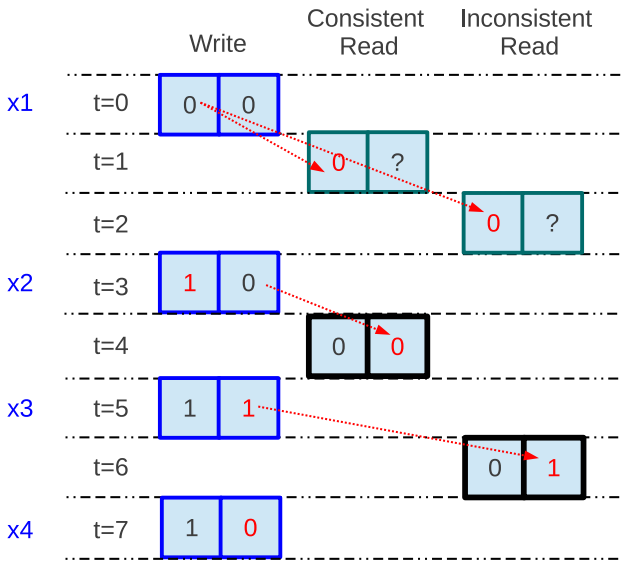

Figure 3.1: Time sequence of writes and reads of a two-variable vector, showing instances of consistent and inconsistent reading. The left column shows the initial vector at time 0, stored in shared memory, with updates to single components at times 3, 5, and 7. The middle column shows a consistent read, in which the first component is read at time 1 and the second component is read at time 4. The read vector is equal to the shared-memory vector at time 0. The right column shows an inconsistent read, in which the first component is read at time 2 and the second component is read at time 6. Beause of intervening writes to these components, the read vector does not match the versions that appeared in shared memory at any time point.

(Niu et al., 2011) and AsySCD do not use any software lock, and in this respect the computations in those papers are not quite compatible with the analysis.

Algorithm 2 Asynchronous Stochastic Coordinate Descent Algorithm $x_J = \text{AsySPCD}(x_0, \gamma, J)$

Require: x_0, γ, and J
Ensure: x_J
1: Initialize $j \leftarrow 0$;
2: **while** $j < J$ **do**
3: Choose $i(j)$ from $\{1, 2, \ldots, n\}$ with equal probability;
4: $x_{j+1} \leftarrow \mathcal{P}_{i(j), \frac{\gamma}{L_{\max}} g_{i(j)}} \left(x_j - \frac{\gamma}{L_{\max}} e_{i(j)} \nabla_{i(j)} f(\hat{x}_j) \right)$;
5: $j \leftarrow j + 1$;
6: **end while**

The "global" view of algorithm AsySPCD is shown in **Algorithm** 2. To obtain this version from the "local" version, we introduce a counter j to track the total number of updates applied to x, so that x_j is the state of x in memory after update j is performed. We use $i(j)$ to denote the component that is updated at iteration j, and $\hat{x}_j$ for value of x that is used in the calculation of the

gradient element $\nabla f_{i(j)}$. The components of $\hat{x}_j$ may have different ages. Some components may be current at iteration j, others may not reflect recent updates made by other processors. We assume however that there is an upper bound of τ on the age of each component, measured in terms of updates. $K(j)$ defines an iterate set such that

$$x_j = \hat{x}_j + \sum_{d \in K(j)} (x_{d+1} - x_d).$$

One can see that $d \leqslant j - 1 \ \forall d \in K(j)$. Here we assume τ to be the upper bound on the age of all elements in $K(j)$, for all j, so that $\tau \geqslant j - \min\{d \mid d \in K(j)\}$. We assume further that $K(j)$ is ordered from oldest to newest index (that is, smallest to largest). Note that $K(j)$ is empty if $x_j = \hat{x}_j$, that is, if the step is simply an ordinary stochastic coordinate gradient update. The value of τ corresponds closely to the number of processors involved in the computation. Note that the consistent read case can be considered as a special case of the inconsistent read model: all coordinates in $\hat{x}$ have the same age which is bounded by τ.

3.4 Main Results

This section presents results on convergence of AsySPCD. The theorem encompasses both the linear rate for optimally strongly convex f and the sublinear rate for general convex f. The result depends strongly on the delay parameter τ. The proofs are highly technical, and are relegated to Appendix. We note the proof techniques differ significantly from those used for the consistent-read algorithms of Niu et al. (2011) and Chapter 2.

We start by describing the key idea of the algorithm, which is reflected in the way that it chooses the steplength parameter γ. Denoting $\bar{x}_{j+1}$ by

$$\bar{x}_{j+1} := \mathcal{P}_{\frac{\gamma}{L_{\max}} g}\left(x_j - \frac{\gamma}{L_{\max}} \nabla f(\hat{x}_j)\right), \tag{3.5}$$

we can see that

$$(x_{j+1})_{i(j)} = (\bar{x}_{j+1})_{i(j)}, \quad (x_{j+1})_i = (x_j)_i \ \text{ for } i \neq i(j),$$

so that $x_{j+1} - x_j = [(\bar{x}_{j+1})_{i(j)} - (x_j)_{i(j)}]e_{i(j)}$. Thus, we have

$$\mathbb{E}_{i(j)}(x_{j+1} - x_j) = \frac{1}{n}\sum_{i=1}^{n}[(\bar{x}_{j+1})_i - (x_j)_i]e_i = \frac{1}{n}[\bar{x}_{j+1} - x_j].$$

Therefore, we can view $\bar{x}_{j+1} - x_j$ as capturing the expected behavior of $x_{j+1} - x_j$. Note that when $g(x) = 0$, we have $\bar{x}_{j+1} - x_j = -(\gamma/L_{\max})\nabla f(\hat{x}_j)$, a standard negative-gradient step. The choice of steplength parameter γ entails a tradeoff: We would like γ to be long enough that significant progress is made at each step, but not so long that the gradient information computed at $\hat{x}_j$ is stale

and irrelevant by the time the update is applied to x_j. We enforce this tradeoff by means of a bound on the ratio of expected squared norms on $x_j - \bar{x}_{j+1}$ at successive iterates; specifically,

$$\mathbb{E}\|x_{j-1} - \bar{x}_j\|^2 \leqslant \rho \mathbb{E}\|x_j - \bar{x}_{j+1}\|^2, \tag{3.6}$$

where $\rho > 1$ is a user defined parameter. The analysis becomes a delicate balancing act in the choice of ρ and steplength γ between aggression and excessive conservatism. We find, however, that these values can be chosen to ensure steady convergence for the asynchronous method at a linear rate, with rate constants that are almost consistent with a standard short-step proximal full-gradient descent, when the optimal strong convexity condition (3.2) is satisfied.

Our main convergence result is the following.

Theorem 3.1. *Suppose that Assumption 3.1 is satisfied. Let ρ be a constant that satisfies $\rho > 1 + 4/\sqrt{n}$, and define the quantities θ, θ', and ψ as follows:*

$$\theta := \frac{\rho^{(\tau+1)/2} - \rho^{1/2}}{\rho^{1/2} - 1}, \quad \theta' := \frac{\rho^{(\tau+1)} - \rho}{\rho - 1}, \quad \psi := 1 + \frac{\tau\theta'}{n} + \frac{2\Lambda\theta}{\sqrt{n}}. \tag{3.7}$$

Suppose that the steplength parameter $\gamma > 0$ satisfies the following two bounds:

$$\gamma \leqslant \frac{1}{\psi}, \quad \gamma \leqslant \frac{\sqrt{n}(1 - \rho^{-1}) - 4}{4(1+\theta)\Lambda}. \tag{3.8}$$

Then we have

$$\mathbb{E}\|x_{j-1} - \bar{x}_j\|^2 \leqslant \rho \mathbb{E}\|x_j - \bar{x}_{j+1}\|^2, \quad j = 1, 2, \ldots. \tag{3.9}$$

If the optimal strong convexity property (3.2) *holds with $l > 0$, we have for $j = 1, 2, \ldots$ that*

$$\begin{aligned} &\mathbb{E}\|x_j - \mathcal{P}_S(x_j)\|^2 + \frac{2\gamma}{L_{\max}}(\mathbb{E}F(x_j) - F^*) \\ &\quad \leqslant \left(1 - \frac{l}{n(l + \gamma^{-1}L_{\max})}\right)^j \left(\|x_0 - \mathcal{P}_S(x_0)\|^2 + \frac{2\gamma}{L_{\max}}(F(x_0) - F^*)\right), \end{aligned} \tag{3.10}$$

while for general smooth convex function f, we have

$$\mathbb{E}F(x_j) - F^* \leqslant \frac{n(\|x_0 - \mathcal{P}_S(x_0)\|^2 L_{\max} + 2\gamma(F(x_0) - F^*))}{2\gamma(n + j)}. \tag{3.11}$$

The following corollary proposes an interesting particular choice for the parameters for which the convergence expressions become more comprehensible. The result requires a condition on the delay bound τ in terms of n and the ratio Λ.

Corollary 3.2. *Suppose that Assumption 3.1 holds and that*

$$4e\Lambda(\tau+1)^2 \leqslant \sqrt{n}. \tag{3.12}$$

If we choose

$$\rho = \left(1 + \frac{4e\Lambda(\tau+1)}{\sqrt{n}}\right)^2, \tag{3.13}$$

then the steplength $\gamma = 1/2$ *will satisfy the bounds (3.8). In addition, when the optimal strong convexity property (3.2) holds with* $l > 0$*, we have for* $j = 1, 2, \ldots$ *that*

$$\mathbb{E}f(x_j) - f^* \leqslant \left(1 - \frac{l}{n(l + 2L_{\max})}\right)^j (L_{\max}\|x_0 - \mathcal{P}_S(x_0)\|^2 + f(x_0) - f^*), \tag{3.14}$$

while for the case of general convex f*, we have*

$$\mathbb{E}f(x_j) - f^* \leqslant \frac{n(L_{\max}\|x_0 - \mathcal{P}_S(x_0)\|^2 + f(x_0) - f^*)}{j + n}. \tag{3.15}$$

We note that the linear rate (3.14) is broadly consistent with the linear rate for the classical steepest descent method applied to strongly convex functions, which has a rate constant of $(1-2l/L)$, where L is the standard Lipschitz constant for ∇f. Suppose we assume (not unreasonably) that n steps of stochastic coordinate descent cost roughly the same as one step of steepest descent, and that $l \leqslant L_{\max}$. It follows from (3.14) that n steps of stochastic coordinate descent would achieve a reduction factor of about

$$1 - \frac{l}{2L_{\max} + l} \leqslant 1 - \frac{l}{3L_{\max}},$$

so a standard argument would suggest that stochastic coordinate descent would require about $6L_{\max}/L$ times more computation. Since $L_{\max}/L \in [1/n, 1]$, the stochastic asynchronous approach may actually require less computation. It may also gain an advantage from the parallel asynchronous implementation. A parallel implementation of standard gradient descent would require synchronization and careful division of the work of evaluating ∇f, whereas the stochastic approach can be implemented in an asynchronous fashion.

For the general convex case, (3.15) defines a sublinear rate, whose relationship with the rate of standard gradient descent for general convex optimization is similar to the previous paragraph.

Note that the results in Theorem 3.1 and Corollary 3.2 are consistent with the analysis for constrained AsySCD in Chapter 2, but this chapter considers the more general case of composite optimization and the inconsistent-read model of parallel computation.

As noted in Section 3.2, the parameter τ corresponds closely to the number of cores that can be involved in the computation, since if all cores are working at the same rate, we would expect each other core to make one update between the times at which x is read and (later) updated. If τ is

small enough that (3.12) holds, the analysis indicates that near-linear speedup in the number of processors is achievable. A small value for the ratio Λ (not much greater than 1) implies a greater degree of potential parallelism. As we note at the end of Section 3.2, this ratio tends to closer to 1 than to $\sqrt{n}$ in some important applications. In these situations, the bound (3.12) indicates that τ can vary like $n^{1/4}$ and still not affect the iteration-wise convergence rate. (That is, linear speedup is possible.) This quantity is consistent with the analysis for constrained AsySCD but weaker than the unconstrained AsySCD (which allows the maximal number of cores being $O(n^{1/2})$) in Chapter 2. A further comparison is with the asynchronous randomized Kaczmarz algorithm (Liu et al., 2014) (also see Chapter 4) which allows $O(m)$ cores to be used efficiently when solving a consistent sparse linear system.

We conclude this section with a high-probability bound. The result follows immediately from Markov's inequality. See Theorem 2.3 for a related result and complete proof.

Theorem 3.3. *Suppose that the conditions of Corollary 3.2 hold, including the choice of* ρ*. Then for* $\epsilon > 0$ *and* $\eta \in (0,1)$*, we have that*

$$\mathbb{P}\left(F(x_j) - F^* \leqslant \epsilon\right) \geqslant 1 - \eta.$$

provided that one of the following conditions holds. In the optimally strongly convex case (3.2) with $l > 0$*, we require*

$$j \geqslant \frac{n(l + 2L_{\max})}{l} \left|\log \frac{L_{\max}\|x_0 - \mathcal{P}_S(x_0)\|^2 + F(x_0) - F^*}{\epsilon\eta}\right|,$$

iterations, while in the general convex case, it suffices that

$$j \geqslant \frac{n(L_{\max}\|x_0 - \mathcal{P}_S(x_0)\|^2 + F(x_0) - F^*)}{\epsilon\eta} - n.$$

3.5 Experiments

This section presents some results to illustrate the effectiveness of AsySPCD, in particular, the fact that near-linear speedup can be observed on a multicore machine. Note that more comprehensive experiments can be found in Section 2.6 and Sridhar et al. (2013), for unconstrained and box-constrained problems. Although the analysis in Chapter 2 assumes consistent read, this is not enforced in the implementation, so apart from the fact that we now include a prox-step to account for the regularization term, the implementations in Chapter 2 and Sridhar et al. (2013) are quite similar to the one employed in this section.

We apply our code for AsySPCD to the following "ℓ_2-ℓ_1" problem:

$$\min_x \quad \frac{1}{2}\|Ax - b\|^2 + \lambda\|x\|_1 \equiv \frac{1}{2}x^T A^T Ax - b^T Ax + \frac{1}{2}b^T b + \lambda\|x\|_1.$$

The elements of $A \in \mathbb{R}^{m \times n}$ are selected i.i.d. from a Gaussian $\mathcal{N}(0,1)$ distribution. To construct a sparse true solution $x^* \in \mathbb{R}^n$, given the dimension n and sparsity s, we select s entries of x^* at random to be nonzero and $\mathcal{N}(0,1)$ normally distributed, and set the rest to zero. The measurement vector $b \in \mathbb{R}^m$ is obtained by $b = Ax^* + \epsilon$, where elements of the noise vector $\epsilon \in \mathbb{R}^m$ are i.i.d. $\mathcal{N}(0, \sigma^2)$, where the value of σ controls the signal-to-noise ratio.

Our experiments run on 1 to 10 threads of an Intel Xeon machine, with all threads sharing a single memory socket. Our implementations deviate modestly from the version of AsySPCD described in Section 3.3. We compute $Q := A^T A \in \mathbb{R}^{n \times n}$ and $c := A^T b \in \mathbb{R}^n$ offline. Q and c are partitioned into slices (row submatrices) and subvectors (respectively) of equal size, and each thread is assigned one submatrix from Q and the corresponding subvector from c. During the algorithm, each thread updates the elements of x corresponding to its slice of Q, in order. After one scan, or "epoch" is complete, it reorders the indices randomly, then repeats the process. This scheme essentially changes the scheme from sampling with replacement (as analyzed) to sampling without replacement, which has demonstrated empirically better performance on many related problems. (The same advantage is noted in the implementations of Hogwild! (Niu et al., 2011).)

For the plots in Figures 3.2 and 3.3, we choose $\sigma = 0.01$ with $m = 6000$, $n = 10000$, and $s = 10$ in Figure 3.2 and $m = 12000$, $n = 20000$, and $s = 20$ in Figure 3.3. We set $\lambda = 20\sqrt{m \log(n)}\sigma$ (a value of the order of $\sqrt{m \log(n)}\sigma$ is suggested by compressed sensing theory) and the steplength γ is set as 1 in both figures. Note that in both cases, we can estimate the ratio of $L_{\text{res}}/L_{\max}$ roughly by $1 + \sqrt{n/m} \approx 2.3$ as suggested in the end of Section 3.2. Our final computed values of x have nonzeros in the same locations as the chosen solution x^*, though the values differ, because of the noise in b.

The left-hand graph in each figure indicates the number of threads / cores and plots objective function value vs epoch count, where one epoch is equivalent to n iterations. Note that the curves are almost overlaid, indicating that the total workload required for AsySPCD is almost independent of the number of cores used in the computation. This observation validates our result in Corollary 3.2, which indicates that provided τ is below a certain threshold, it does not seriously affect the rate of convergence, as a function of total computation performed. The right-hand graph in each figure shows speedup when executed on different numbers of cores. Near-linear speedup is observed in Figure 3.3, while there is a slight dropoff for the larger numbers of cores in Figure 3.2. The difference can be explain by the smaller dimension of the problem illustrated in Figure 3.2.

3.6 Conclusion

This chapter extends the AsySCD algorithm in Chapter 2 for minimizing composite objectives of the form (3.1). Particularly, the "inconsistent read" model (a more realistic situation) is considered in the analysis. Sublinear convergence (at rate $1/k$) is proved for general convex functions, with stronger linear convergence results for problems that satisfy the optimal strong convexity property

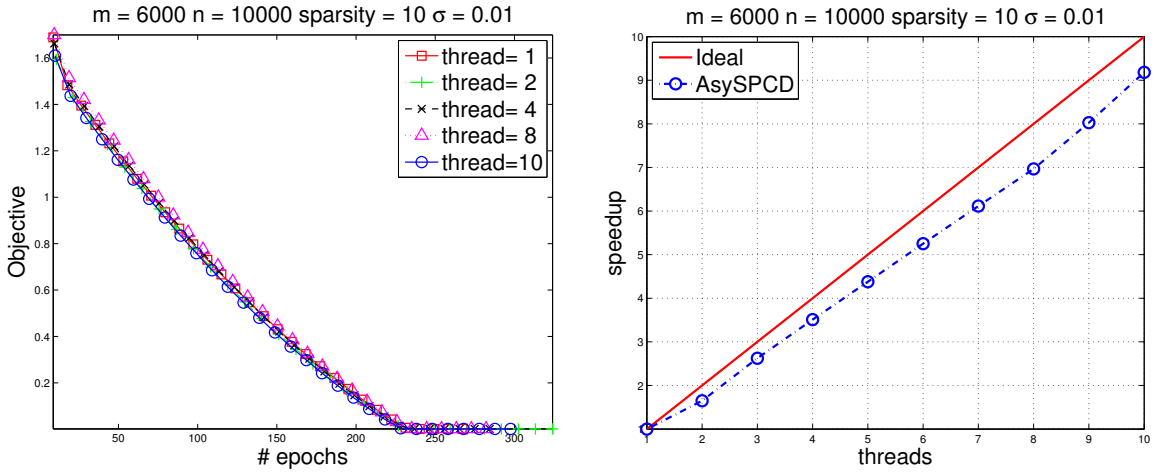

Figure 3.2: The left graph plots objective function vs epochs for 1, 2, 4, 8, and 10 cores. The right graph shows speedup obtained for implementation on 1-10 cores, plotted against the ideal (linear) speedup.

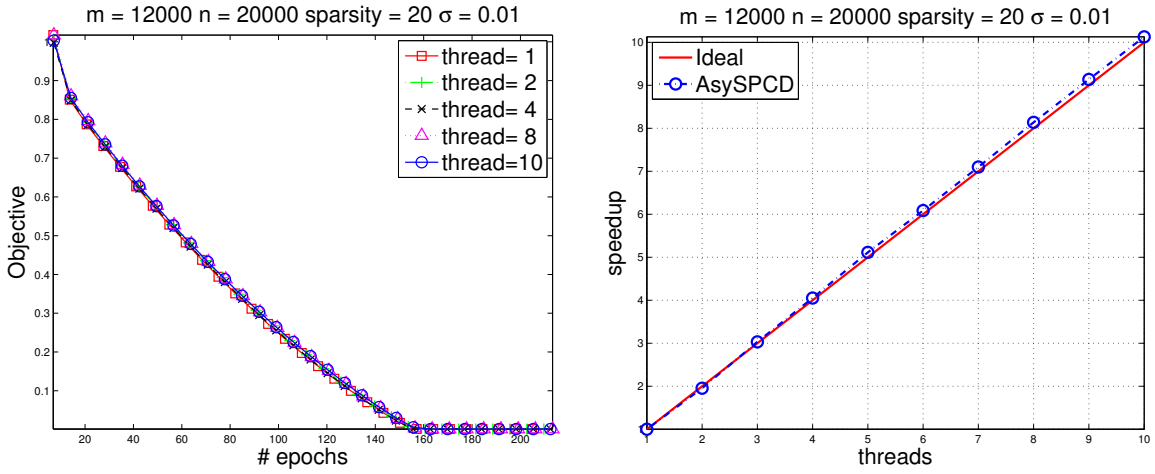

Figure 3.3: The left graph plots objective function vs epochs for 1, 2, 4, 8, and 10 cores. The right graph shows speedup obtained for implementation on 1-10 cores, plotted against the ideal (linear) speedup.

(3.2). Our analysis indicates the extent to which parallel implementations can be expected to yield near-linear speedup, in terms of a parameter that quantifies the cross-coordinate interactions in the gradient ∇f and a parameter τ that bounds the delay in updating. Our computational experience confirms that the linear speedup properties suggested by the analysis can be observed in practice.

Appendix

This section provides the proofs for the main convergence results. We start with some preliminaries, then proceed to proofs of Theorem 3.1 and Corollary 3.2.

Preliminaries

Note that the component indices $i(0), i(1), \ldots, i(j), \ldots$ in Algorithm 2 are independent random variables. We use $\mathbb{E}$ to denote the expectation over all random variables, and $\mathbb{E}_{i(j)}$ to denote the conditional expectation in term of $i(j)$ given $i(0), i(1), \ldots, i(j-1)$. We also denote

$$(\Delta_j)_{i(j)} := (x_j - x_{j+1})_{i(j)}, \tag{3.16}$$

and formulate the update in Step 4 of Algorithm 2 in the following way:

$$x_{j+1} = \arg\min_x \langle \nabla_{i(j)} f(\hat{x}_j), (x - x_j)_{i(j)} \rangle + \frac{L_{\max}}{2\gamma} \|x - x_j\|^2 + g_{i(j)}((x)_{i(j)}).$$

(Note that $(x_{j+1})_i = (x_j)_i$ for $i \neq i(j)$.) From the optimality condition for this formulation, we have $\forall\, x$

$$\left\langle (x - x_{j+1})_{i(j)}, \nabla_{i(j)} f(\hat{x}_j) - \frac{L_{\max}}{\gamma} (\Delta_j)_{i(j)} \right\rangle + g_{i(j)}((x)_{i(j)}) - g_{i(j)}((x_{j+1})_{i(j)}) \geqslant 0.$$

By rearranging this expression and substituting $\mathcal{P}_S(x)$ for x, we find that the following inequality is true for all x:

$$\begin{aligned} g_{i(j)}((\mathcal{P}_S(x))_{i(j)}) - g_{i(j)}((x_{j+1})_{i(j)}) + \langle (\mathcal{P}_S(x) - x_{j+1})_{i(j)}, \nabla_{i(j)} f(\hat{x}_j) \rangle \\ \geqslant \frac{L_{\max}}{\gamma} \langle (\mathcal{P}_S(x) - x_{j+1})_{i(j)}, (\Delta_j)_{i(j)} \rangle. \end{aligned} \tag{3.17}$$

From the definition of $L_{\max}$, and using the notation (3.16), we have

$$f(x_{j+1}) \leqslant f(x_j) + \langle \nabla_{i(j)} f(x_j), -(\Delta_j)_{i(j)} \rangle + \frac{L_{\max}}{2} |(\Delta_j)_{i(j)}|^2,$$

or equivalently,

$$\langle \nabla_{i(j)} f(x_j), (\Delta_j)_{i(j)} \rangle \leqslant f(x_j) - f(x_{j+1}) + \frac{L_{\max}}{2} |(\Delta_j)_{i(j)}|^2. \tag{3.18}$$

From optimality conditions for the definition of $\bar{x}_{j+1}$ in (3.5), we have that

$$\bar{x}_{j+1} = \arg\min_x \langle \nabla f(\hat{x}_j), x - x_j \rangle + \frac{L_{\max}}{2\gamma} \|x - x_j\|^2 + g(x),$$

so that

$$g(x) - g(\bar{x}_{j+1}) + \langle x - \bar{x}_{j+1}, \nabla f(\hat{x}_j) + \frac{L_{\max}}{\gamma} (\bar{x}_{j+1} - x_j) \rangle \geqslant 0 \quad \forall\, x. \tag{3.19}$$

We now define

$$\Delta_j := x_j - \bar{x}_{j+1}, \tag{3.20}$$

and note that this definition is consistent with $(\Delta_j)_{i(j)}$ defined in (3.16). It can be seen that

$$\mathbb{E}_{i(j)}(\|x_{j+1} - x_j\|^2) = \frac{1}{n} \|\bar{x}_{j+1} - x_j\|^2. \tag{3.21}$$

Recalling that all indices in $K(j)$ are sorted in the increasing order from smallest (oldest) iterate to largest (newest) iterate, we use $K(j)_t$ to denote the t-th smallest entry in $K(j)$. We define

$$\hat{x}_{j,T} := \hat{x}_j + \sum_{t=1}^{T} (x_{K(j)_t + 1} - x_{K(j)_t}).$$

We have the following relations:

$$\begin{aligned}
\hat{x}_j &= \hat{x}_{j,0} \\
x_j &= \hat{x}_{j,|K(j)|} \\
x_j - \hat{x}_j &= \sum_{t=0}^{|K(j)|-1} (\hat{x}_{j,t+1} - \hat{x}_{j,t}) \\
\nabla f(x_j) - \nabla f(\hat{x}_j) &= \sum_{t=0}^{|K(j)|-1} (\nabla f(\hat{x}_{j,t+1}) - \nabla f(\hat{x}_{j,t})).
\end{aligned}$$

Furthermore, we have

$$\begin{aligned}
\|\nabla f(x_j) - \nabla f(\hat{x}_j)\| &= \|\nabla f(\hat{x}_{j,0}) - \nabla f(\hat{x}_{j,|K(j)|})\| \\
&= \left\| \sum_{t=0}^{|K(j)|-1} (\nabla f(\hat{x}_{j,t}) - \nabla f(\hat{x}_{j,t+1})) \right\| \\
&\leqslant \sum_{t=0}^{|K(j)|-1} \|\nabla f(\hat{x}_{j,t}) - \nabla f(\hat{x}_{j,t+1})\| \\
&\leqslant L_{res} \sum_{t=0}^{|K(j)|-1} \|\hat{x}_{j,t} - \hat{x}_{j,t+1}\| \\
&= L_{res} \sum_{t=1}^{|K(j)|} \|x_{K(j)_t} - x_{K(j)_t+1}\| \\
&= L_{res} \sum_{d \in K(j)} \|x_{d+1} - x_d\|, \qquad (3.22)
\end{aligned}$$

where the second inequality holds because $\hat{x}_{j,t}$ and $\hat{x}_{j,t+1}$ differ in only a single coordinate.

Proof of Theorem 3.1

Proof. We prove (3.9) by induction. First, note that for any vectors a and b, we have

$$\|a\|^2 - \|b\|^2 = 2\|a\|^2 - (\|a\|^2 + \|b\|^2) \leqslant 2\|a\|^2 - 2\langle a, b\rangle = 2\langle a, a - b\rangle 2\|a\|\|b - a\|.$$

Thus for all j, we have

$$\|x_{j-1} - \bar{x}_j\|^2 - \|x_j - \bar{x}_{j+1}\|^2 \leqslant 2\|x_{j-1} - \bar{x}_j\|\|x_j - \bar{x}_{j+1} - x_{j-1} + \bar{x}_j\|. \qquad (3.23)$$

The second factor in the r.h.s. of (3.23) is bounded as follows:

$$
\begin{aligned}
&\|x_j - \bar{x}_{j+1} - x_{j-1} + \bar{x}_j\| \\
&= \left\| x_j - \mathcal{P}_{\frac{\gamma}{L_{max}} g}\left(x_j - \frac{\gamma}{L_{max}}\nabla f(\hat{x}_j)\right) - \left(x_{j-1} - \mathcal{P}_{\frac{\gamma}{L_{max}} g}(x_{j-1} - \frac{\gamma}{L_{max}}\nabla f(\hat{x}_{j-1}))\right)\right\| \\
&\leqslant \|x_j - x_{j-1}\| + \left\|\mathcal{P}_{\frac{\gamma}{L_{max}} g}\left(x_j - \frac{\gamma}{L_{max}}\nabla f(\hat{x}_j)\right) - \mathcal{P}_{\frac{\gamma}{L_{max}} g}\left(x_{j-1} - \frac{\gamma}{L_{max}}\nabla f(\hat{x}_{j-1})\right)\right\| \\
&\leqslant 2\|x_j - x_{j-1}\| + \frac{\gamma}{L_{max}}\|\nabla f(\hat{x}_j) - \nabla f(\hat{x}_{j-1})\| \\
&\qquad \text{(by the nonexpansive property of } \mathcal{P}_{\frac{\gamma}{L_{max}} g}) \\
&= 2\|x_j - x_{j-1}\| + \frac{\gamma}{L_{max}}\|\nabla f(\hat{x}_j) - \nabla f(x_j) + \nabla f(x_j) - \nabla f(x_{j-1}) \\
&\qquad + \nabla f(x_{j-1}) - \nabla f(\hat{x}_{j-1})\| \\
&\leqslant 2\|x_j - x_{j-1}\| + \frac{\gamma}{L_{max}}\big(\|\nabla f(\hat{x}_j) - \nabla f(x_j)\| + \|\nabla f(x_j) - \nabla f(x_{j-1})\| \\
&\qquad + \|\nabla f(x_{j-1}) - \nabla f(\hat{x}_{j-1})\|\big) \\
&\leqslant (2 + \Lambda\gamma)\,\|x_j - x_{j-1}\| + \frac{\gamma}{L_{max}}\|\nabla f(\hat{x}_j) - \nabla f(x_j)\| \\
&\qquad + \frac{\gamma}{L_{max}}\|\nabla f(x_{j-1}) - \nabla f(\hat{x}_{j-1})\| \\
&\leqslant (2 + \Lambda\gamma)\,\|x_j - x_{j-1}\| + \Lambda\gamma \sum_{d\in K(j)} \|x_d - x_{d+1}\| \\
&\qquad + \Lambda\gamma \sum_{d\in K(j-1)} \|x_d - x_{d+1}\| \quad \text{(from (3.22))} \qquad (3.24) \\
&\leqslant (2 + \Lambda\gamma)\,\|x_j - x_{j-1}\| + \Lambda\gamma \sum_{d=j-\tau}^{j-1} \|x_d - x_{d+1}\| + \Lambda\gamma \sum_{d=j-1-\tau}^{j-2} \|x_d - x_{d+1}\| \\
&\leqslant (2 + 2\Lambda\gamma)\,\|x_j - x_{j-1}\| + 2\Lambda\gamma \sum_{d=j-1-\tau}^{j-2} \|x_d - x_{d+1}\|, \qquad (3.25)
\end{aligned}
$$

where the fourth inequality uses $\|\nabla f(x_j) - \nabla f(x_{j-1})\| \leqslant L_{res}\|x_j - x_{j-1}\|$, since x_j and x_{j-1} differ in just one component.

We set $j = 1$, and note that $K(0) = \emptyset$ and $K(1) \subset \{0\}$. In this case, we obtain a bound from (3.24)

$$
\|x_1 - \bar{x}_2 + x_0 - \bar{x}_1\| \leqslant (2 + \Lambda\gamma)\,\|x_1 - x_0\| + \Lambda\gamma\|x_1 - x_0\| = (2 + 2\Lambda\gamma)\,\|x_1 - x_0\|.
$$

By substituting this bound in (3.23) and setting $j = 1$, and taking expectations, we obtain

$$\begin{aligned}\mathbb{E}(\|x_0 - \bar{x}_1\|^2) - \mathbb{E}(\|x_1 - \bar{x}_2\|^2) &\leqslant 2\mathbb{E}(\|x_0 - \bar{x}_1\|\|x_1 - \bar{x}_2 - x_0 + \bar{x}_1\|) \\ &\leqslant (4 + 4\Lambda\gamma)\,\mathbb{E}(\|\bar{x}_1 - x_0\|\|x_1 - x_0\|). \end{aligned} \tag{3.26}$$

For any positive scalars μ_1, μ_2, and α, we have

$$\mu_1\mu_2 \leqslant \frac{1}{2}(\alpha\mu_1^2 + \alpha^{-1}\mu_2^2). \tag{3.27}$$

It follows that

$$\begin{aligned}\mathbb{E}(\|x_j - x_{j-1}\|\|\bar{x}_j - x_{j-1}\|) &\leqslant \frac{1}{2}\mathbb{E}(n^{1/2}\|x_j - x_{j-1}\|^2 + n^{-1/2}\|\bar{x}_j - x_{j-1}\|^2) \\ &= \frac{1}{2}\mathbb{E}(n^{1/2}\mathbb{E}_{i(j-1)}(\|x_j - x_{j-1}\|^2) + n^{-1/2}\|\bar{x}_j - x_{j-1}\|^2) \\ &= \frac{1}{2}\mathbb{E}(n^{-1/2}\|\bar{x}_j - x_{j-1}\|^2 + n^{-1/2}\|\bar{x}_j - x_{j-1}\|^2) \quad \text{(from (3.21))} \\ &= n^{-1/2}\mathbb{E}\|\bar{x}_j - x_{j-1}\|^2. \end{aligned} \tag{3.28}$$

By taking $j = 1$ in (3.28), and substituting in (3.26), we obtain

$$\mathbb{E}(\|x_0 - \bar{x}_1\|^2) - \mathbb{E}(\|x_1 - \bar{x}_2\|^2) \leqslant n^{-1/2}\,(4 + 4\Lambda\gamma)\,\mathbb{E}\|\bar{x}_1 - x_0\|^2,$$

which implies that

$$\mathbb{E}(\|x_0 - \bar{x}_1\|^2) \leqslant \left(1 - \frac{4 + 4\gamma\Lambda}{\sqrt{n}}\right)^{-1} \mathbb{E}(\|x_1 - \bar{x}_2\|^2) \leqslant \rho\mathbb{E}(\|x_1 - \bar{x}_2\|^2).$$

To see the last inequality, one only needs to verify that

$$\rho^{-1} \leqslant 1 - \frac{4 + 4\gamma\Lambda}{\sqrt{n}} \;\Leftrightarrow\; \gamma \leqslant \frac{\sqrt{n}(1 - \rho^{-1}) - 4}{4\Lambda},$$

where the last inequality follows from the second bound for γ in (3.8). We have thus shown that (3.9) holds for $j = 1$.

To take the inductive step, we assume that (3.9) holds up to index $j - 1$. We have for

$j-1-\tau \leqslant d \leqslant j-2$ and any $\beta > 0$ (using (3.27) again) that

$$\begin{aligned}
&\mathbb{E}(\|x_d - x_{d+1}\|\|\bar{x}_j - x_{j-1}\|) \\
&\quad \leqslant \frac{1}{2}\mathbb{E}(n^{1/2}\beta\|x_d - x_{d+1}\|^2 + n^{-1/2}\beta^{-1}\|\bar{x}_j - x_{j-1}\|^2) \\
&\quad = \frac{1}{2}\mathbb{E}(n^{1/2}\beta\mathbb{E}_{i(d)}(\|x_d - x_{d+1}\|^2) + n^{-1/2}\beta^{-1}\|\bar{x}_j - x_{j-1}\|^2) \\
&\quad = \frac{1}{2}\mathbb{E}(n^{-1/2}\beta\|x_d - \bar{x}_{d+1}\|^2 + n^{-1/2}\beta^{-1}\|\bar{x}_j - x_{j-1}\|^2) \qquad \text{(from (3.21))} \\
&\quad \leqslant \frac{1}{2}\mathbb{E}(n^{-1/2}\beta\rho^{j-1-d}\|x_{j-1} - \bar{x}_j\|^2 + n^{-1/2}\beta^{-1}\|\bar{x}_j - x_{j-1}\|^2) \\
&\qquad \text{(by the inductive hypothesis).}
\end{aligned}$$

Thus by setting $\beta = \rho^{(d+1-j)/2}$, we obtain

$$\mathbb{E}(\|x_d - x_{d+1}\|\|\bar{x}_j - x_{j-1}\|) \leqslant \frac{\rho^{(j-1-d)/2}}{n^{1/2}}\mathbb{E}\left(\|\bar{x}_j - x_{j-1}\|^2\right). \tag{3.29}$$

By substituting (3.25) into (3.23) and taking expectation on both sides of (3.23), we obtain

$$\begin{aligned}
&\mathbb{E}(\|x_{j-1} - \bar{x}_j\|^2) - \mathbb{E}(\|x_j - \bar{x}_{j+1}\|^2) \\
\leqslant &2\mathbb{E}(\|\bar{x}_j - x_{j-1}\|\|\bar{x}_j - \bar{x}_{j+1} + x_j - x_{j-1}\|) \\
\leqslant &2\mathbb{E}\left(\|\bar{x}_j - x_{j-1}\|\left((2+2\Lambda\gamma)\|x_j - x_{j-1}\| + 2\Lambda\gamma\sum_{d=j-1-\tau}^{j-2}\|x_d - x_{d+1}\|\right)\right) \\
= &(4+4\Lambda\gamma)\,\mathbb{E}(\|\bar{x}_j - x_{j-1}\|\|x_j - x_{j-1}\|) + 4\Lambda\gamma\sum_{d=j-1-\tau}^{j-2}\mathbb{E}(\|\bar{x}_j - x_{j-1}\|\|x_d - x_{d+1}\|) \\
\leqslant &n^{-1/2}(4+4\Lambda\gamma)\mathbb{E}(\|\bar{x}_j - x_{j-1}\|^2) \\
&\qquad + n^{-1/2}4\Lambda\gamma\mathbb{E}(\|x_{j-1} - \bar{x}_j\|^2)\sum_{d=j-1-\tau}^{j-2}\rho^{(j-1-d)/2} \quad \text{(from (3.28) and (3.29))} \\
\leqslant &n^{-1/2}(4+4\Lambda\gamma)\mathbb{E}(\|\bar{x}_j - x_{j-1}\|^2) + n^{-1/2}4\Lambda\gamma\mathbb{E}(\|x_{j-1} - \bar{x}_j\|^2)\sum_{t=1}^{\tau}\rho^{t/2} \\
= &n^{-1/2}\left(4+4\Lambda\gamma(1+\theta)\right)\mathbb{E}(\|x_{j-1} - \bar{x}_j\|^2),
\end{aligned}$$

where the last equality follows from the definition of θ in (3.7). It follows that

$$\begin{aligned}
\mathbb{E}(\|x_{j-1} - \bar{x}_j\|^2) &\leqslant \left(1 - n^{-1/2}\left(4+4\Lambda\gamma(1+\theta)\right)\right)^{-1}\mathbb{E}(\|x_j - \bar{x}_{j+1}\|^2) \\
&\leqslant \rho\mathbb{E}(\|x_j - \bar{x}_{j+1}\|^2).
\end{aligned}$$

To see the last inequality, one only needs to verify that

$$\rho^{-1} \leqslant 1 - \frac{4 + 4\gamma\Lambda(1+\theta)}{\sqrt{n}} \;\Leftrightarrow\; \gamma \leqslant \frac{\sqrt{n}(1-\rho^{-1}) - 4}{4\Lambda(1+\theta)},$$

and the last inequality is true because of the upper bound of γ in (3.8). We have thus proved (3.9).

Next we will show the expectation of the objective F is monotonically decreasing. We have using the definition (3.16) that

$$\begin{aligned}
&\mathbb{E}_{i(j)} F(x_{j+1}) = \mathbb{E}_{i(j)} \left[f(x_j - (\Delta_j)_{i(j)}) + g(x_{j+1}) \right] \\
&\leqslant \mathbb{E}_{i(j)} \Bigg[f(x_j) + \langle \nabla_{i(j)} f(x_j), (\bar{x}_{j+1} - x_j)_{i(j)} \rangle + \frac{L_{\max}}{2} \|(x_{j+1} - x_j)_{i(j)}\|^2 \\
&\quad + g_{i(j)}((x_{j+1})_{i(j)}) + \sum_{l \neq i(j)} g_l((x_{j+1})_l) \Bigg] \\
&= \mathbb{E}_{i(j)} \Bigg[f(x_j) + \langle \nabla_{i(j)} f(x_j), (\bar{x}_{j+1} - x_j)_{i(j)} \rangle + \frac{L_{\max}}{2} \|(x_{j+1} - x_j)_{i(j)}\|^2 \\
&\quad + g_{i(j)}((x_{j+1})_{i(j)}) + \sum_{l \neq i(j)} g_l((x_j)_l) \Bigg] \\
&= f(x_j) + \frac{n-1}{n} g(x_j) + n^{-1} \left(\langle \nabla f(x_j), \bar{x}_{j+1} - x_j \rangle + \frac{L_{\max}}{2} \|\bar{x}_{j+1} - x_j\|^2 + g(\bar{x}_{j+1}) \right)
\end{aligned}$$

where we used $\mathbb{E}_{i(j)} \sum_{l \neq i(j)} g_l(x_j)_l = \frac{n-1}{n} g(x_j)$ in the last equality. By adding and subtracting a term involving $\hat{x}_j$, we obtain

$$\begin{aligned}
&\mathbb{E}_{i(j)} F(x_{j+1}) \\
\leqslant &F(x_j) + \frac{1}{n} \left(\langle \nabla f(\hat{x}_j), \bar{x}_{j+1} - x_j \rangle + \frac{L_{\max}}{2} \|\bar{x}_{j+1} - x_j\|^2 + g(\bar{x}_{j+1}) - g(x_j) \right) \\
&\quad + \frac{1}{n} \langle \nabla f(x_j) - \nabla f(\hat{x}_j), \bar{x}_{j+1} - x_j \rangle \\
\leqslant &F(x_j) + \frac{1}{n} \left(\frac{L_{\max}}{2} \|\bar{x}_{j+1} - x_j\|^2 - \frac{L_{\max}}{\gamma} \|\bar{x}_{j+1} - x_j\|^2 \right) \\
&\quad + \frac{1}{n} \langle \nabla f(x_j) - \nabla f(\hat{x}_j), \bar{x}_{j+1} - x_j \rangle \qquad \text{(from (3.19) with } x = x_j) \\
= &F(x_j) - \left(\frac{1}{\gamma} - \frac{1}{2} \right) \frac{L_{\max}}{n} \|\bar{x}_{j+1} - x_j\|^2 + \frac{1}{n} \langle \nabla f(x_j) - \nabla f(\hat{x}_j), \bar{x}_{j+1} - x_j \rangle. \qquad (3.30)
\end{aligned}$$

Consider the expectation of the last term on the right-hand side of this expression. We have

$$\begin{aligned}
&\mathbb{E}\langle \nabla f(x_j) - \nabla f(\hat{x}_j), \bar{x}_{j+1} - x_j \rangle \\
&\quad \leqslant \mathbb{E}\left(\|\nabla f(x_j) - \nabla f(\hat{x}_j)\| \|\bar{x}_{j+1} - x_j\| \right) \\
&\quad \leqslant L_{res} \mathbb{E}\left(\sum_{d \in K(j)} \|x_{d+1} - x_d\| \|\bar{x}_{j+1} - x_j\| \right) \quad \text{(from (3.22))} \\
&\quad \leqslant L_{res} \sum_{d=j-\tau}^{j-1} \frac{\rho^{(j-d)/2}}{n^{1/2}} \mathbb{E}(\|x_j - \bar{x}_{j+1}\|^2) \quad \text{(from (3.29) replace "j" by "j + 1")} \\
&\quad \leqslant n^{-1/2} L_{res} \theta \mathbb{E}(\|x_j - \bar{x}_{j+1}\|^2). \quad \text{(from (3.7))}
\end{aligned} \tag{3.31}$$

By taking expectations on both sides of (3.30) and substituting (3.31), we obtain

$$\mathbb{E}F(x_{j+1}) \leqslant \mathbb{E}F(x_j) - \frac{1}{n}\left(\left(\frac{1}{\gamma} - \frac{1}{2} \right) L_{max} - \frac{L_{res}\theta}{n^{1/2}} \right) \mathbb{E}\|\bar{x}_{j+1} - x_j\|^2.$$

To see $\left(\frac{1}{\gamma} - \frac{1}{2}\right) L_{max} - \frac{L_{res}\theta}{n^{1/2}} \geqslant 0$ or equivalently $\left(\frac{1}{\gamma} - \frac{1}{2}\right) - \frac{\Lambda\theta}{n^{1/2}} \geqslant 0$, we note from (3.7) and (3.8) that

$$\gamma^{-1} \geqslant \psi \geqslant \frac{1}{2} + \frac{\Lambda\theta}{\sqrt{n}}.$$

Therefore, we have proved the monotonicity of the expectation of the objectives, that is,

$$\mathbb{E}F(x_{j+1}) \leqslant \mathbb{E}F(x_j), \quad j = 0, 1, 2, \ldots . \tag{3.32}$$

Next we prove the sublinear convergence rate for the constrained smooth convex case in (3.11).

We have

$$
\begin{aligned}
&\|x_{j+1} - \mathcal{P}_S(x_{j+1})\|^2 \leqslant \|x_{j+1} - \mathcal{P}_S(x_j)\|^2 \\
&= \|x_j - (\Delta_j)_{i(j)} e_{i(j)} - \mathcal{P}_S(x_j)\|^2 \\
&= \|x_j - \mathcal{P}_S(x_j)\|^2 + |(\Delta_j)_{i(j)}|^2 - 2\langle (x_j - \mathcal{P}_S(x_j))_{i(j)}, (\Delta_j)_{i(j)} \rangle \\
&= \|x_j - \mathcal{P}_S(x_j)\|^2 - |(\Delta_j)_{i(j)}|^2 - 2\langle (x_j - \mathcal{P}_S(x_j))_{i(j)} - (\Delta_j)_{i(j)}, (\Delta_j)_{i(j)} \rangle \\
&= \|x_j - \mathcal{P}_S(x_j)\|^2 - |(\Delta_j)_{i(j)}|^2 + 2\langle \mathcal{P}_S(x_j) - x_{j+1})_{i(j)}, (\Delta_j)_{i(j)} \rangle \quad \text{(from (3.16))} \\
&\leqslant \|x_j - \mathcal{P}_S(x_j)\|^2 - |(\Delta_j)_{i(j)}|^2 + \\
&\quad \frac{2\gamma}{L_{\max}} \left[\langle (\mathcal{P}_S(x_j) - x_{j+1})_{i(j)}, \nabla_{i(j)} f(\hat{x}_j) \rangle + g_{i(j)}((\mathcal{P}_S(x_j))_{i(j)}) - g_{i(j)}((x_{j+1})_{i(j)}) \right] \\
&\qquad \text{(from (3.17))} \\
&= \|x_j - \mathcal{P}_S(x_j)\|^2 - |(\Delta_j)_{i(j)}|^2 + \\
&\quad \frac{2\gamma}{L_{\max}} \left[\langle (\mathcal{P}_S(x_j) - x_j)_{i(j)}, \nabla_{i(j)} f(\hat{x}_j) \rangle + g_{i(j)}((\mathcal{P}_S(x_j))_{i(j)}) - g_{i(j)}((x_{j+1})_{i(j)}) \right] + \\
&\qquad \frac{2\gamma}{L_{\max}} \left(\langle (\Delta_j)_{i(j)}, \nabla_{i(j)} f(x_j) \rangle + \langle (\Delta_j)_{i(j)}, \nabla_{i(j)} f(\hat{x}_j) - \nabla_{i(j)} f(x_j) \rangle \right) \\
&\leqslant \|x_j - \mathcal{P}_S(x_j)\|^2 - |(\Delta_j)_{i(j)}|^2 + \\
&\quad \frac{2\gamma}{L_{\max}} \left[\langle (\mathcal{P}_S(x_j) - x_j)_{i(j)}, \nabla_{i(j)} f(\hat{x}_j) \rangle + g_{i(j)}((\mathcal{P}_S(x_j))_{i(j)}) - g_{i(j)}((x_{j+1})_{i(j)}) \right] + \\
&\quad \frac{2\gamma}{L_{\max}} \left(f(x_j) - f(x_{j+1}) + \frac{L_{\max}}{2} |(\Delta_j)_{i(j)}|^2 + \langle (\Delta_j)_{i(j)}, \nabla_{i(j)} f(\hat{x}_j) - \nabla_{i(j)} f(x_j) \rangle \right) \\
&\qquad \text{(from (3.18))} \\
&= \|x_j - \mathcal{P}_S(x_j)\|^2 - (1-\gamma)|(\Delta_j)_{i(j)}|^2 + \frac{2\gamma}{L_{\max}} \underbrace{\langle (\mathcal{P}_S(x_j) - x_j)_{i(j)}, \nabla_{i(j)} f(\hat{x}_j) \rangle}_{T_1} + \\
&\quad \frac{2\gamma}{L_{\max}} \underbrace{\left[f(x_j) - f(x_{j+1}) + g_{i(j)}((\mathcal{P}_S(x_j))_{i(j)}) - g_{i(j)}((x_{j+1})_{i(j)}) \right]}_{T_3} + \\
&\quad \frac{2\gamma}{L_{\max}} \underbrace{\langle (\Delta_j)_{i(j)}, \nabla_{i(j)} f(\hat{x}_j) - \nabla_{i(j)} f(x_j) \rangle}_{T_2}. \qquad (3.33)
\end{aligned}
$$

We now seek upper bounds on the quantities T_1, T_2, and T_3 in the expectation sense. For simplicity, we construct a vector $\mathbf{b} \in \mathbb{R}^{|K(j)|}$ with $\mathbf{b}_t = \|\hat{x}_{j,t-1} - \hat{x}_{j,t}\|$. We have from elementary arguments

that

$$\mathbb{E}(\|\mathbf{b}\|^2) = \sum_{t=0}^{|K(j)|-1} \mathbb{E}(\|\hat{x}_{j,t} - \hat{x}_{j,t+1}\|^2) = \sum_{t=1}^{|K(j)|} \mathbb{E}(\|x_{K(j)_t} - x_{K(j)_t+1}\|^2)$$

$$= \sum_{d \in K(j)} \mathbb{E}(\|x_d - x_{d+1}\|^2) = \frac{1}{n} \sum_{d \in K(j)} \mathbb{E}\|x_d - \bar{x}_{d+1}\|^2 \leqslant \frac{1}{n} \sum_{d=j-\tau}^{j-1} \mathbb{E}\|x_d - \bar{x}_{d+1}\|^2$$

$$\leqslant \frac{1}{n} \sum_{t=1}^{\tau} \rho^t \mathbb{E}\|x_j - \bar{x}_{j+1}\|^2 \quad \text{(from (3.9))}$$

$$\leqslant \frac{\theta'}{n} \mathbb{E}\|x_j - \bar{x}_{j+1}\|^2 \quad \text{(from (3.7))}. \tag{3.34}$$

For the expectation of T_1, defined in (3.33), we have

$$
\begin{aligned}
\mathbb{E}(T_1) &= \mathbb{E}\left((\mathcal{P}_S(x_j) - x_j)_{i(j)} \nabla_{i(j)} f(\hat{x}_j)\right) \\
&= n^{-1}\mathbb{E}\langle \mathcal{P}_S(x_j) - x_j, \nabla f(\hat{x}_j)\rangle \\
&= n^{-1}\mathbb{E}\langle \mathcal{P}_S(x_j) - \hat{x}_j, \nabla f(\hat{x}_j)\rangle + n^{-1}\mathbb{E}\sum_{t=0}^{|K(j)|-1} \langle \hat{x}_{j,t} - \hat{x}_{j,t+1}, \nabla f(\hat{x}_j)\rangle \\
&= n^{-1}\mathbb{E}\langle \mathcal{P}_S(x_j) - \hat{x}_j, \nabla f(\hat{x}_j)\rangle \\
&\quad + n^{-1}\mathbb{E}\sum_{t=0}^{|K(j)|-1} \left(\langle \hat{x}_{j,t} - \hat{x}_{j,t+1}, \nabla f(\hat{x}_{j,t})\rangle + \langle \hat{x}_{j,t} - \hat{x}_{j,t+1}, \nabla f(\hat{x}_j) - \nabla f(\hat{x}_{j,t})\rangle\right) \\
&\leqslant n^{-1}\mathbb{E}(f_j^* - f(\hat{x}_j)) \\
&\quad + n^{-1}\mathbb{E}\sum_{t=0}^{|K(j)|-1} \left(f(\hat{x}_{j,t}) - f(\hat{x}_{j,t+1}) + \frac{L_{\max}}{2}\|\hat{x}_{j,t} - \hat{x}_{j,t+1}\|^2\right) \\
&\quad + n^{-1}\mathbb{E}\sum_{t=0}^{|K(j)|-1} \langle \hat{x}_{j,t} - \hat{x}_{j,t+1}, \nabla f(\hat{x}_j) - \nabla f(\hat{x}_{j,t})\rangle \quad \text{(from (3.3))} \\
&= n^{-1}\mathbb{E}(f_j^* - f(x_j)) + \frac{L_{\max}}{2n}\mathbb{E}\|\mathbf{b}\|^2 \\
&\quad + n^{-1}\mathbb{E}\sum_{t=0}^{|K(j)|-1} \langle \hat{x}_{j,t} - \hat{x}_{j,t+1}, \nabla f(\hat{x}_j) - \nabla f(\hat{x}_{j,t})\rangle \\
&= n^{-1}\mathbb{E}(f_j^* - f(x_j)) + \frac{L_{\max}}{2n}\mathbb{E}\|\mathbf{b}\|^2 \\
&\quad + n^{-1}\mathbb{E}\sum_{t=0}^{|K(j)|-1} \left\langle \hat{x}_{j,t} - \hat{x}_{j,t+1}, \sum_{t'=0}^{t-1} \nabla f(\hat{x}_{j,t'}) - \nabla f(\hat{x}_{j,t'+1})\right\rangle \\
&\leqslant n^{-1}\mathbb{E}(f_j^* - f(x_j)) + \frac{L_{\max}}{2n}\mathbb{E}\|\mathbf{b}\|^2 \\
&\quad + n^{-1}\mathbb{E}\sum_{t=0}^{|K(j)|-1} L_{\max}\left(\|\hat{x}_{j,t} - \hat{x}_{j,t+1}\| \sum_{t'=0}^{t-1} \|\hat{x}_{j,t'} - \hat{x}_{j,t'+1}\|\right) \\
&= n^{-1}\mathbb{E}(f_j^* - f(x_j)) + \frac{L_{\max}}{2n}\mathbb{E}\|\mathbf{b}\|^2 + n^{-1}L_{\max}\mathbb{E}\sum_{t=0}^{|K(j)|-1}\left(\mathbf{b}_{t+1}\sum_{t'=0}^{t-1}\mathbf{b}_{t'+1}\right) \\
&= n^{-1}\mathbb{E}(f_j^* - f(x_j)) + \frac{L_{\max}}{2n}\mathbb{E}\|\mathbf{b}\|^2 + \frac{L_{\max}}{2n}\mathbb{E}(\|\mathbf{b}\|_1^2 - \|\mathbf{b}\|^2) \\
&= n^{-1}\mathbb{E}(f_j^* - f(x_j)) + \frac{L_{\max}}{2n}\mathbb{E}(\|\mathbf{b}\|_1^2) \\
&\leqslant n^{-1}\mathbb{E}(f_j^* - f(x_j)) + \frac{L_{\max}\tau}{2n}\mathbb{E}(\|\mathbf{b}\|^2) \quad (\text{since } \|\mathbf{b}\|_1 \leqslant \sqrt{|K(j)|}\|\mathbf{b}\| \leqslant \sqrt{\tau}\|\mathbf{b}\|) \\
&\leqslant n^{-1}\mathbb{E}(f_j^* - f(x_j)) + \frac{L_{\max}\tau\theta'}{2n^2}\mathbb{E}(\|x_j - \bar{x}_{j+1}\|^2) \quad \text{(from (3.34))}.
\end{aligned} \tag{3.35}
$$

For the expectation of T_2, we have

$$\begin{aligned}
\mathbb{E}(T_2) &= \mathbb{E}(\Delta_j)_{i(j)} \left(\nabla_{i(j)} f(\hat{x}_j) - \nabla_{i(j)} f(x_j)\right) \\
&= n^{-1}\mathbb{E}\langle \Delta_j, \nabla f(\hat{x}_j) - \nabla f(x_j)\rangle \\
&\leqslant n^{-1}\mathbb{E}(\|\Delta_j\|\|\nabla f(\hat{x}_j) - \nabla f(x_j)\|) \\
&\leqslant \frac{L_{res}}{n}\mathbb{E}\left(\sum_{d=j-\tau}^{j-1} \|\Delta_j\|\|x_d - x_{d+1}\|\right) \qquad \text{(from (3.22))} \\
&= \frac{L_{res}}{n}\mathbb{E}\left(\sum_{d=j-\tau}^{j-1} \|x_j - \bar{x}_{j+1}\|\|x_d - x_{d+1}\|\right) \\
&\leqslant \frac{L_{res}}{n^{3/2}}\sum_{d=j-\tau}^{j-1} \rho^{(j-d)/2}\mathbb{E}\|x_j - \bar{x}_{j+1}\|^2 \qquad \text{(from (3.29) with } j \text{ replacing } j-1) \\
&\leqslant \frac{L_{res}\theta}{n^{3/2}}\mathbb{E}\|x_j - \bar{x}_{j+1}\|^2 \qquad \text{(from (3.7)).}
\end{aligned} \tag{3.36}$$

For T_3, let us look the expectation of several individual terms first

$$\mathbb{E}_{i(j)} g_{i(j)}((\mathcal{P}_S(x_j))_{i(j)}) = n^{-1} g(\mathcal{P}_S(x_j)) = n^{-1} g_j^*,$$

and

$$\begin{aligned}
\mathbb{E}_{i(j)} g_{i(j)}((x_{j+1})_{i(j)}) &= \mathbb{E}_{i(j)}(g(x_{j+1}) - g(x_j) + g_{i(j)}((x_j)_{i(j)})) \\
&= \mathbb{E}_{i(j)} g(x_{j+1}) - g(x_j) + n^{-1} g(x_j) \\
&= \mathbb{E}_{i(j)} g(x_{j+1}) - \frac{n-1}{n} g(x_j).
\end{aligned}$$

Now we take the expectation on T_3 and use the equalities above to obtain:

$$\begin{aligned}
\mathbb{E}(T_3) &= \mathbb{E}f(x_j) - \mathbb{E}f(x_{j+1}) + \mathbb{E}g_{i(j)}((\mathcal{P}_S(x_j))_{i(j)}) - \mathbb{E}g_{i(j)}((x_{j+1})_{i(j)}) \\
&= \mathbb{E}f(x_j) - \mathbb{E}f(x_{j+1}) + n^{-1}\mathbb{E}g_j^* - \mathbb{E}g(x_{j+1}) + \frac{n-1}{n}\mathbb{E}g(x_j).
\end{aligned} \tag{3.37}$$

Substituting the upper bounds from (3.35), (3.36), and (3.37), we obtain

$$\begin{aligned}
\mathbb{E}\|x_{j+1} - \mathcal{P}_S(x_{j+1})\|^2 \leqslant{}& \mathbb{E}\|x_j - \mathcal{P}_S(x_j)\|^2 - (1-\gamma)\mathbb{E}|(\Delta_j)_{i(j)}|^2 \\
&+ \frac{2\gamma}{L_{max}}\left(\frac{1}{n}\mathbb{E}(f_j^* - f(x_j)) + \frac{L_{max}\tau\theta'}{2n^2}\mathbb{E}(\|x_j - \bar{x}_{j+1}\|^2)\right) \\
&+ \frac{2\gamma}{L_{max}}\left(\frac{L_{res}\theta}{n^{3/2}}\mathbb{E}\|x_j - \bar{x}_{j+1}\|^2\right) \\
&+ \frac{2\gamma}{L_{max}}\left(\mathbb{E}f(x_j) - \mathbb{E}f(x_{j+1}) + n^{-1}\mathbb{E}g_j^* - \mathbb{E}g(x_{j+1}) + \frac{n-1}{n}\mathbb{E}g(x_j)\right).
\end{aligned}$$

By using

$$\mathbb{E}_{i(j)}(|(\Delta_j)_{i(j)}|^2) = n^{-1}\|x_j - \bar{x}_{j+1}\|^2,$$

it follows that

$$\begin{aligned}
\mathbb{E}\|x_{j+1} - \mathcal{P}_S(x_{j+1})\|^2 &\leqslant \mathbb{E}\|x_j - \mathcal{P}_S(x_j)\|^2 \\
&\quad - \frac{1}{n}\left(1 - \gamma - \frac{\tau\theta'}{n}\gamma - \frac{2\Lambda\theta}{n^{1/2}}\gamma\right)\mathbb{E}\|x_j - \bar{x}_{j+1}\|^2 \\
&\quad + \frac{2\gamma}{L_{\max} n}(\mathbb{E}f_j^* - \mathbb{E}f(x_j) + \mathbb{E}g_j^*) \\
&\quad + \frac{2\gamma}{L_{\max}}(\mathbb{E}f(x_j) + \frac{n-1}{n}\mathbb{E}g(x_j) - \mathbb{E}f(x_{j+1}) - \mathbb{E}g(x_{j+1})) \\
&\leqslant \mathbb{E}\|x_j - \mathcal{P}_S(x_j)\|^2 + \frac{2\gamma}{L_{\max} n}(\mathbb{E}f_j^* - \mathbb{E}f(x_j) + \mathbb{E}g_j^*) \\
&\quad + \frac{2\gamma}{L_{\max}}\left(\mathbb{E}f(x_j) + \frac{n-1}{n}\mathbb{E}g(x_j) - \mathbb{E}f(x_{j+1}) - \mathbb{E}g(x_{j+1})\right) \\
&\leqslant \mathbb{E}\|x_j - \mathcal{P}_S(x_j)\|^2 + \frac{2\gamma}{L_{\max} n}(F^* - \mathbb{E}F(x_j)) + \frac{2\gamma}{L_{\max}}(\mathbb{E}F(x_j) - \mathbb{E}F(x_{j+1}))
\end{aligned} \tag{3.38}$$

In the second inequality, we were able to drop the term involving $\mathbb{E}\|x_j - \bar{x}_{j+1}\|^2$ by using the fact that

$$1 - \gamma\left(1 + \frac{\tau\theta'}{n} + \frac{\Lambda\theta}{\sqrt{n}}\right) = 1 - \gamma\psi \geqslant 0,$$

which follows from the definition (3.7) of ψ and from the first upper bound on γ in (3.8). It follows that

$$\begin{aligned}
&\mathbb{E}\|x_{j+1} - \mathcal{P}_S(x_{j+1})\|^2 + \frac{2\gamma}{L_{\max}}(\mathbb{E}F(x_{j+1}) - F^*) \\
&\quad \leqslant \mathbb{E}\|x_j - \mathcal{P}_S(x_j)\|^2 + \frac{2\gamma}{L_{\max}}(\mathbb{E}F(x_j) - F^*) - \frac{2\gamma}{L_{\max} n}(\mathbb{E}F(x_j) - F^*)
\end{aligned} \tag{3.39}$$

Defining

$$S_j := \mathbb{E}(\|x_j - \mathcal{P}_S(x_j)\|^2) + \frac{2\gamma}{L_{\max}}\mathbb{E}(F(x_j) - F^*), \tag{3.40}$$

we have from (3.39) that

$$S_{j+1} \leqslant S_j - \frac{2\gamma}{L_{\max} n}\mathbb{E}(F(x_j) - F^*), \tag{3.41}$$

so by induction, we have

$$S_{j+1} \leqslant S_0 - \frac{2\gamma}{L_{\max} n}\sum_{t=0}^{j}(\mathbb{E}F(x_t) - F^*) \leqslant S_0 - \frac{2\gamma(j+1)}{L_{\max} n}(F(x_0) - F^*), \tag{3.42}$$

where the second inequality follows from monotonicity of $\mathbb{E}F(x_j)$ (3.32). Note that

$$S_0 := \|x_0 - \mathcal{P}_S(x_0)\|^2 + \frac{2\gamma}{L_{\max}}(F(x_0) - F^*).$$

By substituting the definition of S_{j+1} into (3.42), we obtain

$$\begin{aligned}\mathbb{E}\|x_{j+1} - \mathcal{P}_S(x_{j+1})\|^2 + \frac{2\gamma}{L_{\max}}(\mathbb{E}F(x_{j+1}) - F^*) + \frac{2\gamma(j+1)}{L_{\max} n}(\mathbb{E}F(x_{j+1}) - F^*) \\ \leqslant \|x_0 - \mathcal{P}_S(x_0)\|^2 + \frac{2\gamma}{L_{\max}}(F(x_0) - F^*).\end{aligned}$$

The sublinear convergence expression (3.11) follows when we drop the (nonnegative) first term on the left-hand side of this expression, and rearrange.

Finally, we prove the linear convergence rate (3.10) for the optimally strongly convex case. All bounds proven above continue to hold, and we make use the optimal strong convexity property in (3.2):

$$F(x_j) - F^* \geqslant \frac{l}{2}\|x_j - \mathcal{P}_S(x_j)\|^2.$$

By using this result together with some elementary manipulation, we obtain

$$\begin{aligned}F(x_j) - F^* &= \left(1 - \frac{L_{\max}}{l\gamma + L_{\max}}\right)(F(x_j) - F^*) + \frac{L_{\max}}{l\gamma + L_{\max}}(F(x_j) - F^*) \\ &\geqslant \left(1 - \frac{L_{\max}}{l\gamma + L_{\max}}\right)(F(x_j) - F^*) + \frac{L_{\max} l}{2(l\gamma + L_{\max})}\|x_j - \mathcal{P}_S(x_j)\|^2 \\ &= \frac{L_{\max} l}{2(l\gamma + L_{\max})}\left(\|x_j - \mathcal{P}_S(x_j)\|^2 + \frac{2\gamma}{L_{\max}}(F(x_j) - F^*)\right). \qquad (3.43)\end{aligned}$$

By taking expectations of both sides in this expression, and comparing with (3.40), we obtain

$$\mathbb{E}(F(x_j) - F^*) \geqslant \frac{L_{\max} l}{2(l\gamma + L_{\max})} S_j.$$

By substituting into (3.41), we obtain

$$\begin{aligned}S_{j+1} &\leqslant S_j - \left(\frac{2\gamma}{L_{\max} n}\right)\frac{L_{\max} l}{2(l\gamma + L_{\max})} S_j \\ &= \left(1 - \frac{l\gamma}{n(l\gamma + L_{\max})}\right) S_j \\ &\leqslant \left(1 - \frac{l\gamma}{n(l\gamma + L_{\max})}\right)^{j+1} S_0,\end{aligned}$$

where the last inequality follows from induction over j. We obtain (3.10) by substituting the

definition (3.40) of S_j. □

Proof of Corollary 3.2

Proof. Note that for ρ defined by (3.13), and using (3.12), we have

$$\begin{aligned}\rho^{(1+\tau)/2} &= \left(1+\frac{4e\Lambda(\tau+1)}{\sqrt{n}}\right)^{1+\tau} = \left(\left(1+\frac{4e\Lambda(\tau+1)}{\sqrt{n}}\right)^{\frac{\sqrt{n}}{4e\Lambda(\tau+1)}}\right)^{\frac{4e\Lambda(\tau+1)^2}{\sqrt{n}}} \\ &\leqslant e^{\frac{4e\Lambda(\tau+1)^2}{\sqrt{n}}} \leqslant e. \end{aligned} \tag{3.44}$$

Thus from the definition of ψ (3.7), we have that

$$\begin{aligned}\psi &= 1+\frac{\tau\theta'}{n}+\frac{2\Lambda\theta}{\sqrt{n}} \\ &\leqslant 1+\frac{\tau^2\rho^\tau}{n}+\frac{2\Lambda\tau\rho^{\tau/2}}{\sqrt{n}} \quad \left(\text{from } \theta=\sum_{t=1}^{\tau}\rho^{t/2}\leqslant\tau\rho^{\tau/2} \text{ and } \theta'=\sum_{t=1}^{\tau}\rho^t\leqslant\tau\rho^\tau\right) \\ &\leqslant 1+\frac{\tau^2e^2}{n}+\frac{2\Lambda\tau e}{\sqrt{n}} \qquad \text{(from (3.44))} \\ &\leqslant 1+\frac{1}{16}+\frac{1}{2}\leqslant 2,\end{aligned}$$

where for the second-past inequality we used (3.12) to obtain

$$\frac{\Lambda\tau e}{\sqrt{n}}\leqslant\frac{\Lambda\tau e}{4e\Lambda(\tau+1)^2}\leqslant\frac{1}{4}, \quad \frac{\tau^2e^2}{n}=\left(\frac{\tau e}{\sqrt{n}}\right)^2\leqslant\left(\frac{\Lambda\tau e}{\sqrt{n}}\right)\leqslant\frac{1}{16}.$$

Thus, the steplength parameter choice $\gamma = 1/2$ satisfies the first bound in (3.8). To show that the second bound in (3.8) holds also, we have

$$\begin{aligned}
&\frac{\sqrt{n}(1-\rho^{-1})-4}{4(1+\theta)\Lambda} \\
&\geqslant \frac{\sqrt{n}(1-\rho^{-1})}{4(1+\theta)\Lambda} - \frac{1}{2} && \text{(from } \theta \geqslant 1 \text{ and } \Lambda \geqslant 1\text{)} \\
&\geqslant \frac{\sqrt{n}(1-\rho^{-1/2})}{4(1+\theta)\Lambda} - \frac{1}{2} \\
&\geqslant \frac{\sqrt{n}(\rho^{1/2}-1)}{4(1+\theta)\rho^{1/2}\Lambda} - \frac{1}{2} \\
&\geqslant \frac{\sqrt{n}(\rho^{1/2}-1)}{4(\tau+1)\rho^{(\tau+1)/2}\Lambda} - \frac{1}{2} && \left(\text{from } (1+\theta)\rho^{\frac{1}{2}} \leqslant (1+\tau\rho^{\tau/2})\rho^{1/2} \leqslant (1+\tau)\rho^{(\tau+1)/2}\right) \\
&\geqslant \frac{4e\Lambda(\tau+1)}{4e(\tau+1)\Lambda} - \frac{1}{2} && \text{(from (3.13) and (3.44))} \\
&\geqslant 1 - \frac{1}{2} = \frac{1}{2}.
\end{aligned}$$

We can thus set $\gamma = 1/2$, and by substituting this choice into (3.10), we obtain (3.14). We obtain (3.15) by making the same substitution into (3.11). □

4.1 Overview

This chapter applies the asynchronous parallel scheme used in Chapters 2 and 3 to parallelize the randomized Kaczmarz (RK) algorithm for solving the linear system $Ax = b$. The analysis shows linear convergence under the "consistent read" assumption and indicates that nearly linear speedup can be expected if the number of processors is bounded by a constant that depends linearly on the number of rows in A and inversely on the maximum eigenvalue of A^TA. Please also refer to the original paper in Liu et al. (2014).

4.2 Introduction

Consider the problem of finding a solution to a consistent linear system

$$Ax = b, \tag{4.1}$$

where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. We denote the rows and columns of A by a_i^T and $\bar{a}_j$ respectively, and the elements of b by b_i, $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$. That is,

$$A = \begin{bmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_m^T \end{bmatrix} = \begin{bmatrix} \bar{a}_1, \bar{a}_2, \cdots, \bar{a}_n \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}.$$

Besides consistency of $Ax = b$, we assume that throughout that A has no zero rows. In fact, we assume (to simplify the analysis) that the rows of A are normalized, that is,

$$\|a_i\| = 1, \quad i = 1, 2, \ldots, m,$$

although we define the algorithm as if normalization had not been applied.

We are interested in the case in which A is extremely large and sparse. The randomized Kaczmarz (RK) is an algorithm for solving (4.1) that requires only $O(n)$ storage and has a linear (geometric) rate of convergence. In some situations, it is even more efficient than the conjugate gradient (CG) method (Strohmer and Vershynin, 2009), which forms the basis of the most popular iterative algorithms for solving large linear systems. At iteration j, the RK algorithm randomly selects a row $i(j) \in \{1, 2, \ldots, m\}$ of the linear system (the probability of choosing row i is $\|a_i\|^2/\|A\|_F^2$) and does

an orthogonal projection of the current estimate vector onto the hyperplane $a_{i(j)}^T x = b_{i(j)}$:

$$x_{j+1} = x_j - \frac{a_{i(j)}^T x_j - b_{i(j)}}{\|a_{i(j)}\|^2} a_{i(j)}. \tag{4.2}$$

This update formula can be derived also by applying the basic stochastic gradient algorithm to the objective $\frac{1}{2}\|Ax-b\|^2 = \frac{1}{2}\sum_i (a_i^T x - b_i)^2$ where $(a_i^T x - b_i)a_i$ is the stochastic gradient corresponding to a random choice of index i and $1/\|a_i\|^2$ is the steplength for that gradient estimate. The expected linear convergence rate of RK can be proved trivially as follows (Strohmer and Vershynin, 2009; Needell, 2010; Leventhal and Lewis, 2010). Denoting by $\mathcal{P}_S(x_j)$ the projection of iterate x_j onto the solution set of (4.1), we have from (4.2) that

$$\begin{aligned}\|x_{j+1} - \mathcal{P}_S(x_{j+1})\|^2 &\leqslant \left\| x_j - \frac{1}{\|a_{i(j)}\|^2} a_{i(j)}(a_{i(j)}^T x_j - b_{i(j)}) - \mathcal{P}_S(x_j) \right\|^2 \\ &= \|x_j - \mathcal{P}_S(x_j)\|^2 - \frac{1}{\|a_{i(j)}\|^2}(a_{i(j)}^T x_j - b_{i_j})^2.\end{aligned}$$

Given the probability $\|a_i\|^2/\|A\|_F^2$ of choosing i, we have by taking expectations that

$$\begin{aligned}E_{i(j)}\left[\|x_{j+1} - \mathcal{P}_S(x_{j+1})\|^2 \,|\, x_j\right] &\leqslant \|x_j - \mathcal{P}_S(x_j)\|^2 - E_{i(j)}\left[\frac{1}{\|a_{i(j)}\|^2}(a_{i(j)}^T x_j - b_{i(j)})^2\right] \\ &= \|x_j - \mathcal{P}_S(x_j)\|^2 - \frac{1}{\|A\|_F^2}\|Ax_j - b\|^2 \\ &\leqslant \left(1 - \frac{\lambda_{\min}}{\|A\|_F^2}\right)\|x_j - \mathcal{P}_S(x_j)\|^2, \end{aligned} \tag{4.3}$$

where $\lambda_{\min}$ is the smallest nonzero eigenvalue value of A^TA.

The RK update (4.2) is equivalent to one step of coordinate descent applied to the dual problem

$$\min_y \quad \frac{1}{2}\|A^T y\|^2 - b^T y,$$

(specifically, a negative gradient step in the ith component of y with steplength $1/\|a_i\|_2^2$), where the primal variables x and duals y are related through $x = A^T y$; see (Leventhal and Lewis, 2010).

We apply the same asynchronous parallel technique used in Hogwild! (Niu et al., 2011) as well as Chapters 2 and 3 to the standard RK algorithm. The unknown vector x is stored in a shared location, and all cores simultaneously run a RK process, updating x in an asynchronous fashion. Although our asynchronous parallel randomized Kaczmarz algorithm (AsyRK) can be viewed as an application of Hogwild! to the objective $\frac{1}{2}\|Ax-b\|^2$ (with a particular choice of step length), our analysis shows a linear convergence rate for AsyRK that outperforms the $1/t$ sublinear convergence rate for Hogwild!.

Our analysis also provides an indication of the maximum number of cores that can be involved in the computation while still yielding approximately linear speedup. This "bound" is expressed in terms of the number of equations m and the maximal eigenvalue of $A^T A$.

An outline of the remainder of the chapter is as follows. Section 4.3 illustrates details of the AsyRK algorithm. The convergence rate of AsyRK is described in Section 4.4, with proofs given in Appendix. Some simple experiments illustrate linear speedup in Section 4.6. We discuss extensions to the inconsistent case in Section 4.7, and make some concluding observations in Section 4.8.

Notation and Assumption

We use the following notation.

- $\|x\|_0$ denotes the cardinality or "ℓ_0 norm" of the vector x, that is, the number of nonzero elements in x.
- $\|X\|$ is the spectral norm of the matrix X, while $\|X\|_F$ is the Frobenius norm.
- P_t is the square $n \times n$ matrix of all zeros, except for a 1 in the (t,t) position.
- Several quantities characterize the rows and columns of A: $\theta_i := \|a_i\|_0$, $\mu := \max_i \|a_i\|_0$, $\nu := \max_j \|\bar{a}_j\|_0$.
- $\alpha := \max_{i,t} \|A\theta_i P_t a_i\|$. One can verify that $\alpha \leqslant \sqrt{\nu}\mu$ and $\alpha \leqslant \|A\|\mu$.
- Given $x_j \in \mathbb{R}^n$, $\mathcal{P}_S(x_j)$ denotes the projection of x_j onto the solution set of (4.1).
- The support index set of x is defined as $\mathrm{supp}(x)$.
- $\lambda_{\min}$ is defined as the minimal *nonzero* eigenvalue value of $A^T A$, while $\lambda_{\max}$ is defined as the maximal eigenvalue value of $A^T A$.

We make a few observations about $\lambda_{\max}$. If A is a matrix whose elements are i.i.d Gaussian random variables from $\mathcal{N}(0,1)$, then row-normalized, fundamental results in random matrices (Vershynin, 2011a) yield that $\lambda_{\max}$ is bounded by $\left(\frac{\sqrt{m}+\sqrt{n}}{\sqrt{n}}\right)^2 \leqslant O(1+m/n)$ with high probability. As long as m/n is bounded by a constant, $\lambda_{\max}$ is bounded by a constant as well. If A is a sparse matrix, then

$$\begin{aligned}\lambda_{\max} &:= \max_{\|y\|=1} \|A^T A y\| = \max_{\|y\|=1} \|Ay\|^2 \\ &\leqslant \max_i |\{j : \ \mathrm{supp}(a_i) \cap \mathrm{supp}(a_j) \neq \emptyset\}| \leqslant \mu\nu.\end{aligned}$$

Assumption 4.1. *Assume that*

- *The solution to* (4.1) *exists.*

- A *is row-normalized, that is,* $\|a_i\| = 1\ \forall i \in \{1, 2, \cdots, m\}$.

Note that $\|A\|_F^2 = m$ when the rows of A are normalized.

4.3 Algorithm

Each thread in our AsyRK algorithm performs the following simple steps: (1) Choose an index i randomly from $\{1, 2, \ldots, m\}$; (2) read the components of x that correspond to the nonzeros in a_i from shared memory; (3) calculate $a_i^T x - b_i$; (4) select $t \in \text{supp}(a_i)$; (5) update component t of x in the shared memory by a multiple of $(a_i)_t(a_i^T x - b_i)$. In principle, no memory locking takes place during either read or write, but we assume that the reads are "consistent," according to the discussion above. (We note that inconsistent reading is expected to be rather rare in the case of sparse A, because only those elements of x that correspond to nonzero locations in a_i need to be read, and inconsistency possibly occurs only when this subset of elements is updated at least twice by other processors while it is being read.) The update to component t of x can be implemented as a unitary operation, requiring no memory locking.

Algorithm 3 gives a global, aggregated view of this multithreaded process. An iteration counter j is incremented each time x is updated by a thread. We use $k(j)$ to denote the iterate at which x was read by the thread that updated x_j to x_{j+1}. (We always have $k(j) \leqslant j$, and strict inequality holds when other threads have updated x between the time it is read and the time the update is performed by this thread.) The index $i(j) \in \{1, 2, \ldots, m\}$ denotes the row that was selected by the thread that updated x_j to x_{j+1}. The index $t(j) \in \text{supp}(a_{i(j)})$ denotes the component of x that is chosen (randomly) to be updated at iteration j. We assume that the delay between reading and update for each thread is not too long, that is,

$$k(j) \geqslant j - \tau, \tag{4.4}$$

for some integer $\tau \geqslant 1$. τ can be assumed to be similar to the number of processors that are involved in the computation. (The definition and the purpose of τ here are exactly the same as AsySCD in Section 2.3.) Note that the step depends on $\theta_{i(j)}$ (the cardinality of the chosen row) and a parameter γ which is critical to the analysis of the following sections.

4.4 Main Results

This section presents the convergence analysis for AsyRK. The key issue for AsyRK is to choose an appropriate steplength parameter γ. At an intuitive level, we would like γ to be large enough to make significant progress in the approximate gradient direction. On the other hand, we want to keep it small enough that the approximate gradient information computed at the earlier iterate $k(j)$ is still relevant when the time comes to do the update at iteration j. That is, the difference between

Algorithm 3 Asynchronous Randomized Kaczmarz Algorithm $x_{K+1} = \text{AsyRK}(A, b, x_0, \gamma, K)$

1: Given $A \in \mathbb{R}^{m\times n}$ with normalized rows a_i, $i = 1, 2, \ldots, m$, $b \in \mathbb{R}^m$;
2: Initialize $j \leftarrow 0$;
3: **while** $j \leqslant K$ **do**
4: Choose $i(j)$ from $\{1, 2, \ldots, m\}$ with equal probability;
5: Choose $t(j)$ from $\text{supp}(a_{i(j)})$ with equal probability;
6: Update $x_{j+1} \leftarrow x_j - \gamma\theta_{i(j)}P_{t(j)}a_{i(j)}(a_{i(j)}^T x_{k(j)} - b_{i(j)})$;
7: $j \leftarrow j + 1$;
8: **end while**

$x_{k(j)}$ and x_j should not be too large. Along these lines, we require the ratios of expected residuals at any two successive iterations to be bounded above and below, as follows:

$$\rho^{-1} \leqslant \frac{\mathbb{E}\|Ax_{j+1} - b\|^2}{\mathbb{E}\|Ax_j - b\|^2} \leqslant \rho,$$

where ρ is a user defined parameter, usually set to be slightly larger than 1. The steplength γ depends strongly on ρ.

We state a result about convergence of AsyRK in Algorithm 3.

Theorem 4.1. *Assume that Assumption 4.1 is satisfied. Let ρ be any number greater than 1 and define the quantity ψ as follows:*

$$\psi := \mu + \frac{2\lambda_{\max}\tau\rho^\tau}{m}. \tag{4.5}$$

Suppose the steplength parameter $\gamma > 0$ in Algorithm 3 satisfies the following three bounds:

$$\gamma \leqslant \frac{1}{\psi}, \quad \gamma \leqslant \frac{m(\rho-1)}{2\lambda_{\max}\rho^{\tau+1}}, \quad \gamma \leqslant m\sqrt{\frac{(\rho-1)}{\rho^\tau(m\alpha^2 + \lambda_{\max}^2\tau\rho^\tau)}}. \tag{4.6}$$

Then we have for any $j \geqslant 0$ that

$$\rho^{-1}\mathbb{E}(\|Ax_j - b\|^2) \leqslant \mathbb{E}(\|Ax_{j+1} - b\|^2) \leqslant \rho\mathbb{E}(\|Ax_j - b\|^2) \tag{4.7}$$

and

$$\mathbb{E}(\|x_j - \mathcal{P}_S(x_j)\|^2) \leqslant \left(1 - \frac{\lambda_{\min}\gamma}{m}(2 - \gamma\psi)\right)^j \|x_0 - \mathcal{P}_S(x_0)\|^2. \tag{4.8}$$

This theorem indicates a linear rate of convergence, outperforming the sublinear "$1/j$" convergence rate for the asynchronous stochastic gradient method Hogwild!. The key reason for this improvement is that because $a_i^T x^* - b_i = 0$ for all i, the stochastic gradient *estimates* all approach zero as x approaches x^*, a property that does not hold for general stochastic gradient algorithms.

Note that the upper bound on steplength parameter γ decreases as the bound τ on the age of the iterates increases. This dependency allows us to figure out how many threads can be executed in parallel without significantly degrading the convergence behavior.

This following corollary proposes an interesting particular choice for the parameters for which the convergence expressions become more comprehensible. The result requires a condition on the delay bound τ in terms of m and $\lambda_{\max}$.

Corollary 4.2. *Suppose that Assumption 4.1 is satisfied and that*

$$\frac{2e\lambda_{\max}(\tau+1)}{m} \leqslant 1. \tag{4.9}$$

Then if we choose

$$\rho = 1 + \frac{2e\lambda_{\max}(\tau+1)}{m} \tag{4.10}$$

and set $\gamma = 1/\psi$, where ψ is defined as in (4.5), we have that

$$\mathbb{E}(\|x_j - \mathcal{P}_S(x_j)\|^2) \leqslant \left(1 - \frac{\lambda_{\min}}{m(\mu+1)}\right)^j \|x_0 - \mathcal{P}_S(x_0)\|^2. \tag{4.11}$$

Over a span of m iterations, (4.11) implies a decrease factor of approximately $1 - \lambda_{\min}/(\mu+1)$. This rate estimate indicates that for a delay τ (and hence a number of processors) in the range implied by (4.9), the number of iterations required for convergence is not affected much by the delay, so we can expect an almost linear speedup from the multicore implementation in this regime.

We conclude this section with a high-probability estimate for convergence of $\{\|x_j - \mathcal{P}_S(x_j)\|^2\}_{j=1,2,\ldots}$.

Theorem 4.3. *Suppose that the assumptions of Corollary 4.2 hold, and that ρ and ψ are defined as there. For $\epsilon > 0$ and $\eta \in (0,1)$, if*

$$j \geqslant \frac{m(\mu+1)}{\lambda_{\min}} \left| \log \frac{\|x_0 - \mathcal{P}_S(x_0)\|^2}{\eta\epsilon} \right| \tag{4.12}$$

we have that

$$\mathbb{P}(\|x_j - \mathcal{P}_S(x_j)\|^2 \leqslant \epsilon) \geqslant 1 - \eta. \tag{4.13}$$

The proofs of all results in this section appear in Section 4.8.

4.5 Comparison

This section compares the theoretical performance of RK, AsySCD (applied to minimization of $\frac{1}{2}\|Ax - b\|^2$) and AsyRK. In Table 4.1, we show the complexities (per iteration) and convergence rates (with respect to number of iterations) of three algorithms in the first and second rows.

Table 4.1: Comparison among RK, AsySCD, and AsyRK. The quantity δ is the fraction of nonzero entries in A, while L_{res} is the maximal row norm of $A^T A$ and $L_{\max}$ is the maximal diagonal entry of $A^T A$. (We assume that the nonzeros are roughly evenly distributed in A, so that μ is a modest multiple of δn.) The first row shows the number of operations required per iteration. The linear convergence rate (in the sense of iterations) is shown in the second row. The third row shows the maximum number of cores for which linear speedup is available. The fourth row combines the preceding rows to obtain the convergence rate in the sense of running time, when the method is run on the "maximal" number of processors.

algorithms	RK	AsySCD	AsyRK
# oper. per iter.	$O(\delta n)$	$\min\{O(\delta^2 mn),\ O(n)\}$	$O(\delta n)$
rate (iteration)	$1 - \frac{\lambda_{\min}}{m}$	$1 - \frac{\lambda_{\min}}{2n L_{\max}}$	$1 - \frac{\lambda_{\min}}{m(\mu+1)}$
# processors	1	$O\left(\frac{\sqrt{n} L_{\max}}{L_{\text{res}}}\right)$	$O\left(\frac{m}{\lambda_{\max}}\right)$
rate (time)	$1 - O\left(\frac{\lambda_{\min}}{\delta mn}\right)$	$1 - O\left(\frac{\lambda_{\min}}{n^{1.5} L_{\text{res}} \min\{\delta^2 m,\ 1\}}\right)$	$1 - O\left(\frac{\lambda_{\min}}{\delta^2 n^2 \lambda_{\max}}\right)$

The third row gives the maximal possibly number of processors to parallelize three algorithms respectively. The last row computes the convergence rate in term of the operation using the possibly maximal number of processors, which can be roughly understood as the running time comparison. In reporting statistics for AsySCD, we consider two alternative implementations: (1) randomly choose a coordinate i and compute it by $(a_i A)x$, which needs $O(\delta^2 mn)$ operations per iteration; and (2) compute $A^T A := Q$ offline and randomly choose an coordinate i to compute it by $Q_{i\cdot} x$, which needs $O(n)$ operations per iteration. We report the complexity per iteration of AsySCD as the minimum of these two estimates.

We perform a comparison of convergence behavior of these three algorithms on a Gaussian random matrix A with i.i.d. elements generated from $\mathcal{N}(0, 1/n)$. All rows of A have norm approximately 1, and $\lambda_{\max}$ is approximately $1 + m/n$. For these values, the convergence rates per iteration of AsySCD and RK are similar. By comparison, the convergence rate of AsyRK seems worse than RK and AsyRK by a factor $(\mu + 1)$. This is because AsyRK only updates a single coordinate rather than all coordinates corresponding to the nonzero elements in the stochastic gradient. If we modify Algorithm 3 to updated *all* components in $\text{supp}(a_{i(j)})$ (rather than just the component $t(j)$), the convergence rate for AsyRK becomes quite similar to the other two methods, without an appreciable increase in cost.

Next we compare the parallel implementations. From the last row of Table 4.1, we see that AsyRK improves the rate of RK if $\delta mn \gg \delta^2 n^2 \lambda_{\max}$, or equivalently $m \gg \delta n \lambda_{\max}$. Assuming the Gaussian ensemble for A and that m and n are comparable (so that $\lambda_{\max} = O(1)$), there is a potential factor of improvement in runtime of $O(1/\delta)$ for AsyRK over RK. To compare AsyRK and AsySCD, we note that $\lambda_{\max} = O(L_{\text{res}})$ under the same scenario for m, n, and A. Comparing the rates (running time) in the last row of Table 4.1, we find that when the mild condition $\delta < O(n^{-1/4})$

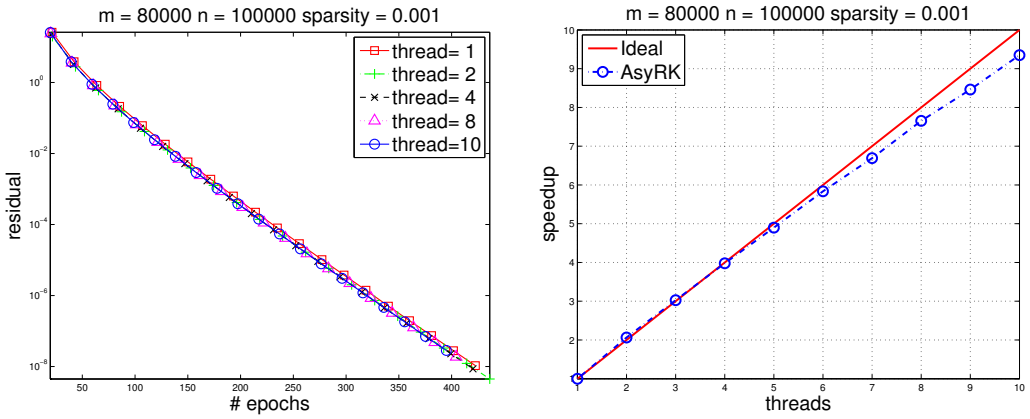

Figure 4.1: The left figure shows one line for each number of threads (=cores), plotting squared residual $\|Ax - b\|^2$ against epochs. The right figure shows the speedup over different numbers of cores.

holds, AsyRK converges much faster than AsySCD. Overall, AsyRK has a clear advantage in complexity when applied to sparse problems.

4.6 Experiments

We illustrate the behavior of AsyRK on sparse synthetic data. Our chief interest is the efficiency of multicore implementations (one thread per core), compared to a single-thread implementation.

To construct a sparse matrix $A \in \mathbb{R}^{m \times n}$, given dimensions m and n and sparsity ratio δ, we select δmn entries of A at random to be nonzero and $\mathcal{N}(0,1)$ normally distributed, and set the rest to zero. Finally, the rows of A are normalized.

Our experiments run on 1 to 10 threads on an Intel Xeon machine, with all threads sharing a single memory socket. Our implementations deviate modestly from the version of AsyRK analyzed here. First, A is partitioned into slices (row submatrices) of equal size, and each thread is assigned one slice. Each thread then selects the rows in its slice to update in order, with the order being reshuffled after each scan. This scheme essentially changes from sampling with replacement (as analyzed) to sampling without replacement, which has empirically better performance. (The same advantage is noted in implementations of Hogwild!.) The second deviation from the analyzed version is that *all* coordinates corresponding to nonzeros in the selected row $a_{i(j)}$ are updated, not just the $t(j)$ component. This scheme makes a single thread behave like $|a_{i(j)}|$ threads, thus implicitly increasing the number of cores involved in the computation. Note that this variant represents the obvious extension of randomized RK. In fact, when implemented on a single thread, it is precisely the usual randomized RK scheme.

For the plots in Figures 4.1 and 4.2, we choose $m = 80000$ and $n = 100000$, with $\delta = 0.001$, and set the steplength γ as 1 in Figure 4.1 and $\delta = 0.003$ in Figure 4.2. The left-hand graph in

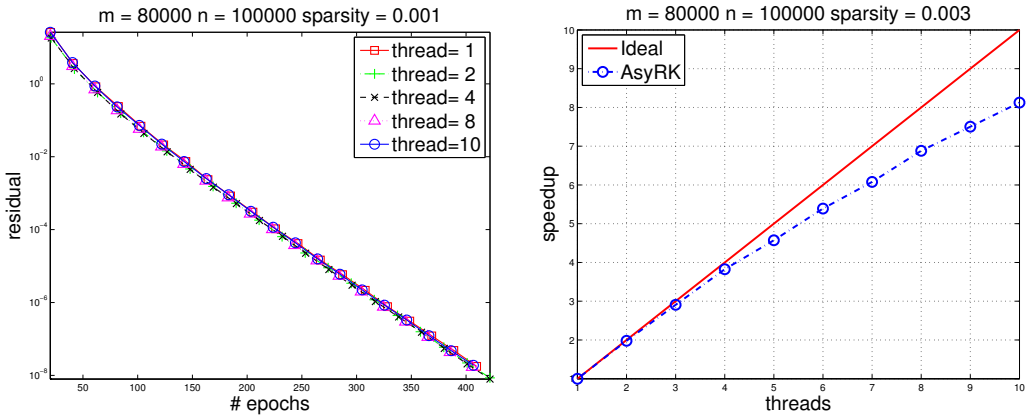

Figure 4.2: The left graph shows one line for each number of threads (=cores), plotting residual $\|Ax - b\|^2$ against epochs. The right graph shows the speedup over different numbers of cores.

each figure indicates the number of threads / cores and plots residual (defined as $\|Ax - b\|^2$) vs epoch count, where one epoch is equivalent to n iterations. Note that the curves tend to merge, indicating that the workload required for AsyRK is almost independent of the number of cores. This observation validates our result in Corollary 4.2, which indicates that provided it is below a certain threshold, the value of τ does not affect convergence rate. The right-hand graph in each figure shows speedup over different numbers of cores. Near-linear speedup is observed for $\delta = 0.001$ (Figure 4.1), while for $\delta = 0.003$ there is a dropoff for larger numbers of cores (Figure 4.2). This can perhaps be explained by the difference between our implementation from the analyzed version, in that the nonzeros in the full row $a_{i(j)}$ are updated rather than just a single element. The effect of this policy can be incorporated into the analysis roughly by increasing the value of the maximum delay parameter τ. In this case, a matrix that is three times more dense could be modeled by a value of τ that is three times larger. The effect may be to raise τ above the threshold for which linear speedup can be expected, thus explaining the (graceful) degradation in speedup for larger numbers of cores in Figure 4.2.

Next, we compare AsyRK to AsySCD in Chapter 2 on sparse synthetic data sets, on 10 cores (single socket) of the Intel Xeon. Various values of m, n, and δ are chosen for comparison in Table 4.2. A similar number of epochs is required by both algorithms, reflecting the similarity of their theoretical convergence rates; see Section 4.5. However, AsyRK is one order of magnitude faster than AsySCD to achieve the same accuracy. The main reason is that, as we showed in Table 4.1, the per-iteration complexity of AsySCD is much higher than AsyRK, for these values of the parameters.

Table 4.2: Comparison of running time and epochs between AsySCD and AsyRK on 10 cores. We report their running time and number of epochs required to attain a residual of 10^{-5}, where the residual is defined by $\|A^T(Ax-b)\|^2$ for purposes of comparison.

synthetic data			size (MB)	running time (sec)		epochs	
m	n	δ		AsySCD	AsyRK	AsySCD	AsyRK
80K	100K	0.0005	43	39.	3.6	199	195
80K	100K	0.001	84	170.	7.6	267	284
80K	100K	0.003	244	1279.	18.4	275	232
500K	1M	0.00005	282	54.	5.8	19	19
500K	1M	0.0001	550	198.	10.4	24	30
500K	1M	0.0002	1086	734.	15.0	29	31

4.7 Extension to Inconsistent Systems

Although this chapter assumes that the linear system is consistent, we can extend the algorithm described above to find the least-squares solution of inconsistent linear systems.

The minimizer of the least-squares objective $\|Ax-b\|^2$ is equivalent to the linear system $A^TAx = A^Tb$, which can be stated as the following square, consistent system of linear equations:

$$Ax - \zeta y = 0, \quad \phi A^T y = \phi A^T b, \tag{4.14}$$

for any positive values of ζ and ϕ. Similar reformulations have appeared previously in the literature; see Eggermont (1981), for example. Here we are mainly interested in the optimal values for ζ and ϕ. We can choose ζ and ϕ to maximize the critical quantity in the analysis of Algorithm 3, which is the ratio of the minimum nonzero eigenvalue of A^TA to its squared Frobenius norm. (In Theorem 4.1, this ratio appears as $\lambda_{\min}/m$, because of the normalization of the rows of A.) To show how this quantity depends on ζ and ϕ, we note first that the coefficient matrix in (4.14) is

$$\tilde{A} = \begin{bmatrix} 0 & \phi A^T \\ A & -\zeta I \end{bmatrix}.$$

Denoting the nonzero singular values of A by $\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_r > 0$ and the full SVD of A by $A = U\Sigma V^T$ where $U \in \mathbb{R}^{m\times m}$, $V \in \mathbb{R}^{n\times n}$, and $\Sigma \in \mathbb{R}^{m\times n}$, we can decompose $\tilde{A}$ as follows:

$$\begin{bmatrix} V & 0 \\ 0 & U \end{bmatrix} \begin{bmatrix} 0 & \phi\Sigma^T \\ \Sigma & -\zeta I \end{bmatrix} \begin{bmatrix} V^T & 0 \\ 0 & U^T \end{bmatrix}.$$

The singular values of $\tilde{A}$ are identical to the singular values of the center matrix, which can be

written after symmetric permutation as follows:

$$\begin{bmatrix} 0 & \phi\sigma_r & & & & & \\ \sigma_r & -\zeta & & & & & \\ & & \ddots & & & & \\ & & & 0 & \phi\sigma_1 & & \\ & & & \sigma_1 & -\zeta & & \\ & & & & & -\zeta I & \\ & & & & & & \mathbf{0} \end{bmatrix}.$$

Hence, the minimal nonzero singular value of $\tilde{A}$ is

$$\min\left\{\zeta,\ -\frac{\zeta}{2}+\frac{1}{2}\sqrt{\zeta^2+4\phi\sigma_r^2}\right\}.$$

Noting that $\|\tilde{A}\|_F^2 = (1+\phi^2)\|A\|_F^2 + m\zeta^2$, we optimize the critical ratio by finding ζ and ϕ that maximize the quantity

$$\begin{aligned} &\frac{\min\left\{\zeta,\ -\frac{\zeta}{2}+\frac{1}{2}\sqrt{\zeta^2+4\phi\sigma_r^2}\right\}^2}{(1+\phi^2)\|A\|_F^2+m\zeta^2} \\ &\qquad = \min\left\{\frac{\zeta^2}{(1+\phi^2)\|A\|_F^2+m\zeta^2},\ \frac{\left\{-\frac{\zeta}{2}+\frac{1}{2}\sqrt{\zeta^2+4\phi\sigma_r^2}\right\}^2}{(1+\phi^2)\|A\|_F^2+m\zeta^2}\right\}. \end{aligned} \tag{4.15}$$

For fixed ϕ, the first term is monotonically increasing with respect to ζ while the second term is monotonically decreasing with respect to ζ. We can thus express the optimal ζ value as $\zeta^* = \sigma_r\sqrt{\phi/2}$, which is the value for which these two terms are equal. By substituting this value into (4.15), we obtain

$$\frac{\sigma_r^2\phi}{2(1+\phi^2)\|A\|_F^2+m\sigma_r^2\phi} = \frac{\sigma_r^2}{2(1/\phi+\phi)\|A\|_F^2+m\sigma_r^2}. \tag{4.16}$$

It is clear from the last expression that the optimal value for ϕ is $\phi^* = 1$, giving the following maximal value for (4.16):

$$\frac{\sigma_r^2}{4\|A\|_F^2+m\sigma_r^2}.$$

We conclude that by normalizing the rows of A, estimating its minimum singular value σ_r, and setting $\phi = 1$ and $\zeta = \sigma_r/\sqrt{2}$, we obtain an optimally conditioned system (4.14), to which the approach of this section can be applied.

4.8 Conclusion

We have proposed a simple asynchronous parallel randomized Kaczmarz algorithm, and proved linear convergence. Our analysis also indicates the proposed method can be expected to yield near-linear speedup if the number of processors is bounded by a multiple of the number of equations in the system. Computational results, including comparison with an asynchronous stochastic coordinate descent method, confirm the effectiveness of the approach.

Appendix

This section provides proofs for our main results in Section 4.4.

In Algorithm 3, the indices $i(0), t(0), i(1), t(1), \ldots, i(j), t(j), \ldots$ are random variables. We denote the expectation over all random variables as $\mathbb{E}$, the conditional expectation with respect to $i(j)$ given $i(0), t(0), i(1), t(1), \ldots, i(j-1), t(j-1)$ as $\mathbb{E}_{i(j)}$ and the conditional expectation with respect to $t(j)$ given $i(0), t(0), i(1), t(1), \ldots, i(j-1), t(j-1), i(j)$ as $\mathbb{E}_{t(j)}$.

Proof of Theorem 4.1

Proof. We start with the following useful results, noting that the random variable $t(j)$ is distributed uniformly over the set $\mathrm{supp}(a_{i(j)})$:

$$\mathbb{E}_{t(j)}(P_{t(j)} a_{i(j)}) = \frac{1}{\theta_{i(j)}} P_{\mathrm{supp}(a_{i(j)})} a_{i(j)} = \frac{1}{\theta_{i(j)}} a_{i(j)}. \tag{4.17}$$

We prove each of the two inequalities in (4.7) by induction. We start from the right-hand inequality. First we consider the expansion of $\|Ax_{j+1} - b\|^2$ for any values of j:

$$\begin{aligned}
\|Ax_{j+1} - b\|^2 &= \|Ax_j - b\|^2 + \gamma^2 \|A\theta_{i(j)} P_{t(j)} a_{i(j)} (a_{i(j)}^T x_{k(j)} - b_{i(j)})\|^2 - \\
&\quad 2\gamma \langle Ax_j - b, A\theta_{i(j)} P_{t(j)} a_{i(j)} (a_{i(j)}^T x_{k(j)} - b_{i(j)}) \rangle \\
&= \|Ax_j - b\|^2 + \gamma^2 \underbrace{\|A\theta_{i(j)} P_{t(j)} a_{i(j)} (a_{i(j)}^T x_{k(j)} - b_{i(j)})\|^2}_{T_1} - \\
&\quad 2\gamma \underbrace{\langle Ax_{k(j)} - b, A\theta_{i(j)} P_{t(j)} a_{i(j)} (a_{i(j)}^T x_{k(j)} - b_{i(j)}) \rangle}_{T_2} + \\
&\quad 2\gamma \underbrace{\langle A(x_{k(j)} - x_j), A\theta_{i(j)} P_{t(j)} a_{i(j)} (a_{i(j)}^T x_{k(j)} - b_{i(j)}) \rangle}_{T_3}.
\end{aligned} \tag{4.18}$$

Next we consider the expectation of three terms T_1, T_2, and T_3 in (4.18). For T_1, we have

$$\begin{aligned}\mathbb{E}(T_1) &= \mathbb{E}(\|A\theta_{i(j)}P_{t(j)}a_{i(j)}(a_{i(j)}^T x_{k(j)} - b_{i(j)})\|^2) \\ &\leqslant \alpha^2\mathbb{E}(\|a_{i(j)}^T x_{k(j)} - b\|^2) \\ &= \frac{\alpha^2}{m}\mathbb{E}(\|Ax_{k(j)} - b\|^2). \end{aligned} \tag{4.19}$$

For T_2, we have

$$\begin{aligned}\mathbb{E}(T_2) &= \mathbb{E}\langle Ax_{k(j)} - b, A\theta_{i(j)}P_{t(j)}a_{i(j)}(a_{i(j)}^T x_{k(j)} - b_{i(j)})\rangle \\ &= \mathbb{E}\langle Ax_{k(j)} - b, A\mathbb{E}_{t(j)}(\theta_{i(j)}P_{t(j)}a_{i(j)})(a_{i(j)}^T x_{k(j)} - b_{i(j)})\rangle \\ &= \mathbb{E}\langle Ax_{k(j)} - b, Aa_{i(j)}(a_{i(j)}^T x_{k(j)} - b_{i(j)})\rangle \quad \text{(by (4.17))} \\ &= \mathbb{E}\langle A^T(Ax_{k(j)} - b), a_{i(j)}(a_{i(j)}^T x_{k(j)} - b_{i(j)})\rangle \\ &= \frac{1}{m}\mathbb{E}(\|A^T(Ax_{k(j)} - b)\|^2). \end{aligned} \tag{4.20}$$

For T_3, we have

$$\begin{aligned}
\mathbb{E}(T_3) &= \mathbb{E}\langle A(x_{k(j)} - x_j), A\theta_{i(j)} P_{t(j)} a_{i(j)} (a_{i(j)}^T x_{k(j)} - b_{i(j)})\rangle \\
&= \gamma \mathbb{E}\Big\langle A \sum_{d=k(j)}^{j-1} \theta_{i(d)} P_{t(d)} a_{i(d)} (a_{i(d)}^T x_{k(d)} - b_{i(d)}), \\
&\qquad A\theta_{i(j)} P_{t(j)} a_{i(j)} (a_{i(j)}^T x_{k(j)} - b_{i(j)})\Big\rangle \\
&= \gamma \sum_{d=k(j)}^{j-1} \mathbb{E}\Big\langle A\theta_{i(d)} \mathbb{E}_{t(d)}(P_{t(d)}) a_{i(d)} (a_{i(d)}^T x_{k(d)} - b_{i(d)}), \\
&\qquad A\theta_{i(j)} \mathbb{E}_{t(j)}(P_{t(j)}) a_{i(j)} (a_{i(j)}^T x_{k(j)} - b_{i(j)})\Big\rangle \\
&= \gamma \sum_{d=k(j)}^{j-1} \mathbb{E}\langle A a_{i(d)} (a_{i(d)}^T x_{k(d)} - b_{i(d)}), A a_{i(j)} (a_{i(j)}^T x_{k(j)} - b_{i(j)})\rangle \\
&= \gamma \sum_{d=k(j)}^{j-1} \mathbb{E}\Big\langle A\mathbb{E}_{i(d)} (a_{i(d)} (a_{i(d)}^T x_{k(d)} - b_{i(d)})), \\
&\qquad A\mathbb{E}_{i(j)} (a_{i(j)} (a_{i(j)}^T x_{k(j)} - b_{i(j)}))\Big\rangle \\
&= \frac{\gamma}{m^2} \sum_{d=k(j)}^{j-1} \mathbb{E}\langle AA^T(Ax_{k(d)} - b), AA^T(Ax_{k(j)} - b)\rangle \\
&\leqslant \frac{\gamma}{2m^2} \sum_{d=k(j)}^{j-1} \mathbb{E}(\|AA^T(Ax_{k(d)} - b)\|^2 + \|AA^T(Ax_{k(j)} - b)\|^2) \\
&\leqslant \frac{\gamma\lambda_{\max}^2}{2m^2} \sum_{d=k(j)}^{j-1} \left[\mathbb{E}(\|Ax_{k(d)} - b\|^2 + \|Ax_{k(j)} - b\|^2)\right], \qquad (4.21)
\end{aligned}$$

where the third line is from the observation that $t(d)$ and $t(j)$ are conditionally independent given $i(d)$ and $i(j)$; the fifth line uses the result that $i(d)$ only affects x_{d+1} and subsequent iterates and $k(j)$ is less than $d+1$ (so $x_{k(j)}$ and $i(d)$ are independent to each other). Combining (4.19), (4.20), (4.21), and (4.18), we obtain

$$\begin{aligned}
&\mathbb{E}(\|Ax_{j+1} - b\|^2) \\
\leqslant &\mathbb{E}(\|Ax_j - b\|^2) + \frac{\alpha^2\gamma^2}{m}\mathbb{E}(\|Ax_{k(j)} - b\|^2) - \frac{2\gamma}{m}\mathbb{E}(\|A^T(Ax_{k(j)} - b)\|^2) + \\
&\frac{\gamma^2\lambda_{\max}^2}{m^2} \sum_{d=k(j)}^{j-1} \left[\mathbb{E}(\|Ax_{k(d)} - b\|^2 + \|Ax_{k(j)} - b\|^2)\right]. \qquad (4.22)
\end{aligned}$$

We can use this bound to show that the right-hand inequality in (4.7) holds for $j = 0$. By setting

$j = 0$ in (4.22) and noting that $k(0) = 0$ and that the last summation is vacuous, we obtain

$$\begin{aligned}
&\mathbb{E}(\|Ax_1 - b\|^2) \\
\leqslant &\mathbb{E}(\|Ax_0 - b\|^2) + \frac{\gamma^2\alpha^2}{m}\mathbb{E}(\|Ax_0 - b\|^2) - \frac{2\gamma}{m}\mathbb{E}(\|A^T(Ax_0 - b)\|^2) \\
\leqslant &\|Ax_0 - b\|^2 + \frac{\gamma^2\alpha^2}{m}\|Ax_0 - b\|^2 \\
\leqslant &\left(1 + \frac{\gamma^2\alpha^2}{m}\right)\|Ax_0 - b\|^2. \qquad (4.23)
\end{aligned}$$

From the third bound in (4.6), we have

$$1 + \frac{\gamma^2\alpha^2}{m} \leqslant 1 + \frac{m(\rho-1)\alpha^2}{\rho^\tau(m\alpha^2 + \lambda_{\max}^2\tau\rho^\tau)} \leqslant 1 + \frac{m(\rho-1)\alpha^2}{\rho^\tau m\alpha^2} \leqslant 1 + (\rho - 1) = \rho,$$

where the third inequality follows from $\rho > 1$. By substituting into (4.23), we obtain $\mathbb{E}(\|x_0 - \mathcal{P}_S(x_0)\|^2) \leqslant \rho\mathbb{E}(\|x_1 - x_1^*\|^2)$.

For the inductive step, we use (4.22) again, assuming that the right-hand inequality in (4.7) holds up to stage j, and thus that

$$\mathbb{E}(\|Ax_{k(j)} - b\|^2) \leqslant \rho^\tau\mathbb{E}(\|Ax_j - b\|^2) \quad \text{and} \quad \mathbb{E}(\|Ax_{k(d)} - b\|^2) \leqslant \rho^{2\tau}\mathbb{E}(\|Ax_j - b\|^2)$$

provided that $0 \leqslant j - k(j) \leqslant \tau$ and $0 \leqslant j - k(d) \leqslant 2\tau$, as assumed. By substituting into the right-hand side of (4.22) again, we obtain

$$\begin{aligned}
&\mathbb{E}(\|Ax_{j+1} - b\|^2) \\
&\leqslant \mathbb{E}(\|Ax_j - b\|^2) + \frac{\alpha^2\gamma^2}{m}\mathbb{E}(\|Ax_{k(j)} - b\|^2) + \\
&\quad \frac{\gamma^2\lambda_{\max}^2}{m^2}\sum_{d=k(j)}^{j-1}\mathbb{E}(\|Ax_{k(d)} - b\|^2 + \|Ax_{k(j)} - b\|^2) \\
&\leqslant \mathbb{E}(\|Ax_j - b\|^2) + \frac{\rho^\tau\alpha^2\gamma^2}{m}\mathbb{E}(\|Ax_j - b\|^2) + \\
&\quad \frac{\gamma^2\lambda_{\max}^2}{m^2}\sum_{d=k(j)}^{j-1}\left(\rho^{2\tau}\mathbb{E}(\|Ax_j - b\|^2) + \rho^\tau\mathbb{E}(\|Ax_j - b\|^2)\right) \\
&\leqslant \mathbb{E}(\|Ax_j - b\|^2) + \frac{\rho^\tau\alpha^2\gamma^2}{m}\mathbb{E}(\|Ax_j - b\|^2) + \frac{\gamma^2\lambda_{\max}^2}{m^2}(2\tau\rho^{2\tau})\mathbb{E}(\|Ax_j - b\|^2) \\
&= \left(1 + \gamma^2\left(\frac{\rho^\tau(m\alpha^2 + 2\tau\lambda_{\max}^2\rho^\tau)}{m^2}\right)\right)\mathbb{E}(\|Ax_j - b\|^2) \\
&\leqslant (1 + (\rho - 1))\mathbb{E}(\|Ax_j - b\|^2) \\
&= \rho\mathbb{E}(\|Ax_j - b\|^2),
\end{aligned}$$

where the last inequality uses the third bound on γ from (4.6). We conclude that the right-hand side inequality in (4.7) holds for all j.

We now work on the left-hand inequality in (4.7). For all j, we have the following:

$$\begin{aligned}
&\mathbb{E}(\|Ax_{j+1} - b\|^2) \\
&= \mathbb{E}(\|Ax_j - b\|^2) + \gamma^2 \mathbb{E}(\|A\theta_{i(j)} P_{t(j)} a_{i(j)} (a_{i(j)}^T x_{k(j)} - b_{i(j)})\|^2) \\
&\qquad - 2\gamma \mathbb{E}(\langle Ax_j - b, A\theta_{i(j)} P_{t(j)} a_{i(j)} (a_{i(j)}^T x_{k(j)} - b_{i(j)})\rangle) \\
&\geqslant \mathbb{E}(\|Ax_j - b\|^2) - 2\gamma \mathbb{E}(\langle Ax_j - b, A\theta_{i(j)} P_{t(j)} a_{i(j)} (a_{i(j)}^T x_{k(j)} - b_{i(j)})\rangle) \\
&\geqslant \mathbb{E}(\|Ax_j - b\|^2) - 2\gamma \mathbb{E}(\langle Ax_j - b, A\mathbb{E}_{t(j)} (\theta_{i(j)} P_{t(j)} a_{i(j)}) (a_{i(j)}^T x_{k(j)} - b_{i(j)})\rangle) \\
&= \mathbb{E}(\|Ax_j - b\|^2) - 2\gamma \mathbb{E}\langle A^T (Ax_j - b), a_{i(j)} (a_{i(j)}^T x_{k(j)} - b_{i(j)})\rangle \quad \text{(using (4.17))} \\
&= \mathbb{E}(\|Ax_j - b\|^2) - \frac{2\gamma}{m} \mathbb{E}\langle A^T (Ax_j - b), A^T (Ax_{k(j)} - b)\rangle \\
&\geqslant E(\|Ax_j - b\|^2) - \frac{\gamma}{m} \mathbb{E}(\|A^T (Ax_j - b)\|^2 + \|A^T (Ax_{k(j)} - b)\|^2) \\
&\geqslant E(\|Ax_j - b\|^2) - \frac{\gamma \lambda_{\max}}{m} \mathbb{E}(\|Ax_j - b\|^2 + \|(Ax_{k(j)} - b\|^2) \\
&= \left(1 - \frac{\gamma \lambda_{\max}}{m}\right) \mathbb{E}(\|Ax_j - b\|^2) - \frac{\gamma \lambda_{\max}}{m} \mathbb{E}(\|Ax_{k(j)} - b\|^2). \qquad (4.24)
\end{aligned}$$

We can use this bound to show that the left-hand inequality in (4.7) holds for $j = 0$. By setting $j = 0$ in (4.22) and noting that $k(0) = 0$, we obtain

$$\mathbb{E}(\|Ax_1 - b\|^2) \geqslant \left(1 - \frac{2\gamma \lambda_{\max}}{m}\right) \mathbb{E}(\|Ax_0 - b\|^2).$$

From the second bound in (4.6), we have

$$1 - \frac{2\gamma \lambda_{\max}}{m} \geqslant 1 - \frac{\rho - 1}{\rho^{\tau+1}} = 1 - \frac{1 - \rho^{-1}}{\rho^{\tau}} \geqslant \rho^{-1},$$

where the last inequality follows from $\rho > 1$. By substituting into (4.24), we obtain $\rho^{-1}\mathbb{E}(\|Ax_0 - b\|^2) \leqslant \mathbb{E}(\|Ax_1 - b\|^2)$. For the inductive step, we use (4.24) again, assuming that the left-hand inequality in (4.7) holds up to stage j, and thus that

$$\mathbb{E}(\|Ax_j - b\|^2) \geqslant \rho^{-\tau} \mathbb{E}(\|Ax_{k(j)} - b\|^2),$$

provided that $0 \leqslant j - k(j) \leqslant \tau$, as assumed (4.4). By substituting into the left-hand side of (4.24)

again, we obtain

$$\begin{aligned}\mathbb{E}(\|Ax_{j+1}-b\|^2) &\geqslant \left(1-\frac{\gamma\lambda_{\max}}{m}\right)\mathbb{E}(\|Ax_j-b\|^2)-\frac{\gamma\rho^\tau\lambda_{\max}}{m}\mathbb{E}(\|Ax_j-b\|^2)\\ &\geqslant \left(1-\frac{2\gamma\rho^\tau\lambda_{\max}}{m}\right)\mathbb{E}(\|Ax_j-b\|^2). \end{aligned} \tag{4.25}$$

From the second bound in (4.6), we have

$$1-\frac{2\gamma\rho^\tau\lambda_{\max}}{m}\geqslant 1-\frac{\rho-1}{\rho}=\rho^{-1}.$$

We conclude that the left-hand side inequality in (4.7) holds for all j.

At this point, we have shown that both inequalities in (4.7) are satisfied for all j.

We next prove (4.8). Consider the expansion of $\|x_{j+1}-\mathcal{P}_S(x_{j+1})\|^2$:

$$\begin{aligned}\|x_{j+1}-\mathcal{P}_S(x_{j+1})\|^2 &= \|x_j-\gamma\theta_{i(j)}P_{t(j)}a_{i(j)}(a_{i(j)}^Tx_{k(j)}-b_{i(j)})-\mathcal{P}_S(x_{j+1})\|^2\\ &\leqslant \|x_j-\gamma\theta_{i(j)}P_{t(j)}a_{i(j)}(a_{i(j)}^Tx_{k(j)}-b_{i(j)})-\mathcal{P}_S(x_j)\|^2\\ &= \|x_j-\mathcal{P}_S(x_j)\|^2+\gamma^2\theta_{i(j)}^2\|P_{t(j)}a_{i(j)}(a_{i(j)}^Tx_{k(j)}-b_{i(j)})\|^2\\ &\quad -2\gamma\langle x_j-\mathcal{P}_S(x_j),\theta_{i(j)}P_{t(j)}a_{i(j)}(a_{i(j)}^Tx_{k(j)}-b_{i(j)})\rangle\\ &= \|x_j-\mathcal{P}_S(x_j)\|^2+\gamma^2\underbrace{\theta_{i(j)}^2\|P_{t(j)}a_{i(j)}(a_{i(j)}^Tx_{k(j)}-b_{i(j)})\|^2}_{T_4}-\\ &\quad 2\gamma\underbrace{\langle x_{k(j)}-\mathcal{P}_S(x_j),\theta_{i(j)}P_{t(j)}a_{i(j)}(a_{i(j)}^Tx_{k(j)}-b_{i(j)})\rangle}_{T_5}+\\ &\quad 2\gamma\underbrace{\langle x_{k(j)}-x_j,\theta_{i(j)}P_{t(j)}a_{i(j)}(a_{i(j)}^Tx_{k(j)}-b_{i(j)})\rangle}_{T_6}, \end{aligned} \tag{4.26}$$

Next, we estimate the expectations of T_4, T_5, and T_6. For T_4, we have

$$\begin{aligned}\mathbb{E}(T_4) &= \mathbb{E}(\theta_{i(j)}^2\|P_{t(j)}a_{i(j)}(a_{i(j)}^Tx_{k(j)}-b_{i(j)})\|^2)\\ &= \mathbb{E}(\theta_{i(j)}^2\mathbb{E}_{t(j)}(\|P_{t(j)}a_{i(j)}(a_{i(j)}^Tx_{k(j)}-b_{i(j)})\|^2)\\ &= \theta_{i(j)}^2\mathbb{E}\left(\frac{1}{\theta_{i(j)}}\sum_{t\in\mathrm{supp}(a_{i(j)})}(a_{i(j)}^Tx_{k(j)}-b_{i(j)})^2a_{i(j)}^TP_ta_{i(j)}\right)\\ &= \mathbb{E}(\theta_{i(j)}\|a_{i(j)}(a_{i(j)}^Tx_{k(j)}-b_{i(j)})\|^2)\\ &\leqslant \mu\mathbb{E}(a_{i(j)}^Tx_{k(j)}-b_{i(j)})^2\\ &= \frac{\mu}{m}\mathbb{E}(\|Ax_{k(j)}-b\|^2), \end{aligned} \tag{4.27}$$

For T_5, we have

$$\begin{aligned}\mathbb{E}(T_5) &= \mathbb{E}(\langle x_{k(j)} - \mathcal{P}_S(x_j), \theta_{i(j)} P_{t(j)} a_{i(j)} (a_{i(j)}^T x_{k(j)} - b_{i(j)})\rangle) \\ &= \mathbb{E}(\langle x_{k(j)} - \mathcal{P}_S(x_j), \mathbb{E}_{t(j)}(\theta_{i(j)} P_{t(j)} a_{i(j)}) (a_{i(j)}^T x_{k(j)} - b_{i(j)})\rangle) \\ &= \mathbb{E}(\langle x_{k(j)} - \mathcal{P}_S(x_j), a_{i(j)} (a_{i(j)}^T x_{k(j)} - b_{i(j)})\rangle) \\ &= \frac{1}{m}\mathbb{E}(\|Ax_{k(j)} - b\|^2). \end{aligned} \tag{4.28}$$

By following a derivation similar to (4.21) for T_6, we obtain

$$\begin{aligned}\mathbb{E}(T_6) &= \mathbb{E}(\langle x_{k(j)} - x_j, \theta_{i(j)} P_{t(j)} a_{i(j)} (a_{i(j)}^T x_{k(j)} - b_{i(j)})\rangle) \\ &= \mathbb{E}\left(\gamma \sum_{d=k(j)}^{j-1} \langle \theta_{i(d)} P_{t(d)} a_{i(d)} (a_{i(d)}^T x_{k(d)} - b_{i(d)}), \theta_{i(j)} P_{t(j)} a_{i(j)} (a_{i(j)}^T x_{k(j)} - b_{i(j)})\rangle\right) \\ &= \frac{\gamma}{m^2}\mathbb{E}\left(\sum_{d=k(j)}^{j-1} \langle A^T(Ax_{k(d)} - b), A^T(Ax_{k(j)} - b)\rangle\right) \\ &\leqslant \frac{\gamma}{m^2}\mathbb{E}\left(\sum_{d=k(j)}^{j-1} \left|\langle A^T(Ax_{k(d)} - b), A^T(Ax_{k(j)} - b)\rangle\right|\right) \\ &\leqslant \frac{\gamma}{2m^2}\mathbb{E}\left(\sum_{d=k(j)}^{j-1} \|A^T(Ax_{k(d)} - b)\|^2 + \|A^T(Ax_{k(j)} - b)\|^2\right) \\ &\leqslant \frac{\gamma\lambda_{\max}}{2m^2}\mathbb{E}\left(\sum_{d=k(j)}^{j-1} \|Ax_{k(d)} - b\|^2 + \|Ax_{k(j)} - b\|^2\right). \end{aligned} \tag{4.29}$$

Since for $d = k(j), k(j)+1, \ldots, j-1$, we have

$$k(j) - \tau \leqslant k(d) \leqslant j - 2 \leqslant k(j) + \tau - 1,$$

it follows from (4.7) that

$$\|Ax_{k(d)} - b\|^2 \leqslant \rho^\tau \|Ax_{k(j)} - b\|^2)$$

Thus from (4.29), we have

$$\mathbb{E}(T_6) \leqslant \frac{\gamma\tau\lambda_{\max}(1+\rho^\tau)}{2m^2}\mathbb{E}(\|Ax_{k(j)} - b\|^2) \leqslant \frac{\gamma\tau\lambda_{\max}\rho^\tau}{m^2}\mathbb{E}(\|Ax_{k(j)} - b\|^2). \tag{4.30}$$

By substituting (4.27), (4.28), and (4.30) into (4.26), we obtain

$$\mathbb{E}(\|x_{j+1} - \mathcal{P}_S(x_{j+1})\|^2) \leqslant \mathbb{E}(\|x_j - \mathcal{P}_S(x_j)\|^2) - \left(\frac{2\gamma - \mu\gamma^2}{m} - \frac{2\gamma^2\tau\lambda_{\max}\rho^\tau}{m^2}\right)\mathbb{E}(\|Ax_{k(j)} - b\|^2).$$

Since by (4.5) and (4.6), we have

$$\frac{2\gamma - \mu\gamma^2}{m} - \frac{2\gamma^2\tau\lambda_{\max}\rho^\tau}{m^2} = \frac{\gamma}{m}\left[2 - \gamma\left(\mu + \frac{2\tau\lambda_{\max}\rho^\tau}{m}\right)\right] = \frac{\gamma}{m}(2 - \gamma\psi) > 0,$$

we have from the bound above that

$$\begin{aligned}
\mathbb{E}(\|x_{j+1} - \mathcal{P}_S(x_{j+1})\|^2) &\leqslant \mathbb{E}(\|x_j - \mathcal{P}_S(x_j)\|^2) - \frac{\gamma}{m}(2 - \gamma\psi)\mathbb{E}(\|Ax_{k(j)} - b\|^2) \\
&\leqslant \mathbb{E}(\|x_j - \mathcal{P}_S(x_j)\|^2) - \frac{\gamma}{m}(2 - \psi\gamma)\lambda_{\min}\mathbb{E}(\|x_{k(j)} - \mathcal{P}_S(x_{k(j)})\|^2) \\
&\leqslant \left(1 - \frac{\lambda_{\min}\gamma}{m}(2 - \psi\gamma)\right)\mathbb{E}(\|x_j - \mathcal{P}_S(x_j)\|^2),
\end{aligned}$$

where the second line implies that $\mathbb{E}(\|x_j - \mathcal{P}_S(x_j)\|^2)$ is monotonically decreasing, and the last line is obtained by the implication from the second line. This completes the proof of (4.8). □

Proof of Corollary 4.2

Proof. Note first that for ρ defined by (4.10), and using (4.9), we have

$$\rho^\tau \leqslant \rho^{\tau+1} = \left[\left(1 + \frac{2e\lambda_{\max}}{m}\right)^{\frac{m}{2e\lambda_{\max}}}\right]^{\frac{2e\lambda_{\max}(\tau+1)}{m}} \leqslant e^{\frac{2e\lambda_{\max}(\tau+1)}{m}} \leqslant e.$$

Thus from the definition of ψ (4.5), and using (4.9) again, we have

$$\psi = \mu + \frac{2\lambda_{\max}\tau\rho^\tau}{m} \leqslant \mu + \frac{2e\tau\lambda_{\max}}{m} \leqslant \mu + 1. \tag{4.31}$$

We show now that the steplength parameter choice $\gamma = 1/\psi$ satisfies all the bounds in (4.6), by showing that the second and third bounds are implied by the first. For the second bound, we have

$$\frac{m(\rho - 1)}{2\lambda_{\max}\rho^{\tau+1}} \geqslant \frac{m(\rho - 1)}{2e\lambda_{\max}} \geqslant \tau + 1 \geqslant 1 \geqslant \frac{1}{\psi},$$

where the second inequality follows from (4.10) and the final inequality follows from the definition of ψ in (4.5) and the fact that $\psi > \mu \geqslant 1$.

For the third bound in (4.6), we have (by taking squares) that

$$\begin{aligned}
&m^2 \frac{(\rho-1)}{\rho^\tau(m\alpha^2+\lambda_{\max}^2\tau\rho^\tau)} \\
&= \frac{2me\lambda_{\max}(\tau+1)}{\rho^\tau(m\alpha^2+\lambda_{\max}^2\tau\rho^\tau)} && \text{(from the definition of } \rho \text{ in (4.10))} \\
&\geqslant \frac{2me\lambda_{\max}(\tau+1)}{e(m\lambda_{\max}\mu^2+\lambda_{\max}^2\tau e)} \\
&= \frac{1}{\frac{\mu^2}{2(\tau+1)}+\frac{\lambda_{\max}\tau e}{2m(\tau+1)}} \\
&\geqslant \frac{1}{\frac{\mu^2}{2(\tau+1)}+\frac{\tau}{4(\tau+1)^2}} && \text{(from the lower bound of } m \text{ in (4.9))} \\
&\geqslant \frac{1}{\frac{\mu^2}{2}+\frac{1}{16}} && \left(\text{from } \frac{\tau}{(\tau+1)^2} \leqslant \frac{1}{4}\right) \\
&\geqslant \frac{1}{\mu^2} \geqslant \frac{1}{\psi^2}.
\end{aligned}$$

We can thus set $\gamma = 1/\psi$, and by substituting this choice into (4.8) and using (4.31), we obtain (4.11). □

Proof of Theorem 4.3

Proof. From Markov's inequality, we have

$$\begin{aligned}
\mathbb{P}(\|x_j - \mathcal{P}_S(x_j)\|^2 \geqslant \epsilon) &\leqslant \epsilon^{-1}\mathbb{E}(\|x_j - \mathcal{P}_S(x_j)\|^2) \\
&\leqslant \epsilon^{-1}\left(1-\frac{\lambda_{\min}}{m(\mu+1)}\right)^j \|x_0 - \mathcal{P}_S(x_0)\|^2 \\
&\leqslant \epsilon^{-1}(1-c)^{(1/c)\left|\log\frac{\|x_0-\mathcal{P}_S(x_0)\|^2}{\eta\epsilon}\right|}\|x_0 - \mathcal{P}_S(x_0)\|^2 \quad \left(\text{with } c = \frac{\lambda_{\min}}{m(\mu+1)}\right) \\
&\leqslant \epsilon^{-1}\|x_0 - \mathcal{P}_S(x_0)\|^2 e^{-\left|\log\frac{\|x_0-\mathcal{P}_S(x_0)\|^2}{\eta\epsilon}\right|} \\
&= \eta e^{\log\frac{\|x_0-\mathcal{P}_S(x_0)\|^2}{\eta\epsilon}} e^{-\left|\log\frac{\|x_0-\mathcal{P}_S(x_0)\|^2}{\eta\epsilon}\right|} \\
&\leqslant \eta,
\end{aligned}$$

where the second inequality applies (4.11), the third inequality uses the definition of j (4.12), and the second last inequality uses the inequality $(1-c)^{1/c} \leqslant e^{-1}\ \forall c \in (0,1)$, which completes the proof. □

5.1 Overview

The randomized Kaczmarz (RK) algorithm is a simple but powerful approach for solving consistent linear systems. This chapter considers the same feasibility problem (4.1) as Chapter 4 – $Ax = b$ – but in the context of a single core and a different speedup scheme – Nesterov's accelerated scheme. The main purpose of this chapter is to improve the convergence rate of the RK algorithm for ill conditioned problems. By applying an acceleration scheme due to Nesterov to the standard RK algorithm, we obtain an accelerated randomized Kaczmarz algorithm (ARK) in Section 5.2 and show (Section 5.4) that its linear convergence is faster than the original method when the linear system has poor conditioning, as measured by the minimum nonzero eigenvalue of A^TA. The cost per iteration of both RK and ARK is $O(n)$ if the matrix A is dense. If A is sparse, however, the calculus changes. The cost of an iteration of RK is proportional to the number of nonzeros in a_i, whereas the cost of each ARK iteration is still $O(n)$ in general. We therefore propose in Section 5.3 a scheme called SARK in which the ARK updates are cached, to preserve sparsity in the intermediate vectors. (In the absence of numerical error, the iterates generated by ARK and SARK are identical.) The average cost per iteration of SARK is $O(\sqrt{\delta}n)$, where δ is the fraction of nonzero elements in A. In Section 5.4, we compare the theoretical performance of RK, ARK, and SARK for different values of the sparsity ratio δ and the minimal eigenvalue $\lambda_{\min}$, thus giving guidance about how to choose between these algorithms under various scenarios. We illustrate the computational performance of the algorithm on some random problems in Section 5.5.

Notation

We adopt the same definitions for $\lambda_{\min}$, $\lambda_{\max}$, $\|X\|$ and $\|X\|_F$ in Section 4.2 and summarize additional notations used in the remainder of the chapter below.

- X^+ is the Moore-Penrose pseudoinverse of X. Denoting the compact singular value decomposition of $X \in \mathbb{R}^{m\times n}$ as $X = U\Sigma V^T$ where U and V are orthonormal matrices (that is, $U^TU = I$ and $V^TV = I$) and Σ is nonsingular and diagonal, we have $X^+ = V\Sigma^{-1}U^T$. Note that $\lambda_{\min} = 1/\|(A^TA)^+\|$.
- Given a positive semidefinite matrix M, $\|X\|_M$ is defined as $\sqrt{\text{trace}(X^TMX)}$.
- Define $\mathcal{P}_{c,d}(x)$ as the orthogonal projection of x onto the hyperplane given by $c^Tx = d$, that is,
$$\mathcal{P}_{c,d}(x) = x - \frac{c}{\|c\|^2}(c^Tx - d).$$
- $\mathcal{P}_{A,b}(x)$ denotes the (Euclidean-norm) projection of x onto the solution set of $Ax = b$.

- $e_j \in \mathbb{R}^n$, $j = 1, 2, \ldots, n$, denotes the jth Euclidean basis vector — a vector of n zeros except for 1 in position j.

- Denote by $i(k)$ the index selected at iteration k, and note that x_k depends on all the indices selected up to iteration k, namely, $i(0), i(1), \ldots, i(k-1)$.

5.2 Algorithm

In this section, we state the randomized Kaczmarz algorithm (RK, Algorithm 4) for convenience of comparison and propose an accelerated variant called ARK (Algorithm 5). Finally, we describe an equivalent version of ARK that can be implemented with fewer operations (Algorithm 6).

Each iteration of RK randomly selects a hyperplane $a_i^T x = b_i$, for some $i \in \{1, 2, \ldots, m\}$, and obtains x_{k+1} by orthogonally projecting x_k onto this hyperplane. As shown at the start of Section 5.4 (also see Section 4.2), this algorithm guarantees linear convergence in the expectation sense.

Algorithm 4 Randomized Kaczmarz: $x_{K+1} = \mathrm{RK}(A, b, x_0, K)$

1: Initialize $k \leftarrow 0$;
2: **while** $k \leqslant K$ **do**
3: Choose $i = i(k)$ from $\{1, 2, 3, \ldots, m\}$ with equal probability;
4: Set $x_{k+1} \leftarrow \mathcal{P}_{a_i, b_i}(x_k)$, that is, $x_{k+1} = x_k - a_i(a_i^T x_k - b_i)/\|a_i\|^2$;
5: $k \leftarrow k + 1$;
6: **end while**

We note again that Step 4 does not change if we omit the normalization step, that is, if $\|a_i\| \neq 1$. However, when the rows are not normalized, our RK algorithm becomes inconsistent with the versions described in Leventhal and Lewis (2010); Strohmer and Vershynin (2009), which select the index i in Step 3 with probability $\|a_i\|_2^2/\|A\|_F^2$. We could simulate the effects of non-normalized rows by defining a matrix $\bar{A}$ in which row a_i is replaced by $\|a_i\|^2$ copies of the normalized rows $a_i/\|a_i\|$. Our Algorithm 4 applied to this virtual matrix $\bar{A}$ would then be equivalent to Algorithm 1 of Strohmer and Vershynin (2009) applied to A, with the same convergence results as in that paper (see (5.9), with $\|A\|_F^2$ replacing m in the denominator of the rate constant). Analysis of the accelerated algorithm to be discussed below could also be performed without the assumption of normalization, but the situation becomes considerably more complicated in this case. In particular, several subtle issues related to allowable scalings of A (not dealt with in existing analyses of accelerated methods) must be addressed. We believe that any additional generality to be gained by dropping our assumptions of normalization and uniform probabilities is minor, and would be obscured by the additional complication in the analysis.

The ARK algorithm applies Nesterov's accelerated procedure (Nesterov, 2004) — more familiar in the context of gradient descent for optimization — to the standard RK algorithm. When applied

to $\min_x f(x)$, gradient descent sets $x_{k+1} \leftarrow x_k - \theta_k \nabla f(x_k)$, where ∇f is the objective gradient and θ_k is the stepsize. Nesterov's accelerated procedure introduces two sequences $\{y_k\}$ and $\{v_k\}$ and defines the following iterative scheme:

$$\begin{aligned} y_k &\leftarrow \alpha_k v_k + (1-\alpha_k) x_k \\ x_{k+1} &\leftarrow y_k - \theta_k \nabla f(y_k) \\ v_{k+1} &\leftarrow \beta_k v_k + (1-\beta_k) y_k - \gamma_k \nabla f(y_k). \end{aligned}$$

With appropriate choices of α_k, β_k, and γ_k, this procedure yields better convergence rates than standard gradient descent.

If we treat the projection operation of Step 4 in Algorithm 4 analogously to the gradient descent step, we can obtain an accelerated version of the RK algorithm. This accelerated randomized Kaczmarz (ARK) procedure is detailed in Algorithm 5. The scalars α_k, β_k, and γ_k in Algorithm 5 are independent of the vector sequences $\{x_k\}$, $\{y_k\}$, and $\{v_k\}$, and can be calculated offline.

Algorithm 5 Accelerated Randomized Kaczmarz: $x_{K+1} = \mathrm{ARK}(A, b, \lambda, x_0, K)$

1: Check that $\lambda \in [0, \lambda_{\min}]$;
2: Initialize $v_0 \leftarrow x_0$, $\gamma_{-1} \leftarrow 0$, $k \leftarrow 0$;
3: **while** $k \leqslant K$ **do**
4: Choose γ_k to be the larger root of

$$\gamma_k^2 - \frac{\gamma_k}{m} = \left(1 - \frac{\gamma_k \lambda}{m}\right) \gamma_{k-1}^2; \tag{5.1}$$

5: Set α_k and β_k as follows:

$$\alpha_k \leftarrow \frac{m - \gamma_k \lambda}{\gamma_k (m^2 - \lambda)}, \tag{5.2}$$

$$\beta_k \leftarrow 1 - \frac{\gamma_k \lambda}{m}; \tag{5.3}$$

6: Set $y_k \leftarrow \alpha_k v_k + (1-\alpha_k) x_k$;
7: Choose $i = i(k)$ from $\{1, 2, 3, \ldots, m\}$ with equal probability;
8: Set $x_{k+1} \leftarrow \mathcal{P}_{a_i, b_i}(y_k)$, that is, $x_{k+1} \leftarrow y_k - a_i (a_i^T y_k - b_i)/\|a_i\|^2$;
9: Set $v_{k+1} \leftarrow \beta_k v_k + (1-\beta_k) y_k - \gamma_k a_i (a_i^T y_k - b_i)/\|a_i\|^2$;
10: $k \leftarrow k + 1$;
11: **end while**

We now describe the complexity of these methods for the case of dense A. We make the standing assumption that the quantities $\|a_i\|^2$, $i = 1, 2, \ldots, m$ are precomputed via an initial pass through the matrix. The main computation in Algorithm 4 is in step 4, which requires about $4n$ operations per iteration. The cost per iteration of Algorithm 5 is about $11n$, incurred in steps 6, 8, and 9.

Although Algorithm 5 is useful for purposes of convergence analysis of the accelerated Kaczmarz algorithm, we describe an equivalent implementation in Algorithm 6 that has a lower cost per iteration. Denoting

$$g_k := a_i(a_i^T y_k - b_i)/\|a_i\|^2,$$

we have $x_{k+1} = y_k - g_k$. Substituting for ν_k from Step 6 into Step 9, we obtain

$$\begin{aligned}\nu_{k+1} &= \beta_k \nu_k + (1-\beta_k) y_k - \gamma_k g_k \\ &= \left(\frac{\beta_k}{\alpha_k} + 1 - \beta_k\right) y_k - \beta_k \frac{1-\alpha_k}{\alpha_k} x_k - \gamma_k g_k.\end{aligned}$$

By substituting for ν_{k+1} in Step 6 of Algorithm 5, for iterate $k+1$, we obtain

$$\begin{aligned}y_{k+1} &= \alpha_{k+1}\nu_{k+1} + (1-\alpha_{k+1})x_{k+1} \\ &= \alpha_{k+1}\left(\frac{\beta_k}{\alpha_k} + 1 - \beta_k\right) y_k - \alpha_{k+1}\beta_k \frac{1-\alpha_k}{\alpha_k} x_k \\ &\quad - \alpha_{k+1}\gamma_k g_k + (1-\alpha_{k+1})(y_k - g_k) \\ &= \left[1 + \alpha_{k+1}\beta_k\left(\frac{1-\alpha_k}{\alpha_k}\right)\right] y_k - \alpha_{k+1}\beta_k \frac{1-\alpha_k}{\alpha_k} x_k \\ &\quad - (1 - \alpha_{k+1} + \alpha_{k+1}\gamma_k)\, g_k. \end{aligned} \tag{5.4}$$

From (5.2) and (5.3), we have

$$\begin{aligned}-\beta_k\left(\frac{1-\alpha_k}{\alpha_k}\right) &= \frac{m - \gamma_k\lambda}{m}\left(1 - \frac{\gamma_k(m^2-\lambda)}{m-\gamma_k\lambda}\right) \\ &= \frac{m-\gamma_k\lambda}{m} - \frac{\gamma_k(m^2-\lambda)}{m} \\ &= 1 - m\gamma_k.\end{aligned}$$

Thus, by substituting into (5.4), we obtain

$$y_{k+1} = (1-m\gamma_k)\alpha_{k+1}x_k + (1-\alpha_{k+1}+m\alpha_{k+1}\gamma_k)y_k - (1-\alpha_{k+1}+\alpha_{k+1}\gamma_k)g_k.$$

By making these substitutions into Algorithm 5, we obtain the equivalent implementation of Algorithm 6.

The main computations are in Step 6 to Step 11 which have operation counts of about $2n$, n, n, $3n$, n, and n, respectively, giving a total of $9n$. If parallel computation is possible, then Steps 6, 7, and 8 can be performed simultaneously with Step 9 (in time complexity about $3n$) while Steps 10 and 11 can be performed simultaneously (in time about n). In this setting, the total complexity can be reduced to about $4n$ — a count identical to the RK algorithm.

Algorithm 6 Efficient ARK: $x_{K+1} = \mathrm{ARK}(A, b, \lambda, x_0, K)$

Check that $\lambda \in [0, \lambda_{\min}]$;
Initialize $y_0 = x_0$, $\gamma_{-1} = 0$, $k = 0$.
Generate the sequences $\{\gamma_k : \ k = 0, 1, \ldots, K+1\}$ and $\{\alpha_k : \ k = 0, 1, \ldots, K+1\}$ as in (5.1) and (5.2);
while $k \leqslant K$ **do**
  Choose $i = i(k)$ from $\{1, 2, 3, \ldots, m\}$ with equal probability;
  Set $s_k \leftarrow (a_i^T y_k - b_i)/\|a_i\|^2$;
  Set $g_k \leftarrow s_k a_i$;
  Set $g'_k \leftarrow (1 - \alpha_{k+1} + \alpha_{k+1}\gamma_k)\, s_k a_i$;
  Set $y'_{k+1} \leftarrow (1 - m\gamma_k)\alpha_{k+1} x_k + (1 - \alpha_{k+1} + m\alpha_{k+1}\gamma_k) y_k$;
  Set $y_{k+1} \leftarrow y'_{k+1} - g'_k$;
  Set $x_{k+1} \leftarrow y_k - g_k$;
  $k \leftarrow k + 1$;
end while

5.3 Efficient Implementation for Sparse Data

This section considers the case in which the data matrix A is sparse, with a fraction of δ nonzeros (with $0 < \delta \ll 1$) and seeks an efficient implementation of Algorithm 6 for this case. We assume that the nonzeros are not concentrated in certain rows of A, that is, the sparsity of each row a_i^T is also approximately δ.

Note that the ARK approach starts at a significant disadvantage in the sparse setting. While sparsity can be exploited easily in RK — the average number of operations for each iteration of Algorithm 4 is approximately $4\delta n$ — the operation counts of the ARK algorithms remain at $O(n)$, since the vectors x_k, y_k, and v_k are dense in general. (Algorithm 6 has a count of approximately $3n + 6\delta n$ per iteration.) We now seek a modification of Algorithm 6 that "caches" the updates in order to maintain some sparsity in the update vectors, thus reducing the average complexity of each ARK iteration.

We start by writing the main updating steps in Algorithm 6 as follows:

$$s_k = (a_{i(k)}^T y_k - b_{i(k)})/\|a_{i(k)}\|^2, \tag{5.5a}$$

$$x_{k+1} = y_k - s_k a_{i(k)}, \tag{5.5b}$$

$$y_{k+1} = P_k x_k + Q_k y_k - R_k s_k a_{i(k)}, \tag{5.5c}$$

where

$$P_k = \alpha_{k+1}(1 - m\gamma_k), \tag{5.6a}$$

$$Q_k = 1 - \alpha_{k+1} + m\alpha_{k+1}\gamma_k, \tag{5.6b}$$

$$R_k = 1 - \alpha_{k+1} + \alpha_{k+1}\gamma_k. \tag{5.6c}$$

Since updating x_{k+1} and y_{k+1} is quite expensive, we only update them once on each cycle (that is, once per T iterations). We see by recursive application of (5.5) that each iterate x_{k+t}, y_{k+t} for $t \geqslant 1$ can be expressed as a linear combination of x_k and y_k, plus one other vector. The successive updates from rows $a_{i(k)}, a_{i(k+1)}, \ldots, a_{i(k+t)}$ can be "cached" in vectors z_t and w_t, so that x_{k+t} and y_{k+t} can be written as follows:

$$x_{k+t} = \rho_t x_k + \tau_t y_k + z_t, \tag{5.7a}$$

$$y_{k+t} = \sigma_t x_k + \nu_t y_k + w_t, \tag{5.7b}$$

where ρ_t, τ_t, σ_t, and ν_t are scalars. Rather than forming x_{k+t} and y_{k+t} explicitly, we could instead update the quantities ρ_t, τ_t, σ_t, ν_t, z_t, and w_t at each iteration. The advantage of doing so is that, provided t is not too large, the vectors z_t and w_t are not dense, so the cost of updating this implicit representation is usually lower than the explicit version. At some point, when t grows too large, the vectors z_t and w_t "fill in" enough that the advantages of implicit representation are lost. At this point — after T steps, say — we can store the latest vectors x_{k+T} and y_{k+T} explicitly, and start a new cycle of T iterations.

We now obtain the update formulae for the quantities ρ_t, τ_t, σ_t, ν_t, z_t, and w_t. At the starting point of a cycle, we set $t = 0$ and

$$\rho_0 = 1, \tau_0 = 0, \sigma_0 = 0, \nu_0 = 1, z_0 = w_0 = 0,$$

so that (5.7) holds for $t = 0$. In the step from iteration t to iteration $(t + 1)$ of a cycle, we have

$$x_{k+t+1} = y_{k+t} - s_{k+t}a_{i(k+t)} = \sigma_t x_k + \nu_t y_k + w_t - s_{k+t}a_{i(k+t)},$$

implying that

$$\rho_{t+1} = \sigma_t,$$

$$\tau_{t+1} = \nu_t,$$

$$z_{t+1} = w_t - s_{k+t}a_{i(k+t)}.$$

Similarly, from

$$\begin{aligned} y_{k+t+1} &= P_{k+t}x_{k+t} + Q_{k+t}y_{k+t} - R_{k+t}s_{k+t}a_{i(k+t)} \\ &= P_{k+t}(\rho_t x_k + \tau_t y_k + z_t) + Q_{k+t}(\sigma_t x_k + \nu_t y_k + w_t) - R_{k+t}s_{k+t}a_{i(k+t)} \\ &= (P_{k+t}\rho_t + Q_{k+t}\sigma_t)x_k + (P_{k+t}\tau_t + Q_{k+t}\nu_t)y_k \\ &\quad + (P_{k+t}z_t + Q_{k+t}w_t - R_{k+t}s_{k+t}a_{i(k+t)}), \end{aligned}$$

we have

$$\begin{aligned} \sigma_{t+1} &= P_{k+t}\rho_t + Q_{k+t}\sigma_t, \\ \nu_{t+1} &= P_{k+t}\tau_t + Q_{k+t}\nu_t, \\ w_{t+1} &= P_{k+t}z_t + Q_{k+t}w_t - R_{k+t}s_{k+t}a_{i(k+t)}. \end{aligned}$$

The scalar s_{k+t} can be computed from

$$\begin{aligned} s_{k+t} &= (a_{i(k+t)}^T y_{k+t} - b_{i(k+t)})/\|a_{i(k+t)}\|^2 \\ &= [a_{i(k+t)}^T(\sigma_t x_k + \nu_t y_k + w_t) - b_{i(k+t)}]/\|a_{i(k+t)}\|^2 \\ &= [\sigma_t a_{i(k+t)}^T x_k + \nu_t a_{i(k+t)}^T y_k + a_{i(k+t)}^T w_t - b_{i(k+t)}]/\|a_{i(k+t)}\|^2. \end{aligned} \tag{5.8}$$

We show this approach in full detail, for cycles of fixed length T, in Algorithm 7.

Note that w_t and z_t have nonzeros in locations where any of the vectors $a_{i(k)}, a_{i(k+1)}, \ldots, a_{i(k+t-1)}$ contain nonzeros. Thus, assuming that these vectors do not overlap significantly, and that each of them has about δn nonzeros, we can estimate that w_t and z_t have about $t\delta n$ nonzeros, in the same locations as each other. The major costs at each iteration are as follows:

- s_{k+t} costs about $6\delta n$ operations when evaluated according to (5.8), since $a_{i(k+t)}$ has about δn nonzeros.
- z_{t+1} costs about $2\delta n$ operations, for the same reason.
- w_{t+1} costs about $3t\delta n + 2\delta n$ operations, since z_t and w_t both have about δnt nonzeros, in the same locations, and $a_{i(k+t)}$ has about δn nonzeros.

The cost of updating x_k and y_k in Step 15 is about $3n + T\delta n$ each. Therefore, over a complete cycle of T iterations, we expect an approximate operation count of

$$\sum_{t=1}^{T-1} (3t\delta n + 10\delta n) + 6n + 2T\delta n \approx 1.5(T-1)T\delta n + 6n + 12T\delta n,$$

Algorithm 7 Efficient ARK for Sparse A: $x_{K+1} = \mathrm{SARK}(A, b, \lambda, x_0, T, K)$

Check that $\lambda \in [0, \lambda_{\min}]$;
Initialize $y_0 = x_0$, $\gamma_{-1} = 0$, $k = 0$;
Generate the sequences

$$\begin{aligned} P_k &= \alpha_{k+1}(1 - m\gamma_k), && k = 0, 1, \ldots, K, \\ Q_k &= 1 - \alpha_{k+1} + m\alpha_{k+1}\gamma_k, && k = 0, 1, \ldots, K, \\ R_k &= 1 - \alpha_{k+1} + \alpha_{k+1}\gamma_k, && k = 0, 1, \ldots, K; \end{aligned}$$

while $k \leqslant K$ **do**
  Set $t \leftarrow 0$, $\bar{x} \leftarrow x_k$, $\bar{y} \leftarrow y_k$, $\rho_0 \leftarrow 1$, $\tau_0 \leftarrow 0$, $\sigma_0 \leftarrow 0$, $\nu_0 \leftarrow 1$, $z_k \leftarrow 0$, $w_k \leftarrow 0$;
  while $t < T$ **do**
    if $k + t \geqslant K$ **then**
      break;
    end if
    Choose $i = i(k)$ from $\{1, 2, 3, \ldots, m\}$ with equal probability;
    Set

$$\begin{aligned} s_{k+t} &= (a_i^T(\sigma_t \bar{x} + \nu_t \bar{y} + w_t) - b_i)/\|a_i\|^2, \\ \rho_{t+1} &= \sigma_t, \\ \tau_{t+1} &= \nu_t, \\ \sigma_{t+1} &= P_{k+t}\rho_t + Q_{k+t}\sigma_t, \\ \nu_{t+1} &= P_{k+t}\tau_t + Q_{k+t}\nu_t, \\ z_{t+1} &= w_t - s_{k+t}a_i, \\ w_{t+1} &= P_{k+t}z_t + Q_{k+t}w_t - R_{k+t}s_{k+t}a_i; \end{aligned}$$

    Set $t \leftarrow t + 1$;
  end while
  Set $k \leftarrow k + T$;
  Set

$$\begin{aligned} x_k &= \rho_t \bar{x} + \tau_t \bar{y} + z_t, \\ y_k &= \sigma_t \bar{x} + \nu_t \bar{y} + w_t; \end{aligned}$$

end while

giving an approximate average cost per iteration of

$$1.5(T - 1)\delta n + \frac{6n}{T} + 12\delta n.$$

This count is minimized by setting $T = T^* = 2/\sqrt{\delta}$; for this value we obtain an average count per

iteration of $6\sqrt{\delta}n + 10.5\delta n$. This is still worse than the iteration cost for RK (which is $O(\delta n)$) but much better than that of ARK (which is $O(n)$). We show in the next section that the total number of iterations required by ARK to achieve a prescribed accuracy is lower than for RK, in general, which makes Algorithm 7 competitive in some regimes.

5.4 Convergence Rate

In this section, we study the convergence behavior of Algorithms 4, 6, and 7, estimating in particular the total number of operations required to achieve a specified level of accuracy. We also compare the approach with the conjugate gradient (CG) algorithm, applied to the "normal equations" system $A^T Ax = A^T b$. To simplify the comparisons, we assume throughout that Assumption 4.1 holds, so that $\|A\|_F^2 = m$.

The convergence of RK (Algorithm 4) is studied in (Strohmer and Vershynin, 2009; Leventhal and Lewis, 2010)[1] It is shown that

$$\mathbb{E}(\|x_{k+1} - \mathcal{P}_{A,b}(x_{k+1})\|^2) \leqslant \left(1 - \frac{\lambda_{\min}}{m}\right)^{k+1} \|x_0 - \mathcal{P}_{A,b}(x_0)\|^2, \tag{5.9}$$

where the expectation is taken over the indices $i(0), i(1), i(2), \ldots$ selected at each iteration.

For ARK, we have the following result. The proof can be found in the appendix. It is quite technical, and follows to some extent the framework developed by Nesterov (2012) for the accelerated coordinate descent method.

Theorem 5.1. *Apply* ARK *to the problem* (4.1) *with* $\lambda \in [0, \lambda_{\min}]$*, and define* $\sigma_1 = 1 + \frac{\sqrt{\lambda}}{2m}$ *and* $\sigma_2 = 1 - \frac{\sqrt{\lambda}}{2m}$*. Then we have for any* $k \geqslant 0$ *that*

$$\mathbb{E}(\|v_{k+1} - x^*\|^2_{(A^TA)^+}) \leqslant \frac{4\|x_0 - x^*\|^2_{(A^TA)^+}}{(\sigma_1^{k+1} + \sigma_2^{k+1})^2} \tag{5.10}$$

and

$$\mathbb{E}(\|x_{k+1} - x^*\|^2) \leqslant \frac{4\lambda\|x_0 - x^*\|^2_{(A^TA)^+}}{(\sigma_1^{k+1} - \sigma_2^{k+1})^2}, \tag{5.11}$$

where $x^* := \mathcal{P}_{A,b}(x_0) = x_0 + A^+(b - Ax_0)$.

Essentially, Theorem 5.1 ensures that the ARK algorithm converges in expectation to the projection of the initial point x_0 onto the affine space defined by $Ax = b$.

[1] In Strohmer and Vershynin (2009), it is required that A has full column rank, but this requirement is removed in Leventhal and Lewis (2010, Theorem 4.3), where the Hoffman constant L is equivalent to $1/\sqrt{\lambda_{\min}}$.

Theorem 5.1 shows that when $\lambda > 0$, the ARK algorithm converges at a linear rate. If the value of $\lambda = 0$, we can obtain a sublinear rate. By taking limits as $\lambda \to 0^+$ in (5.11), we have

$$\begin{aligned}
&\lim_{\lambda\to 0^+} \frac{4\lambda\|x_0 - x^*\|^2_{(A^TA)^+}}{(\sigma_1^{k+1} - \sigma_2^{k+1})^2} \\
&= \lim_{\lambda\to 0^+} \frac{4\lambda\|x_0 - x^*\|^2_{(A^TA)^+}}{\left(\left(1 + \frac{(k+1)\sqrt{\lambda}}{2m} + o(\sqrt{\lambda})\right) - \left(1 - \frac{(k+1)\sqrt{\lambda}}{2m} + o(\sqrt{\lambda})\right)\right)^2} \\
&= \lim_{\lambda\to 0^+} \frac{4\lambda\|x_0 - x^*\|^2_{(A^TA)^+}}{\left(\frac{(k+1)\sqrt{\lambda}}{m} + o(\sqrt{\lambda})\right)^2} \\
&= \frac{4m^2\|x_0 - x^*\|^2_{(A^TA)^+}}{(k+1)^2}.
\end{aligned} \tag{5.12}$$

Next, we compare convergence rates of RK, ARK, and CG. We assume further that λ is set to its optimal value $\lambda_{\min}$ in ARK. Since all algorithms converge rapidly when $\lambda_{\min}(A^TA)$ is large, we are particularly interested in the case in which $\lambda_{\min}$ is small, that is, the linear system is ill-conditioned.

Comparison between RK and ARK for Dense A

The right-hand side of the bound (5.9) decreases by a factor of $1-\lambda/m$ at each iteration. For ARK, we have that $\sigma_2^k \to 0$, so the decrease of the right-hand side is governed mainly by the behavior of the σ_1 term in the denominator. Asymptotically, we have a decrease factor per iteration of approximately

$$\sigma_1^{-2} = \left(1 + \frac{\sqrt{\lambda}}{2m}\right)^{-2} \approx 1 - \frac{\sqrt{\lambda}}{m}. \tag{5.13}$$

We conclude that for small values of λ, the ARK approach will have significantly faster linear convergence. Even if we measure convergence rate *per operation*, ARK is still faster in general, since in the implementation of Algorithm 6, it requires only twice as many operations per iteration as RK.

Comparison among RK, ARK, and SARK for Sparse A

When the coefficient matrix A is sparse, the comparisons change, because each iteration of RK costs less than each iteration of either ARK or SARK. On the other hand, fewer iterations of ARK are required to reduce the expected error below a specified tolerance. From (5.9), we deduce that the number N of iterations needed to reduce $\mathbb{E}(\|x_N - \mathcal{P}_{A,b}(x_N)\|^2)$ below a target threshold ϵ is $O((m/\lambda_{\min})|\log\epsilon|)$. We have from (5.11) and (5.13) (and ignoring a $\log\lambda$ term) that the number of iterations N of ARK and SARK needed to reduce $\mathbb{E}(\|x_k - x^*\|^2)$ below ϵ is $O((m/\sqrt{\lambda})|\log\epsilon|)$.

	Approx Operations per Iteration	Approx Iterations
RK (Algorithm 4)	$4\delta n$	$\lvert\log\epsilon\rvert(m/\lambda_{\min})$
ARK (Algorithm 6)	$3n + 6\delta n$	$\lvert\log\epsilon\rvert(m/\sqrt{\lambda})$
SARK (Algorithm 7)	$6\sqrt{\delta}n + 10.5\delta n$	$\lvert\log\epsilon\rvert(m/\sqrt{\lambda})$

Table 5.1: Operation and Iteration Counts for to achieve expected accuracy ϵ for randomized Kaczmarz variants.

Assuming approximately δn nonzeros in each row of each row of A, we summarize the operation and iteration counts for RK, ARK, and SARK in Table 5.1.

From the data in Table 5.1, and assuming that λ is set to its optimal value $\lambda_{\min}$ in the ARK and SARK algorithms, we conclude the following about the relative performance of these three approaches for various values of δ and $\lambda_{\min}$.

- RK will be approximately the best option if

$$\lambda \geqslant \max\left\{\left(\frac{4\delta}{3+6\delta}\right)^2,\ \left(\frac{4\sqrt{\delta}}{6+10.5\sqrt{\delta}}\right)^2\right\};$$

- SARK will be approximately best if

$$\lambda \leqslant \left(\frac{4\sqrt{\delta}}{6+10.5\sqrt{\delta}}\right)^2 \text{ and } \delta \leqslant 0.1;$$

- ARK will be approximately best, otherwise.

We illustrate these claims in Figure 5.1. Note that our comparison is based on approximate and worst-case analyses, which is why we claim only "approximate" superiority for each set of values in question. We can confidently say, however, RK will be superior for larger values of $\lambda_{\min}$, while ARK favors small $\lambda_{\min}$ and large δ, and SARK is superior to ARK for small values of δ. For small fixed values of $\lambda_{\min}$, RK will be superior for small values of δ, then SARK will be superior for intermediate δ values, and ARK superior for larger δ values.

Comparison among RK, SARK, and CG

We next compare RK and ARK with conjugate gradient (CG) applied to the normal-equations system $A^TAx = A^Tb$. CG is a deterministic algorithm that requires matrix-vector multiplications with the entire data matrix A and its transpose at every iteration, while RK and ARK are randomized algorithms for which each iteration requires access to just one row of A, but which require many more iterations than CG in general. CG does not require estimates of parameters such

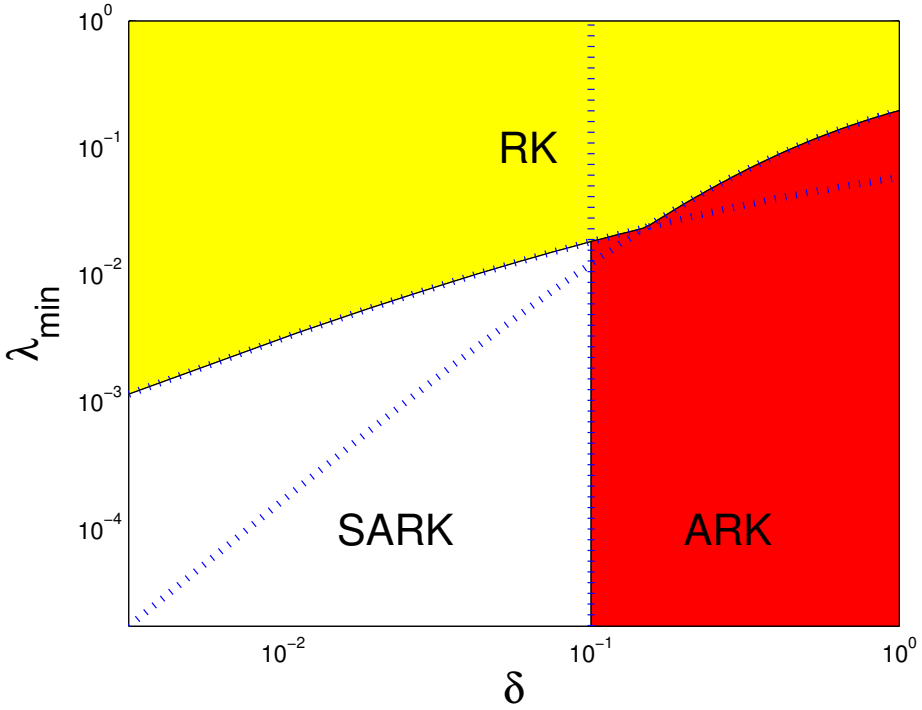

Figure 5.1: Illustration of the regions of the $(\delta, \lambda_{\min})$ space for which RK, ARK, and SARK are approximately superior (that is, $\lambda_{\min}$ in the graph). The white (yellow, red) area indicates that SARK (RK, ARK) is approximately best for the given combination of values.

as $\lambda_{\min}$ (though we show in the next section that estimation of this parameter can be incorporated into RK algorithms efficiently). Because the CG and RK approaches have very different convergence properties, and because their data access requirements are quite different, there are situations in which one or other of them will have an advantage. Here we do a simple comparison between CG and the RK methods based only on convergence rate as a function of operation count, and put aside the issues of suitability of one class or the other to various contexts and various computational platforms.

The asymptotic convergence rate for CG is

$$\|Ax_{k+1} - b\|^2 \leqslant \left(\frac{\sqrt{\lambda_{\max}} - \sqrt{\lambda_{\min}}}{\sqrt{\lambda_{\max}} + \sqrt{\lambda_{\min}}} \right)^{2(k+1)} \|Ax_0 - b\|^2. \tag{5.14}$$

(See, for example, formula (5.36) in (Nocedal and Wright, 2006).) The decrease factor per iteration is thus approximately

$$1 - 4\frac{\sqrt{\lambda_{\min}}}{\sqrt{\lambda_{\max}}}. \tag{5.15}$$

If A has sparsity δ, the cost of the main operation of CG — multiplication by A^TA — is about $4\delta mn$

operations. This is the approximate cost of m iterations of RK and about $(2/3)\sqrt{\delta}m$ iterations of SARK. Thus, for a roughly equivalent number of operations, assuming again that $\lambda = \lambda_{\min}$, we obtain the following approximate decrease factors for RK and SARK:

$$\text{RK}: \left(1 - \frac{\lambda_{\min}}{m}\right)^{m} \approx 1 - \lambda_{\min}, \tag{5.16a}$$

$$\text{ARK}: \left(1 - \frac{\sqrt{\lambda_{\min}}}{m}\right)^{(2/3)\sqrt{\delta}m} \approx 1 - \frac{2}{3}\sqrt{\delta}\sqrt{\lambda_{\min}}. \tag{5.16b}$$

By comparing (5.15) and (5.16a), we see that RK may be competitive with CG if $\sqrt{\lambda_{\min}\lambda_{\max}}$ (the geometrically averaged eigenvalue of $A^T A$) is significantly larger than 1. From (5.15) and (5.16b), we see that SARK may be competitive with CG if $\delta\lambda_{\max}$ is significantly great than 1.

We note however that the asymptotic rate (5.14) for CG is somewhat pessimistic. In practice, performance of CG depends on the distribution of the eigenvalues of $A^T A$. Rapid convergence is often seen on early iterations, as the largest eigenvalues are "resolved," but the method often settles into a steady linear rate on later iterations.

5.5 Computational Results

In this section, we study the computational behavior of RK, ARK, SARK, and CG on a variety of test problems. We start by comparing RK and ARK for dense A, then compare RK, ARK, and SARK for sparse A. Finally, we compare the randomized algorithms (RK and ARK) to the deterministic algorithm CG.

Since we need to supply the parameter λ to ARK, we introduce three ways of setting this parameter:

- ARK($\lambda_{\min}$): set $\lambda = \lambda_{\min}$. This choice gives the theoretically best convergence rate, and should be used if $\lambda_{\min}$ is known.

- ARK(0): set $\lambda = 0$. This choice requires no additional knowledge of A and guarantees convergence, though at a sublinear rate (see (5.12)).

- ARK(auto): λ determined automatically. Run RK for K_2 iterations and record x_{K_1+1} and x_{K_2+1}, where $K_2 = \lceil \frac{K}{10} \rceil$ and $K_1 = \max(1, K_2 - 10m)$. From (5.9), we can say roughly that $\mathbb{E}(\|Ax_k - b\|^2) \sim (1 - \lambda_{\min}/m)^k$, so by setting $k = K_1$ and $k = K_2$, we deduce that $\lambda_{\min}$ could be estimated by the formula

$$m\left[1 - \left(\frac{\|Ax_{K_2} - b\|}{\|Ax_{K_1} - b\|}\right)^{\frac{2}{K_2 - K_1}}\right].$$

We find that a more conservative estimate of $\lambda_{\min}$ works better in practice, in which we replace the exponent $2/(K_2 - K_2)$ by $0.5/(K_2 - K_1)$ in our experiments. If the entire matrix A cannot be obtained at one time, one could estimate $\|Ax - b\|^2$ by using a sample of the rows of $Ax - b$.

We measure performance by plotting residual error $\|Ax - b\|$ against the number of iterations and the number of operations The initial point $x_0 = 0$ is used in all algorithms.

Comparison between RK and ARK for Dense Data

Synthetic data for these tests is generated as follows: All elements of the data matrix $A \in \mathbb{R}^{m \times n}$ and the optimal solution $x^* \in \mathbb{R}^n$ are chosen to be i.i.d. $\mathcal{N}(0, 1)$. The length of all rows in A is normalized to 1. The right-hand side b is set to $b = Ax^*$. We run all algorithms 20 times (with 20 different sample sequences) and report the averaged performance.

Figures 5.2 and 5.3 show residual errors for RK and ARK with different values of λ. Figure 5.2 focuses on small problems while Figure 5.3 shows larger cases. In the graphs in the left column, the horizontal axis is iteration number, while in the right column, the horizontal axis is operation count, which is our proxy for computation cost. Operation count is obtained by scaling the number of iterations by our estimate of the average number of floating-point operations per iteration (see Table 5.1). From these figures, we observe the following.

- ARK($\lambda_{\min}$) and ARK(auto) converge much faster than RK (in both iterations and operations), except for very well conditioned problems.
- After the initial phase in which $\lambda_{\min}$ is estimated, ARK(auto) converges at about the same rate as ARK($\lambda_{\min}$).
- ARK(0) is not competitive with the other variants of ARK, but is competitive with RK on ill conditioned problems.

Comparison among RK, ARK, and SARK for Sparse Data

We compare RK, ARK(auto), and SARK on sparse data. Each element of A is set to 0 with the probability $1 - \delta$, so that the proportion of nonzero entries in A is approximately δ. The nonzero entries are chosen to be i.i.d. Gaussian $\mathcal{N}(0, 1)$, then the zeros rows are removed from A and the nonzero rows are normalized. The optimal solution x^* and right-hand side b are generated as in the dense case.

Figure 5.4 fixes $m = 1000$ and $n = 950$, and chooses $\delta = 0.8$, 0.08, and 0.01 for different levels of sparsity. For this small value of $\lambda_{\min}$ (about .0006 in all three cases), ARK/SARK outperforms RK with respect to number of iterations, as we see in the graphs in the left column of Figure 5.4.

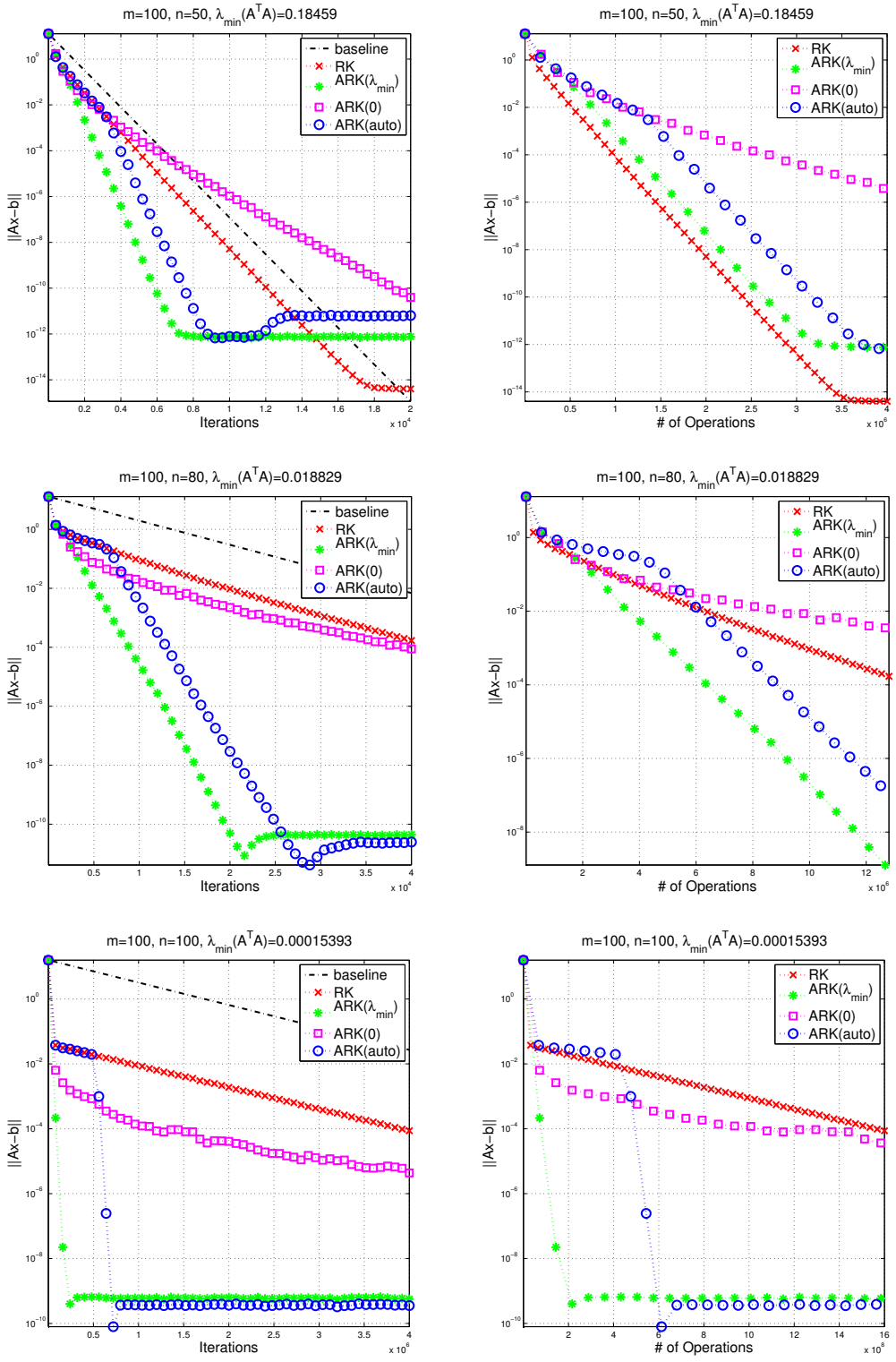

Figure 5.2: Comparison among RK, ARK($\lambda_{\min}$), ARK(0), and ARK(auto) on the dense data for $m = 100$ and $n = 50$, 80, 100. The graphs on the left (right) column plot iterations (operations) against residual error, averaged over 20 trials. The left graphs show a reference baseline sequence $\{(1 - \lambda_{\min}/m)^k : \ k = 0, 1, \dots\}$.

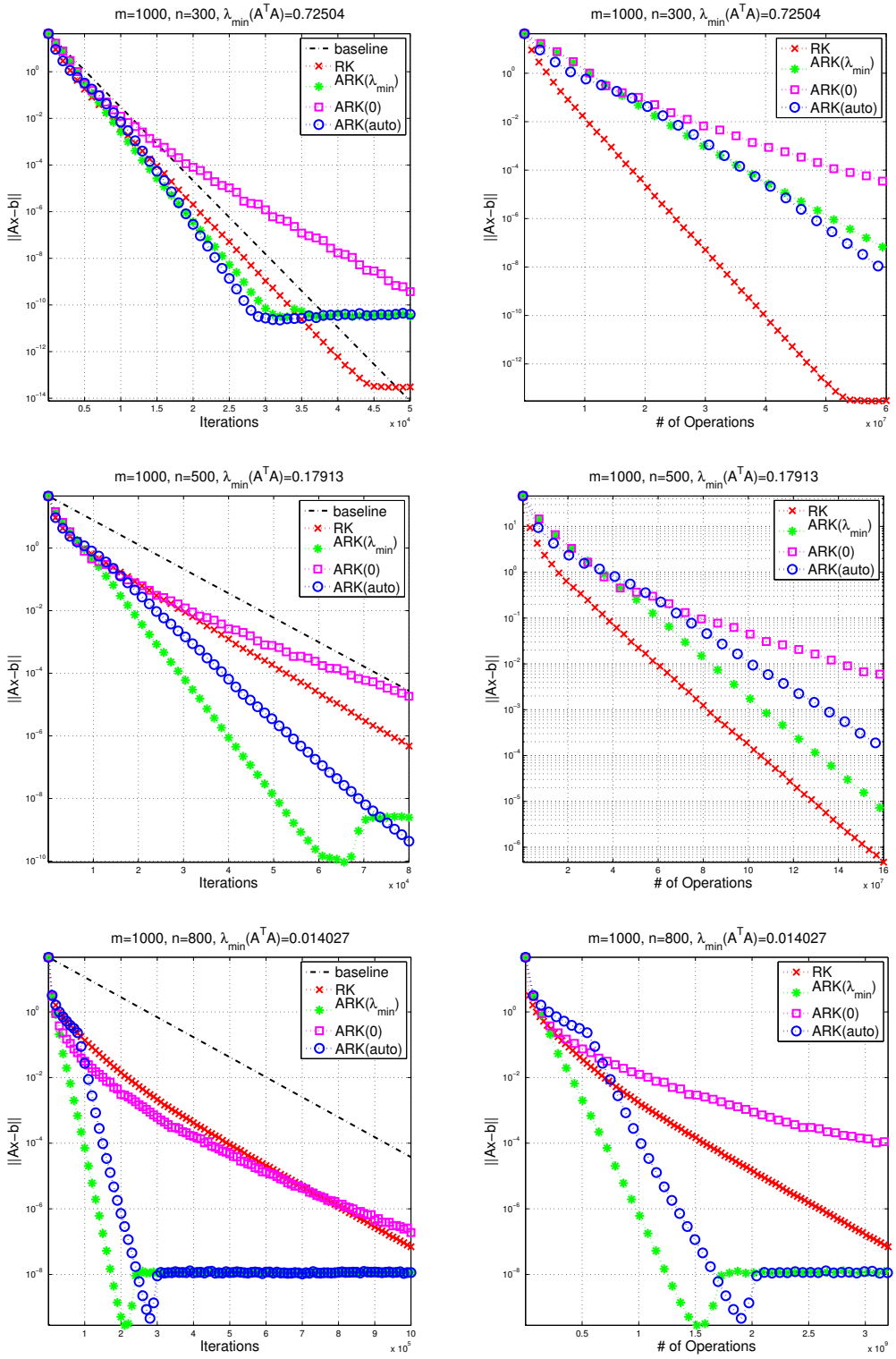

Figure 5.3: Comparison among RK, ARK($\lambda_{\min}$), ARK(0), and ARK(auto) on dense data for $m = 1000$ and $n = 300$, 500, 800. The graphs on the left (right) plot iterations (operations) against residual error, averaged over 20 trials. A reference baselien showing $\{(1 - \lambda_{\min}/m)^k : \ k = 0, 1, \dots\}$ is shown in the left plots.

For the highest density ($\delta = 0.8$; top right graph), both ARK and SARK take fewer operations than RK, and ARK is more efficient than SARK. For moderate sparsity $\delta = 0.08$ (middle right), ARK is dominated by RK in operation count, while SARK is the best option of the three. For the most sparse case ($\delta = 0.01$; bottom right), RK dominates both ARK and SARK in the number of operations. These observations are consistent with our analysis of Section 5.4.

Comparison among RK, ARK, and CG

A comparison between CG and RK was made in Strohmer and Vershynin (2009), where A is chosen to be Gaussian (elements are i.i.d. from $N(0, 1/n)$) with $m \gg n$. Problems of this type are particularly advantageous for RK. From random matrix theory (Vershynin, 2011b), we have for these matrices that $\lambda_{\min} \approx (\sqrt{m/n} - 1)^2$ and $\lambda_{\max} \approx (\sqrt{m/n} + 1)^2$, so that when $m \gg n$, we have $\sqrt{\lambda_{\min}\lambda_{\max}} \gg 1$. The convergence rates observed in (Strohmer and Vershynin, 2009) are thus consistent with our analysis of Section 5.4. We do not consider the case $m \gg n$ further here, because $\lambda_{\min}$ is large in this setting, so all algorithms converge rapidly. We focus instead on cases in which $m = n$ and A is ill conditioned.

For a given choice of $\lambda_{\min}$, we see from Section 5.4 that CG favors a smaller *maximum* eigenvalue, while RK and ARK favor a smaller *geometric average* eigenvalue. We control the distribution of eigenvalues of $A^T A$ by generating our test matrices as follows. First, find the SVD $U\Lambda V^T$ of a random $n \times n$ Gaussian matrix. Next, define an $n \times n$ diagonal matrix $\tilde{\Lambda}$ by $\tilde{\Lambda}_{ii} = i^{-\alpha}$, $i = 1, 2, \ldots, n$, for some parameter $\alpha > 0$, and compute $U\tilde{\Lambda}V^T$. Finally, normalize the rows of this matrix to obtain A. We generate x^* and b in the same way as in Section 5.5. The rows of A are normalized, so trace $(A^T A) = n$ and the average eigenvalue of $A^T A$ is 1. The parameter α controls the distribution of eigenvalues of $A^T A$; as α increases, $\lambda_{\max}$ tends to grow while $\lambda_{\min}$ shrinks.

We choose three values of α — 0.5, 0.75, and 0.9 — and fix $n = 500$ in Figure 5.5. Each row of plots in Figure 5.5 corresponds to a particular value of α, increasing from top to bottom. The left column plots the number of iterations of each method, but since the complexity of CG per iteration is $O(n^2)$ while that of other algorithms is $O(n)$, we do a rough calibration by making each iteration of CG occupy n units on the horizontal axes of the graphs in this column. We note that CG converges rapidly in its early iterations but then slows. This behavior is consistent with the analysis of CG, which shows that the asymptotic rate (5.15) is somewhat pessimistic, and that early iterations tend to behave in a manner dictated by the distribution of eigenvalues of $A^T A$ rather than the ratio of the extreme eigenvalues. Rapid initial convergence is enabled by the fact that each iteration of CG does a sweep over the entire matrix, giving it a global view of the data which is lacking in the randomized approaches. By contrast with CG the convergence of randomized algorithms is consistent and stable, and well predicted by the analysis.

As the value of α increases (that is, as we move from the top row of plots to the bottom row in Figure 5.5), we observe the following changes.

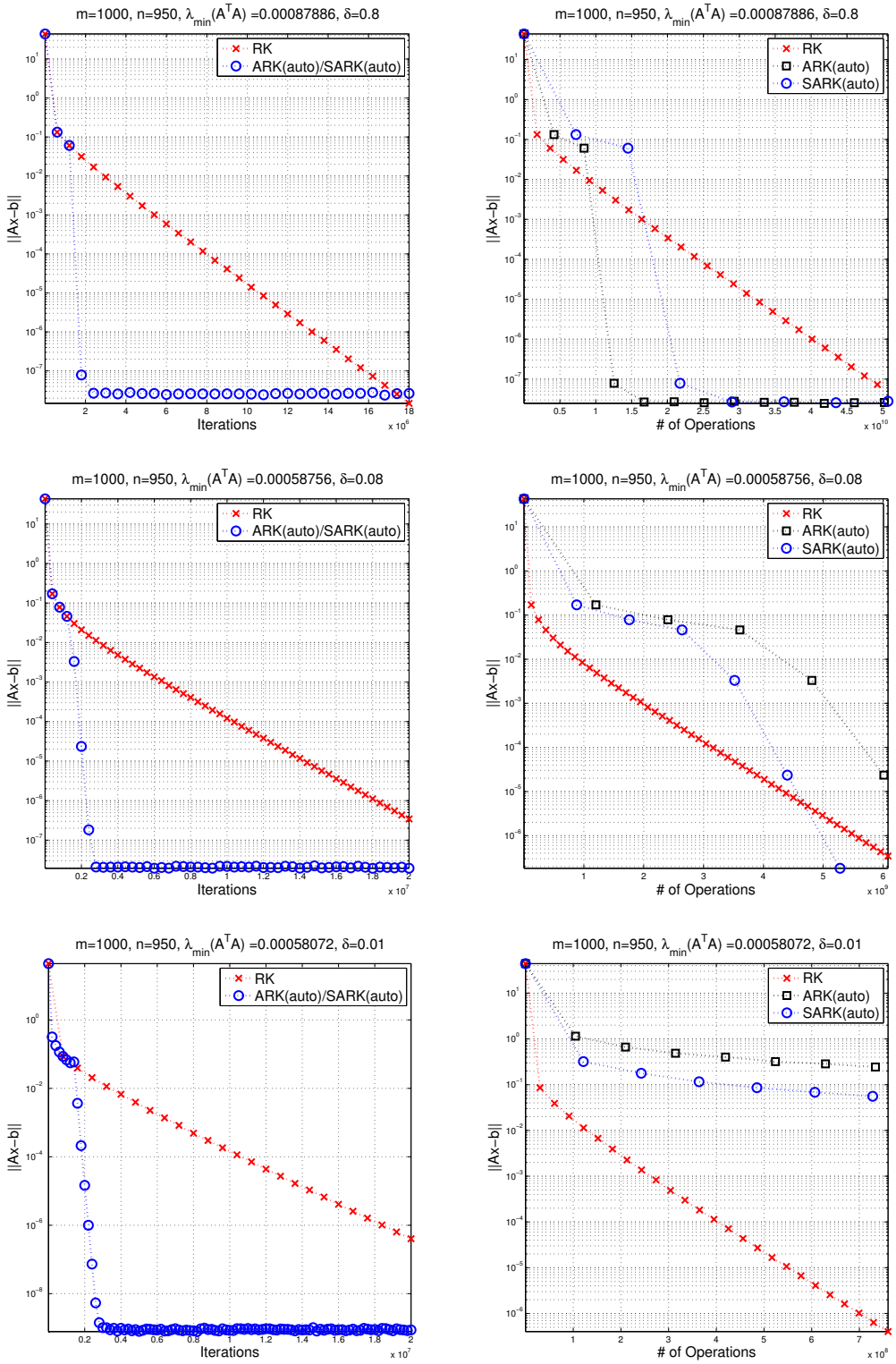

Figure 5.4: Comparison among RK, ARK(0), and SARK(auto) on sparse data with $\mathtt{m} = 1000$, $\mathtt{n} = 950$, and $\delta = 0.01$, 0.08, and 0.8. The graphs on the left (right) column plot iterations (operations) against residual errors, averaged over 20 trials.

- $\lambda_{\min}$ becomes smaller, $\lambda_{\max}$ becomes larger, and $\sqrt{\lambda_{\min}\lambda_{\max}}$ becomes smaller, as α increases.
- The asymptotic convergence rate of CG, after resolution of the leading eigenspaces, becomes slower as α increases.
- The performance of RK becomes worse compared to CG as α grows. This observation is consistent with our analysis in Section 5.4, which predicts poorer performance as $\sqrt{\lambda_{\min}\lambda_{\max}}$ decreases.
- The performance of ARK (including ARK(λ) and ARK(auto)) is comparable to CG. CG decreases faster in the beginning but ARK is better at achieving high precision. An effective hybrid strategy might be to run CG in early iterations and turn to RK or ARK in later iterations.

5.6 Conclusion

This chapter proposes an accelerated randomized Kaczmarz (ARK) algorithm with better convergence than the standard RK algorithm on ill conditioned problems. Both algorithms have similar complexities per iteration for the dense data. To deal with the sparse data, an efficient implementation of ARK, called SARK, is proposed. A comparison of convergence rates and average per-iteration complexities among RK, ARK, and SARK is provided, taking account different levels of sparseness and conditioning. Comparisons with the leading deterministic algorithm CG are also given. The analysis is validated via computational testing.

Appendix: Proof of Theorem 5.1

In proving Theorem 5.1, we refer to the particular implementation in Algorithm 5 of ARK. We assume throughout that $\|a_i\|_2 = 1$ for $i = 1, 2, \ldots, m$.

We start with two useful technical lemmas.

Lemma 5.2. *For any $y \in \mathbb{R}^n$, we have*

$$\mathbb{E}_i \left(\left\| a_i (a_i^T y - b_i) \right\|^2_{(A^T A)^+} \right) \leqslant \frac{1}{m} \|Ay - b\|^2, \tag{5.17}$$

where the random variable i follows the uniform distribution over the set $\{1, 2, \ldots, m\}$.

Proof. Define the compact singular value decomposition of A as $A = U\Sigma V^T$, where $U^T U = I$, $V^T V = I$, and Σ is positive diagonal, so that $(A^T A)^+ = V\Sigma^{-2} V^T$. Denoting $U^T = [u_1\, u_2\, \ldots\, u_m]$, it

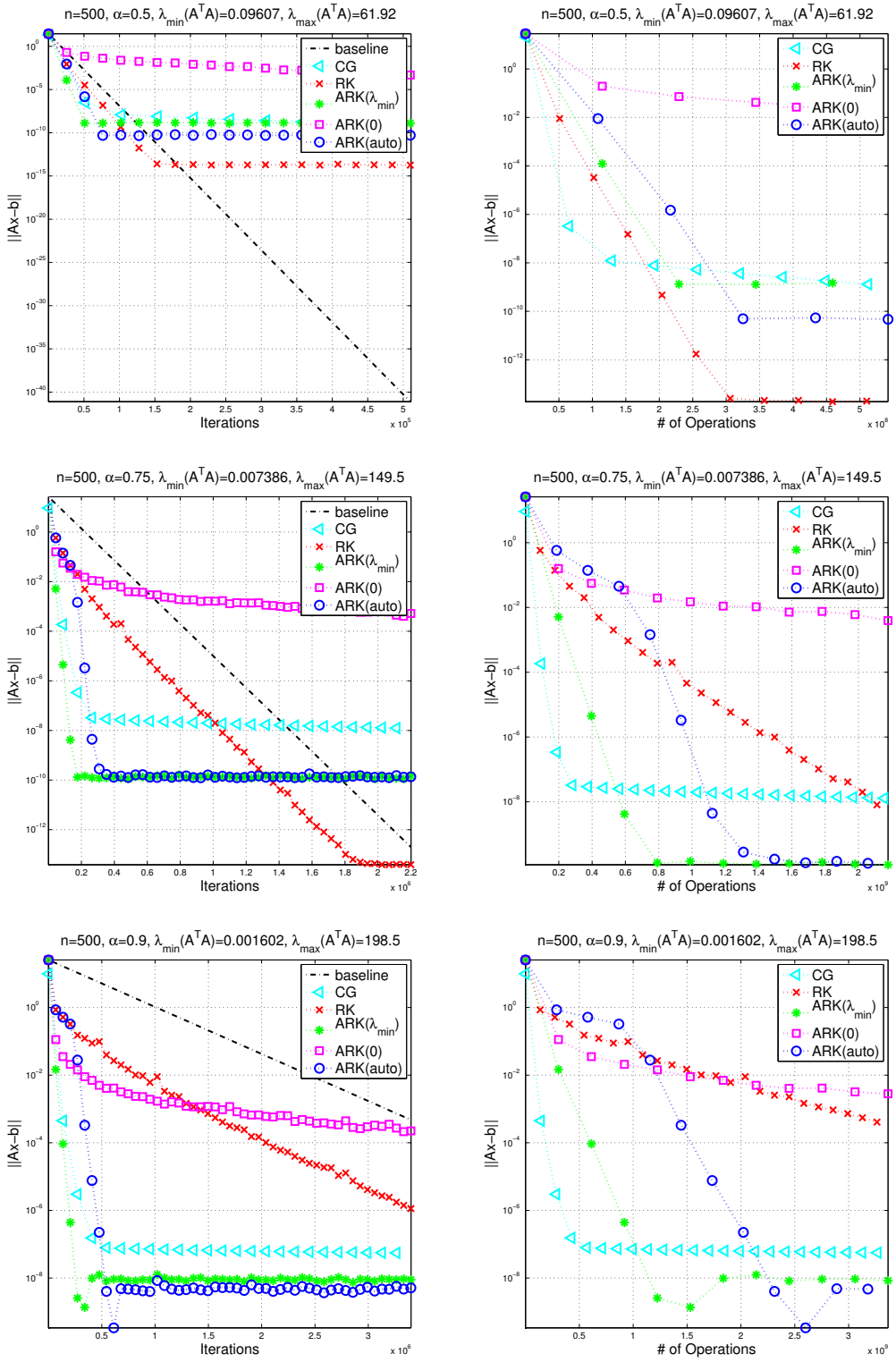

Figure 5.5: Comparison amomg CG, RK, ARK($\lambda_{\min}$), ARK(0), and ARK(auto) on dense data. The figures on the left (right) plot residual against iterations (operations). A reference baseline sequence of $\{(1-\lambda_{\min}/m)^k : \ k = 0, 1, 2, \dots\}$ is shown in the left plots.

is easy to show that $\|u_i\|_2 \leqslant 1$ for all $i = 1, 2, \ldots, m$. Using $\mathbb{E}_i$ to denote expectation with respect to the index i, we have

$$\begin{aligned}
&\mathbb{E}_i \left(\left\| a_i (a_i^T y - b_i) \right\|^2_{(A^T A)^+} \right) \\
&= \frac{1}{m} \sum_{i=1}^{m} \langle (A^T A)^+ a_i (a_i^T y - b_i),\ a_i (a_i^T y - b_i) \rangle \\
&= \frac{1}{m} \text{trace} \left[(A^T A)^+ \sum_{i=1}^{m} a_i (a_i^T y - b_i)^2 a_i^T \right] \\
&= \frac{1}{m} \text{trace} \left[(A^T A)^+ A^T \text{diag}\,(Ay - b)^2 A \right] \\
&= \frac{1}{m} \text{trace} \left[V \Sigma^{-1} U^T \text{diag}\,(Ay - b)^2 U \Sigma V^T \right] \\
&= \frac{1}{m} \text{trace} \left[U^T \text{diag}\,(Ay - b)^2 U \right] \\
&= \frac{1}{m} \| \text{diag}\,(Ay - b) U \|_F^2 \\
&= \frac{1}{m} \sum_{i=1}^{m} (a_i^T y - b)^2 \|u_i\|^2 \\
&\leqslant \frac{1}{m} \|Ay - b\|^2.
\end{aligned}$$

□

Lemma 5.3. *For any solution x^* to (4.1) and any $y \in \mathbb{R}^n$, we have*

$$\mathbb{E}_i (\|\mathcal{P}_{a_i, b_i}(y) - x^*\|^2) = \|y - x^*\|^2 - \frac{1}{m} \|Ay - b\|^2, \tag{5.18}$$

where the random variable i follows the uniform distribution over the set $\{1, 2, \ldots, m\}$.

Proof. We have

$$\begin{aligned}
&\mathbb{E}_i (\|\mathcal{P}_{a_i, b_i}(y) - x^*\|^2) \\
&= \mathbb{E}_i \left(\left\| y - a_i (a_i^T y - b_i) - x^* \right\|^2 \right) \\
&= \|y - x^*\|^2 + \mathbb{E}_i \left(\|a_i^T y - b_i\|^2 \right) - 2 \left\langle y - x^*,\ \mathbb{E}_i \left(a_i (a_i^T y - b_i) \right) \right\rangle \\
&= \|y - x^*\|^2 + \frac{1}{m} \|Ay - b\|^2 - \frac{2}{m} \langle A(y - x^*), Ay - b \rangle \\
&= \|y - x^*\|^2 - \frac{1}{m} \|Ay - b\|^2,
\end{aligned}$$

where the last equality uses $Ax^* = b$. □

The proof of Theorem 5.1 below essentially follows the proof for accelerated coordinate descent algorithm in Nesterov (2012) to construct the key inequality (5.28).

Proof. From Algorithm 5 one can verify that if the sequence $\{x_k, y_k, v_k\}$ is generated from $\mathrm{ARK}(A, b, \lambda, x_0, K)$, then the sequence generated from $\mathrm{ARK}(A, b - Ax_0, \lambda, 0, K)$ must be $\{x_k - x_0, y_k - x_0, v_k - x_0\}$. Thus, solving $Ax = b$ is equivalent to solving $Ax = b - Ax_0$ from initial point 0. It therefore suffices to study convergence from the zero initial point.

Recall from (5.1) that γ_k is the larger root of the following convex quadratic function:

$$t(\gamma) := \gamma^2 - \frac{\gamma}{m}(1 - \lambda\gamma_{k-1}^2) - \gamma_{k-1}^2.$$

Since $\lambda \leqslant \lambda_{\min} \leqslant m$, and using $\gamma_{-1} = 0$, we can note the following, from a simple recursive argument:

$$t(0) = -\gamma_{k-1}^2 \leqslant 0, \quad t(1/m) = \gamma_{k-1}^2(\lambda/m^2 - 1) \leqslant 0,$$

and thus $\gamma_k \geqslant 1/m$ for all $k \geqslant 0$. We can also verify that if $\gamma_{k-1} \leqslant 1/\sqrt{\lambda}$, we have

$$\begin{aligned} t(\gamma_{k-1}) &= -(\gamma_{k-1}/m)(1 - \lambda\gamma_{k-1}^2) \leqslant 0 \\ t\left(\frac{1}{\sqrt{\lambda}}\right) &= \frac{1}{\lambda} - \frac{1}{m\sqrt{\lambda}}(1 - \lambda\gamma_{k-1}^2) - \gamma_{k-1}^2 \\ &= \frac{1}{\lambda} - \frac{1}{m\sqrt{\lambda}} + \gamma_{k-1}^2\left(\frac{\sqrt{\lambda}}{m} - 1\right) \\ &\geqslant \frac{1}{\lambda} - \frac{1}{m\sqrt{\lambda}} + \frac{1}{\lambda}\left(\frac{\sqrt{\lambda}}{m} - 1\right) = 0, \end{aligned}$$

which together imply that

$$\gamma_k \in \left[\gamma_{k-1}, \frac{1}{\sqrt{\lambda}}\right].$$

It follows from these bound (together with the initialization $\gamma_{-1} = 0$) that $\{\gamma_k\}_{k=0}^{\infty}$ is an increasing sequence, bounded below by $1/m$ and above by $1/\sqrt{\lambda}$. It follows from these bounds and from $\lambda \leqslant m$ that α_k and β_k both lie in the interval $[0, 1]$ for all k.

Recalling that $x_0 = 0$, we have $x^* = A^+ b$. It can be verified that x_k, y_k, v_k, and x^* are all in $\mathcal{R}(A^T)$. We observe some useful relationships among the scalars in the algorithm. We have from (5.1) and (5.2) that

$$\frac{1 - \alpha_k}{\alpha_k} = \frac{m^2\gamma_k - m}{m - \gamma_k\lambda} = \frac{m}{\gamma_k}\frac{m\gamma_k^2 - \gamma_k}{m - \gamma_k\lambda} = \frac{m\gamma_{k-1}^2}{\gamma_k}. \tag{5.19}$$

From (5.1) and (5.3), we have

$$\gamma_k^2 - \frac{\gamma_k}{m} - \beta_k\gamma_{k-1}^2 = 0. \tag{5.20}$$

Defining

$$r_k := \|v_k - x^*\|_{(A^TA)^+}, \tag{5.21}$$

we consider the following expansion of r_{k+1}^2.

$$\begin{aligned}
r_{k+1}^2 &= \|v_{k+1} - x^*\|_{(A^TA)^+}^2 \\
&= \left\|\beta_k v_k + (1-\beta_k)y_k - \gamma_k a_i(a_i^T y_k - b_i) - x^*\right\|_{(A^TA)^+}^2 \\
&= \|\beta_k v_k + (1-\beta_k)y_k - x^*\|_{(A^TA)^+}^2 + \gamma_k^2 \left\|a_i(a_i^T y_k - b_i)\right\|_{(A^TA)^+}^2 \\
&\quad - 2\gamma_k \left\langle \beta_k v_k + (1-\beta_k)y_k - x^*,\ (A^TA)^+ a_i(a_i^T y_k - b_i)\right\rangle \\
&= \|\beta_k v_k + (1-\beta_k)y_k - x^*\|_{(A^TA)^+}^2 + \gamma_k^2 \left\|a_i(a_i^T y_k - b_i)\right\|_{(A^TA)^+}^2 \\
&\quad - 2\gamma_k \left\langle \beta_k \left(\frac{1}{\alpha_k} y_k - \frac{1-\alpha_k}{\alpha_k} x_k\right) + (1-\beta_k)y_k - x^*,\ (A^TA)^+ a_i(a_i^T y_k - b_i)\right\rangle \\
&= \|\beta_k v_k + (1-\beta_k)y_k - x^*\|_{(A^TA)^+}^2 + \gamma_k^2 \left\|a_i(a_i^T y_k - b_i)\right\|_{(A^TA)^+}^2 \\
&\quad + 2\gamma_k \left\langle x^* - y_k + \frac{1-\alpha_k}{\alpha_k}\beta_k(x_k - y_k),\ (A^TA)^+ a_i(a_i^T y_k - b_i)\right\rangle.
\end{aligned} \tag{5.22}$$

Denote by $i(k)$ the index randomly generated at iteration k, and let $I(k)$ denote all random indices seen at or before iteration k, that is,

$$I(k) := \{i(k), i(k-1), \ldots, i(0)\}.$$

Note that x_{k+1}, y_{k+1}, and v_{k+1} are determined by $I(k)$. In the remainder of the proof, we use $\mathbb{E}_{i(k)|I(k-1)}(\cdot)$ to denote the expectation of a random variable with respect to the index $i(k)$, conditioned on $I(k-1)$. Note that $\mathbb{E}_{I(k)}(\cdot) = \mathbb{E}_{I(k-1)}(\mathbb{E}_{i(k)|I(k-1)}(\cdot))$. When the context is clear, we use i in place of $i(k)$.

We consider the three terms in (5.22) in turn. From the convexity of $\|.\|_{(A^TA)^+}^2$ and the definition of β_k, the first item can be bounded as follows:

$$\begin{aligned}
&\|\beta_k v_k + (1-\beta_k)y_k - x^*\|_{(A^TA)^+}^2 \\
&\leqslant \beta_k \|v_k - x^*\|_{(A^TA)^+}^2 + (1-\beta_k)\|y_k - x^*\|_{(A^TA)^+}^2 \\
&= \beta_k \|v_k - x^*\|_{(A^TA)^+}^2 + \frac{\gamma_k \lambda}{m}\|y_k - x^*\|_{(A^TA)^+}^2 \\
&\leqslant \beta_k \|v_k - x^*\|_{(A^TA)^+}^2 + \frac{\gamma_k}{m}\|y_k - x^*\|^2,
\end{aligned} \tag{5.23}$$

where the last inequality is a consequence of $\lambda \leqslant \lambda_{\min}$ and the fact that y_k and x^* are in $\mathcal{R}(A^T)$. Using Lemmas 5.2 and 5.3, the second item in (5.22) can be bounded in the expectation sense as

follows:

$$\mathbb{E}_{i(k)|I(k-1)}\left(\left\|a_i(a_i^T y_k - b_i)\right\|^2_{(A^TA)^+}\right)$$
$$\leqslant \frac{1}{m}\|Ay_k - b\|^2 = \|y_k - x^*\|^2 - \mathbb{E}_{i(k)|I(k-1)}(\|x_{k+1} - x^*\|^2). \tag{5.24}$$

For the third term in (5.22), we have by taking an expectation that

$$\begin{aligned}
&\mathbb{E}_{i(k)|I(k-1)}\left\langle x^* - y_k + \frac{1-\alpha_k}{\alpha_k}\beta_k(x_k - y_k),\ (A^TA)^+\left(a_i(a_i^T y_k - b_i)\right)\right\rangle \\
&= \left\langle x^* - y_k + \frac{1-\alpha_k}{\alpha_k}\beta_k(x_k - y_k),\ (A^TA)^+\mathbb{E}_{i(k)|I(k-1)}\left(a_i(a_i^T y_k - b_i)\right)\right\rangle \\
&= \frac{1}{m}\left\langle x^* - y_k + \frac{1-\alpha_k}{\alpha_k}\beta_k(x_k - y_k),\ (A^TA)^+\sum_i a_i(a_i^T y_k - b_i)\right\rangle \\
&= \frac{1}{m}\left\langle x^* - y_k + \frac{1-\alpha_k}{\alpha_k}\beta_k(x_k - y_k),\ (A^TA)^+A^TA(y_k - x^*)\right\rangle \\
&\leqslant \frac{1}{m}\left\langle x^* - y_k + \frac{1-\alpha_k}{\alpha_k}\beta_k(x_k - y_k),\ y_k - x^*\right\rangle \\
&= \frac{1}{m}\left(-\|y_k - x^*\|^2 + \frac{1-\alpha_k}{\alpha_k}\beta_k\left\langle x_k - y_k,\ y_k - x^*\right\rangle\right) \\
&= \frac{1}{m}\left(-\|y_k - x^*\|^2 + \frac{1-\alpha_k}{2\alpha_k}\beta_k\left(\|x_k - x^*\|^2 - \|y_k - x^*\|^2 - \|x_k - y_k\|^2\right)\right) \\
&= \frac{1}{m}\left(-\left(1 + \frac{1-\alpha_k}{2\alpha_k}\beta_k\right)\|y_k - x^*\|^2 + \frac{1-\alpha_k}{2\alpha_k}\beta_k\left(\|x_k - x^*\|^2 - \|x_k - y_k\|^2\right)\right) \\
&= -\left(\frac{1}{m} + \frac{\beta_k\gamma_{k-1}^2}{2\gamma_k}\right)\|y_k - x^*\|^2 + \frac{\beta_k\gamma_{k-1}^2}{2\gamma_k}\left(\|x_k - x^*\|^2 - \|x_k - y_k\|^2\right) \quad \text{(from (5.19))} \\
&\leqslant -\left(\frac{1}{m} + \frac{\beta_k\gamma_{k-1}^2}{2\gamma_k}\right)\|y_k - x^*\|^2 + \frac{\beta_k\gamma_{k-1}^2}{2\gamma_k}\|x_k - x^*\|^2.
\end{aligned} \tag{5.25}$$

By substituting (5.23), (5.24), and (5.25) into (5.22), we obtain

$$\begin{aligned}
&\mathbb{E}_{i(k)|I(k-1)}(r_{k+1}^2) \\
&\leqslant \beta_k\|\nu_k - x^*\|^2_{(A^\top A)^+} + \frac{\gamma_k}{m}\|y_k - x^*\|^2 \\
&\quad + \gamma_k^2(\|y_k - x^*\|^2 - \mathbb{E}_{i(k)|I(k-1)}(\|x_{k+1} - x^*\|^2)) \\
&\quad - \left(\frac{2\gamma_k}{m} + \beta_k\gamma_{k-1}^2\right)\|y_k - x^*\|^2 + \beta_k\gamma_{k-1}^2\|x_k - x^*\|^2 \\
&\leqslant \beta_k\|\nu_k - x^*\|^2_{(A^\top A)^+} - \gamma_k^2\mathbb{E}_{i(k)|I(k-1)}(\|x_{k+1} - x^*\|^2) + \beta_k\gamma_{k-1}^2\|x_k - x^*\|^2 \\
&\quad + \left(\gamma_k^2 - \frac{\gamma_k}{m} - \beta_k\gamma_{k-1}^2\right)\|y_k - x^*\|^2 \\
&= \beta_k\|\nu_k - x^*\|^2_{(A^\top A)^+} - \gamma_k^2\mathbb{E}_{i(k)|I(k-1)}(\|x_{k+1} - x^*\|^2) + \beta_k\gamma_{k-1}^2\|x_k - x^*\|^2, \qquad (5.26)
\end{aligned}$$

where the final equality is a consequence of (5.20).

We now define two scalar sequences $\{A_k\}$ and $\{B_k\}$ as follows:

$$A_k \geqslant 0,\ B_k \geqslant 0,\ B_0 \neq 0,\ B_{k+1}^2 = \frac{B_k^2}{\beta_k},\ A_{k+1}^2 = \gamma_k^2 B_{k+1}^2. \qquad (5.27)$$

We set $A_0 = 0$ (to be consistent with the definition (5.27) and the fact that $\gamma_{-1} = 0$ in Algorithm 5) and note that $B_{k+1} \geqslant B_k$, since $\beta_k \in (0, 1]$. Since from (5.27) together with (5.1) and (5.3), we have

$$A_{k+1}^2 = \frac{B_k^2\gamma_k^2}{\beta_k} = \frac{\gamma_k^2 A_k^2}{\beta_k\gamma_{k-1}^2} = \frac{A_k^2\gamma_k^2}{\gamma_k^2 - \gamma_k/m},$$

we obtain that $\{A_k\}$ is also an increasing sequence.

Multiplying the last inequality (5.26) by B_{k+1}^2, and using the definition of r_k (5.21) along with (5.27) (in particular, the identities $B_{k+1}^2\gamma_k^2 = A_{k+1}^2$, $B_{k+1}^2\beta_k = B_k^2$, and $B_{k+1}^2\beta_k\gamma_{k-1}^2 = A_k^2$), we obtain

$$\begin{aligned}
B_{k+1}^2\mathbb{E}_{i(k)|I(k-1)}(r_{k+1}^2) + A_{k+1}^2\mathbb{E}_{i(k)|I(k-1)}(\|x_{k+1} - x^*\|^2) \\
\leqslant B_k^2 r_k^2 + A_k^2\|x_k - x^*\|^2. \qquad (5.28)
\end{aligned}$$

It follows that

$$\begin{aligned}
&\mathbb{E}_{I(k)}(B_{k+1}^2 r_{k+1}^2 + A_{k+1}^2(\|x_{k+1} - x^*\|^2)) \\
&\qquad = \mathbb{E}_{I(k-1)}(B_{k+1}^2\mathbb{E}_{i(k)|I(k-1)}(r_{k+1}^2) + A_{k+1}^2\mathbb{E}_{i(k)|I(k-1)}(\|x_{k+1} - x^*\|^2)) \\
&\qquad \leqslant \mathbb{E}_{I(k-1)}(B_k^2 r_k^2 + A_k^2\|x_k - x^*\|^2).
\end{aligned}$$

By applying this inequality recursively, we obtain

$$\begin{aligned}\mathbb{E}_{I(k)}(B_{k+1}^2 r_{k+1}^2 + A_{k+1}^2(\|x_{k+1} - x^*\|^2)) &\leqslant \mathbb{E}_{I(0)}(B_1^2 r_1^2 + A_1^2\|x_1 - x^*\|^2) \\ &\leqslant B_0^2 r_0^2 + A_0^2\|x_0 - x^*\|^2 = B_0^2 r_0^2,\end{aligned}$$

where we dropped the last term because $A_0 = 0$. It follows from this bound that

$$\mathbb{E}(r_{k+1}^2) \leqslant \frac{B_0^2}{B_{k+1}^2} r_0^2 \quad \text{and} \quad \mathbb{E}(\|x_{k+1} - x^*\|^2) \leqslant \frac{B_0^2}{A_{k+1}^2} r_0^2. \tag{5.29}$$

We now need to estimate the growth of two sequences $\{A_k\}$ and $\{B_k\}$. Here we follow the proof for the accelerated coordinate descent algorithm of Nesterov (2012), but spelling out some details skipped in that paper. We have

$$B_k^2 = B_{k+1}^2 \beta_k = \left(1 - \frac{\lambda}{m}\gamma_k\right) B_{k+1}^2 = \left(1 - \frac{\lambda A_{k+1}}{m B_{k+1}}\right) B_{k+1}^2,$$

which implies that

$$\frac{\lambda}{m} A_{k+1} B_{k+1} = B_{k+1}^2 - B_k^2 = (B_k + B_{k+1})(B_{k+1} - B_k),$$

so by recalling that $B_{k+1} \geqslant B_k$, we obtain

$$B_{k+1} \geqslant B_k + \frac{\lambda}{2m} A_k. \tag{5.30}$$

We have

$$\begin{aligned}\frac{A_{k+1}^2}{B_{k+1}^2} - \frac{A_{k+1}}{B_{k+1} m} &= \gamma_k^2 - \frac{\gamma_k}{m} && \text{(from (5.27))} \\ &= (1 - \frac{\gamma_k \lambda}{m})\gamma_{k-1}^2 && \text{(from (5.1))} \\ &= \frac{\beta_k A_k^2}{B_k^2} = \frac{A_k^2}{B_{k+1}^2} && \text{(from (5.3) and (5.27)),}\end{aligned}$$

so we obtain by multiplying both sides of this expression by B_{k+1}^2 and using $A_{k+1} \geqslant A_k$ that

$$\frac{1}{m} A_{k+1} B_{k+1} = A_{k+1}^2 - A_k^2 = (A_{k+1} + A_k)(A_{k+1} - A_k) \leqslant 2A_{k+1}(A_{k+1} - A_k)$$

and therefore

$$A_{k+1} \geqslant A_k + \frac{B_{k+1}}{2m} \geqslant A_k + \frac{B_k}{2m}. \tag{5.31}$$

By combining the inequalities (5.30) and (5.31) and applying a recursive argument, we can estimate

A_{k+1} and B_{k+1} as follows:

$$\begin{bmatrix} A_{k+1} \\ B_{k+1} \end{bmatrix} \geqslant \begin{bmatrix} 1 & \frac{1}{2m} \\ \frac{\lambda}{2m} & 1 \end{bmatrix}^{k+1} \begin{bmatrix} A_0 \\ B_0 \end{bmatrix}.$$

The Jordan decomposition of the matrix in this expression is

$$\begin{bmatrix} 1 & \frac{1}{2m} \\ \frac{\lambda}{2m} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ \sqrt{\lambda} & -\sqrt{\lambda} \end{bmatrix}^{-1} \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ \sqrt{\lambda} & -\sqrt{\lambda} \end{bmatrix},$$

with

$$\sigma_1 = 1 + \frac{\sqrt{\lambda}}{2m}, \quad \sigma_2 = 1 - \frac{\sqrt{\lambda}}{2m}.$$

Thus we have

$$\begin{aligned} \begin{bmatrix} 1 & \frac{1}{2m} \\ \frac{\lambda}{2m} & 1 \end{bmatrix}^{k+1} &= \frac{1}{2} \begin{bmatrix} 1 & \frac{1}{\sqrt{\lambda}} \\ 1 & -\frac{1}{\sqrt{\lambda}} \end{bmatrix} \begin{bmatrix} \sigma_1^{k+1} & 0 \\ 0 & \sigma_2^{k+1} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ \sqrt{\lambda} & -\sqrt{\lambda} \end{bmatrix} \\ &= \frac{1}{2} \begin{bmatrix} \sigma_1^{k+1} + \sigma_2^{k+1} & (\sigma_1^{k+1} - \sigma_2^{k+1})/\sqrt{\lambda} \\ (\sigma_1^{k+1} - \sigma_2^{k+1})\sqrt{\lambda} & \sigma_1^{k+1} + \sigma_2^{k+1} \end{bmatrix} \end{aligned}$$

which implies $A_{k+1} \geqslant B_0(\sigma_1^{k+1} - \sigma_2^{k+1})/(2\sqrt{\lambda})$ and $B_{k+1} \geqslant (\sigma_1^{k+1} + \sigma_2^{k+1})B_0/2$. By combining these bounds with (5.29), we obtain

$$\begin{aligned} \mathbb{E}(r_{k+1}^2) = \mathbb{E}(\|v_{k+1} - x^*\|_{(A^TA)^+}^2) &\leqslant \frac{B_0^2}{B_{k+1}^2} r_0^2 \leqslant \frac{4\|x_0 - x^*\|_{(A^TA)^+}^2}{(\sigma_1^{k+1} + \sigma_2^{k+1})^2}, \\ \mathbb{E}(\|x_{k+1} - x^*\|^2) &\leqslant \frac{B_0^2}{A_{k+1}^2} r_0^2 \leqslant \frac{4\lambda\|x_0 - x^*\|_{(A^TA)^+}^2}{(\sigma_1^{k+1} - \sigma_2^{k+1})^2}, \end{aligned}$$

completing the proof. □

Agarwal, A., and J. C. Duchi. 2012. Distributed delayed stochastic optimization. *CDC* 5451–5452.

Aharoni, R., and Y. Censor. 1989. Block-iterative projection methods for parallel computation of solutions to convex feasibility problems. *Linear Algebra and Its Applications* 120:165–175.

Anitescu, M. 2000. Degenerate nonlinear programming with a quadratic growth condition. *SIAM Journal on Optimization* 10(4):1116–1135.

Avron, H., A. Druinsky, and A. Gupta. 2014. Revisiting asynchronous linear solvers: Provable convergence rate through randomization. *IPDPS*.

Beck, A., and M. Teboulle. 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sciences* 2(1):183–202.

Beck, A., and L. Tetruashvili. 2013. On the convergence of block coordinate descent type methods. To appear in *SIAM Journal on Optimization*.

Bertsekas, D. P., and J. N. Tsitsiklis. 1989. *Parallel and distributed computation: numerical methods*. Pentice Hall.

Boyd, S., N. Parikh, E. Chu, B. Peleato, and J. Eckstein. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning* 3(1):1–122.

Bradley, J. K., A. Kyrola, D. Bickson, and C. Guestrin. 2011. Parallel coordinate descent for L1-regularized loss minimization. *ICML*.

Censor, Y., D. Gordon, and R. Gordon. 2001. Component averaging: An efficient iterative parallel algorithm for large and sparse unstructured problems. *Parallel Computing* 27(6):777–808.

Chang, C.-C., and C.-J. Lin. 2011. LIBSVM: A library for support vector machines.

Cortes, C., and V. Vapnik. 1995. Support vector networks. *Machine Learning* 273–297.

Cotter, A., O. Shamir, N. Srebro, and K. Sridharan. 2011. Better mini-batch algorithms via accelerated gradient methods. *arXiv: 1106.4574*.

Duchi, J. C., A. Agarwal, and M. J. Wainwright. 2012. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Transactions on Automatic Control* 57(3):592–606.

Eggermont, P. P. B. 1981. Iterative algorithms for large partitioned linear systems, with applications to image reconstruction. *Linear Algebra and its Applications* 40:37–67.

Eldar, Y. C., and D. Needell. 2011. Acceleration of randomized Kaczmarz method via the Johnson-Lindenstrauss lemma. *Numerical Algorithms* 58(2):163–177.

Elfving, T., and T. Nikazad. 2009. Properties of a class of block-iterative methods. *Inverse problems* 25.

Elsner, L., I. Koltracht, and M. Neumann. 1990. On the convergence of asynchronous paracontractions with application to tomographic reconstruction from incomplete data. *Linear Algebra and its Applications* 130:65–82.

———. 1992. Convergence of sequential and asynchronous paracontractions nonlinear paracontractuions. *Numerische Mathematik* 62:305–316.

Fercoq, O., and P. Richtárik. 2013a. Accelerated, parallel, and proximal coordinate descent. Technical Report, School of Mathematics, University of Edinburgh. ArXiv: 1312.5799.

———. 2013b. Smooth minimization of nonsmooth functions by parallel coordinate descent. Technical Report, School of Mathematics, University of Edinburgh. ArXiv:1309.5885.

Ferris, M. C., and O. L. Mangasarian. 1994. Parallel variable distribution. *SIAM Journal on Optimization* 4(4):815–832.

Frommer, A., and D. B. Szyld. 2000. On asynchronous iterations. *Journal of Computational and Applied Mathematics* 123:201–216.

Galantai, A. 2005. On the rate of convergence of the alternating projection method in finite dimensional spaces. *Journal of Mathematical Analysis and Applications* 310:30–44.

Goldfarb, D., and S. Ma. 2012. Fast multiple-splitting algorithms for convex optimization. *SIAM Journal on Optimization* 22(2):533–556.

Herman, G. T. 1980. *Image reconstruction from projections: The fundamentals of computerized tomography.* Academic Press.

———. 2009. *Fundamentals of computerized tomography.* Springer.

Hoffman, A. J. 1952. On approximate solutions of systems of linear inequalities. *Journal of Research of the National Bureau of Standards* 49(4):263–265.

Kaczmarz, S. 1937. Angenaherte auflsung von systemen linearer gleichungen. *Bulletin International de l'Acadmie Polonaise des Sciences et des Letters* 35:355–357.

Lai, M., and W. Yin. 2013. Augmented L1 and nuclear-norm models with a globally linearly convergent algorithm. *SIAM Journal on Imaging Sciences* 6(2):1059–1091.

Leventhal, D., and A. S. Lewis. 2010. Randomized methods for linear constraints: Convergence rates and conditioning. *Mathematics of Operations Research* 35(3):641–654.

Liu, J., and S. J. Wright. 2014. Asynchronous stochastic coordinate descent: Parallelism and convergence properties. *ArXiv: 1403.3862*.

Liu, J., S. J. Wright, C. Ré, V. Bittorf, and S. Sridhar. 2013. An asynchronous parallel stochastic coordinate descent algorithm. *arXiv: 1311.1873*.

Liu, J., S. J. Wright, and S. Sridhar. 2014. An asynchronous parallel randomized Kaczmarz algorithm. Technical Report, Computer Sciences Department, University of Wisconsin-Madison. ArXiv: 1401.4780.

Lu, Z., and L. Xiao. 2013. On the complexity analysis of randomized block-coordinate descent methods. *TechReport*.

Luo, Z. Q., and P. Tseng. 1992. On the convergence of the coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications* 72:7–35.

Mangasarian, O. L. 1995. Parallel gradient distribution in unconstrained optimization. *SIAM Journal on Optimization* 33(1):916–1925.

Necoara, I., and D. Clipici. 2013. Efficient parallel coordinate descent algorithm for convex optimization problems with separable constraints: application to distributed MPC. Technical Report, Automation and Systems Engineering Department, University Politechnica Bucharest. ArXiv: 1302.3092.

Necoara, I., and A. Patrascu. 2013. A random coordinate descent algorithm for optimization problems with composite objective function and linear coupled constraints. Technical Report, Automation and Systems Engineering Department, University Politechnica Bucharest. ArXiv: 1302.3074.

Needell, D. 2010. Randomized Kaczmarz solver for noisy linear systems. *BIT Numerical Mathematics* 50(2):1422–1436.

Nemirovski, A., A. Juditsky, G. Lan, and A. Shapiro. 2009. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization* 19:1574–1609.

Nesterov, Y. 2004. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer Academic Publishers.

———. 2012. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization* 22(2):341–362.

Nikazad, T. 2008. Algebraic reconstruction methods. *Thesis*.

Niu, F., B. Recht, C. Ré, and S. J. Wright. 2011. HOGWILD!: A lock-free approach to parallelizing stochastic gradient descent. *Advances in Neural Information Processing Systems 24* 693–701.

Nocedal, J., and S. J. Wright. 2006. Numerical optimization. *Springer Verlag*.

Peng, Z., M. Yan, and W. Yin. 2013. Parallel and distributed sparse optimization. *preprint*.

Popa, C. 1999. Characterization of the solutions set of least-squares problems by an extension of Kaczmarz's projections method. *Journal of Applied Mathematics and Computing* 6:51–64.

Richtárik, P., and M. Takáč. 2011. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *arXiv:1107.2848*.

———. 2012. Parallel coordinate descent methods for big data optimization. *ArXiv: 1212.0873*.

Scherrer, C., A. Tewari, M. Halappanavar, and D. Haglin. 2012. Feature clustering for accelerating parallel coordinate descent. *NIPS* 28–36.

Shalev-Shwartz, S., and T. Zhang. 2013. Accelerated mini-batch stochastic dual coordinate ascent. *Arxiv: 1305.2581*.

Shamir, O., and T. Zhang. 2013. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. *ICML*.

Sridhar, S., V. Bittorf, J. Liu, C. Zhang, C. Ré, and S. J. Wright. 2013. An approximate, efficient solver for LP rounding. *NIPS*.

Strohmer, T., and R. Vershynin. 2009. A randomized kaczmarz algorithm with exponential convergence. *Journal of Fourier Analysis and Applications* 15:262–278.

Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B* 58:267–288.

Tseng, P. 2001. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications* 109:475–494.

Tseng, P., and S. Yun. 2009. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming, Series B* 117:387–423.

———. 2010. A coordinate gradient descent method for linearly constrained smooth optimization and support vector machines training. *Computational Optimization and Applications* 47(2):179–206.

Vershynin, R. 2011a. Introduction to the non-asymptotic analysis of random matrices. *ArXiv: 1011.3027*.

———. 2011b. Introduction to the non-asymptotic analysis of random matrices. Preprint arXiv:1011.3027.

Wang, P.-W., and C.-J. Lin. 2013. Iteration complexity of feasible descent methods for convex optimization. *Technical report*.

Wright, S. J. 2012. Accelerated block-coordinate relaxation for regularized optimization. *SIAM Journal on Optimization* 22(1):159–186.

Yang, T. 2013. Trading computation for communication: Distributed stochastic dual coordinate ascent. *NIPS* 629–637.

Zouzias, A., and N. M. Freris. 2012. Randomized extended Kaczmarz for solving least-squares. *arXiv:1205.5770v2*.