An Asynchronous Parallel Randomized Kaczmarz Algorithm

Ji Liu, Stephen J. Wright, and Srikrishna Sridhar
\{ji-liu, swright, srikris\}@cs.wisc.edu
Department of Computer Sciences, University of Wisconsin-Madison

January 21, 2014

Abstract

We describe an asynchronous parallel variant of the randomized Kaczmarz (RK) algorithm for solving the linear system \(Ax = b\). The analysis shows linear convergence and indicates that nearly linear speedup can be expected if the number of processors is bounded by a multiple of the number of rows in \(A\).

1 Introduction

We consider the problem of finding a solution to a consistent linear system

\[
Ax = b,
\]

where \(A \in \mathbb{R}^{m \times n}\) and \(b \in \mathbb{R}^m\). We denote the rows and columns of \(A\) by \(a^T_i\) 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,

\[
A = \begin{bmatrix}
a^T_1 \\
a^T_2 \\
\vdots \\
a^T_m \\
\end{bmatrix} = [\bar{a}_1, \bar{a}_2, \ldots, \bar{a}_n], \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}.
\]

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 (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 \(\frac{\|a_i\|^2}{\|A\|_F^2}\)) and does an orthogonal projection of the current estimate vector onto the hyperplane \(a^T_{i(j)} x = b_{i(j)}\):

\[
x_{j+1} = x_j - \frac{a_{i(j)}^T x_j - b_{i(j)}}{\|a_{i(j)}\|^2} a_{i(j)},
\]
This update formula can be derived also by applying the basic stochastic gradient algorithm to the objective \( \frac{1}{2} \| Ax - b \|^2 \) = \( \frac{1}{2} \sum_i \langle a_i^T x - b_i \rangle \) where \( \langle a_i^T x - b_i \rangle a_i \) is the stochastic gradient corresponding to a random choice of index \( i \) and \( 1/\|a_i\|^2 \) is the step length 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]:

\[
\begin{align*}
\|x_{j+1} - x^*_j\|^2 &\leq \|x_j - \frac{1}{\|a_{i(j)}\|^2} a_{i(j)}(a_{i(j)}^T x_j - b_{i(j)}) - x_j\|^2 \\
&= \|x_j - x^*_j\|^2 - \frac{1}{\|a_{i(j)}\|^2} (a_{i(j)}^T x_j - b_{i(j)})^2,
\end{align*}
\]

where \( x^*_j \) is the projection \( x_j \) onto the solution set. Given the probability \( \|a_i\|^2/\|A\|_F^2 \) of choosing \( i \), we have by taking expectations that

\[
\begin{align*}
E_{i(j)} \left[ \|x_{j+1} - x^*_j\|^2 \mid x_j \right] &\leq \|x_j - x^*_j\|^2 - E_{i(j)} \left[ \frac{1}{\|a_{i(j)}\|^2} (a_{i(j)}^T x_j - b_{i(j)})^2 \right] \\
&\leq \left( 1 - \frac{\lambda_{\min}}{\|A\|_F^2} \right) \|x_j - x^*_j\|^2,
\end{align*}
\]

where \( \lambda_{\min} \) is the smallest nonzero eigenvalue value of \( A^T A \).

Recently, asynchronous parallel stochastic algorithms have received broad attention for solving large convex optimization problems. [Niu et al., 2011] proposed a simple but efficient asynchronous scheme to parallelize the stochastic gradient algorithm. In this approach, the unknown vector \( x \) is stored in memory locations accessible to all cores, and all cores are free to update \( x \) in an asynchronous, uncoordinated fashion. We assume that there is a bound \( \tau \) on the age of the updates, that is, no more that \( \tau \) updates in total can be occur between the time at which any processor reads the current \( x \) and the time at which it makes its update. HOGWILD! [Niu et al., 2011] allows a lock-free implementation, since the update to a single element of \( x \) is an atomic operation. [Avron et al., 2013] and [Liu et al., 2013] applied a similar asynchronous scheme to stochastic coordinate descent, and have proved attractive convergence properties.

We apply the same asynchronous parallel technique used in HOGWILD! [Niu et al., 2011] 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!.

We also define a sufficient condition on the maximum number of cores for which near-linear speedup can be observed. This condition relates the number of threads / cores to the number of equations \( m \), and to the maximal eigenvalue of \( A^T A \).

We review related work in Section 2. Section 3 illustrates details of the ASYRK algorithm. The convergence rate of ASYRK is described in Section 4 with proofs given in Appendix A. Some simple experiments illustrate linear speedup in Section 5. We discuss extensions to the inconsistent case in Section 7 and make some concluding observations in Section 8.
Notation and Assumption

We use the following notation.

- $\|x\|_0$ denotes the cardinality or “$\ell_0$ 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 $\alpha \leq \sqrt{\nu \mu}$ and $\alpha \leq \|A\| \mu$.
- Given $x_j \in \mathbb{R}^n$, $x_j^*$ denotes the projection of $x_j$ onto the solution set of (1).
- The support index set of $x$ is defined as $\text{supp}(x)$.
- $\lambda_{\text{min}}$ is defined as the minimal nonzero eigenvalue value of $A^T A$, while $\lambda_{\text{max}}$ is defined as the maximal eigenvalue value of $A^T A$.

We make a few observations about $\lambda_{\text{max}}$. 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, 2011] yield that $\lambda_{\text{max}}$ is bounded by $(\sqrt{m}+\sqrt{n})^2 \leq O(1+m/n)$ with high probability. As long as $m/n$ is bounded by a constant, $\lambda_{\text{max}}$ is bounded by a constant as well. If $A$ is a sparse matrix, then

$$\lambda_{\text{max}} := \max_{\|y\|=1} \|A^T A\| = \max_{\|y\|=1} \|A y\|^2 \leq \max \{ j : \text{supp}(a_i) \cap \text{supp}(a_j) \neq \emptyset \} \leq \mu \nu.$$  

**Assumption 1.** Assume that

- The solution to (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.

## 2 Related Work

The algorithm proposed by [Kaczmarz, 1937] used a cyclic projection procedure to solve consistent linear systems $Ax = b$. He proved convergence to the unique solution if $A$ 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.

[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 the 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. 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 at each iteration selects the optimal projection from a randomly chosen set. This technique improves the convergence rate but requires more computation cost in each iteration. Liu and Wright [2013] proposed an accelerated RK algorithm that uses a Nesterov-type accelerated scheme, improving the linear convergence rate constant from $1 - \lambda_{\text{min}}/m$ (corresponding to (2), after normalization of rows) to $1 - \sqrt{\lambda_{\text{min}}}/m$.

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: $A_I x \geq b_I$, $A_E x = b_E$.

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. For other row-action methods and variants of RK, we refer to Censor [1981].

Zouzias and Freris [2012] considered the case of possibly inconsistent (1). 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 $Ax = b_\perp$. Essentialy, the RK algorithm is applied twice. The convergence rate is proven to be $1 - \lambda_{\text{min}}/\|A\|_F^2$, 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].

Among synchronous parallel methods, Censor et al. [2001] proposed a parallel component averaging method to solve (1). This approach parallel-projects the current $x$ onto all 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.

Another synchronous parallel approach (for general convex optimization) due to Ferris and Mangasarian [1994] distributes variables among multiple processors and optimizes concurrently over each subset. A synchronization step searches the affine hull formed by the current iterate and the partial optima found by each processor.

In discussing asynchronous parallel methods, we make a distinction according to whether it is assumed that the reading of $x$ by each processor is “consistent” or not. The term “consistent” in this context means that the $x$ used by each processor to evaluate its update is an iterate that actually existed at some point in time, whose components were not changed repeatedly by other processors during reading (therefore making a hybrid of two or more iterates). 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 of minimizing $f$ over a closed convex set $\Omega$ can be formulated as a fixed-point problem by defining $q(x) := P_\Omega[(I-\alpha \nabla f)(x)]$, where $P_\Omega$ denotes Euclidean projection onto $\Omega$. All processors update $x$ stored in commonly accessible memory, and update the value of $x$ without locking or coordination. Inconsistent reading of $x$ is allowed. Linear convergence is established (using admirably straightforward analysis) provided that $\nabla^2 f(x)$ satisfies a diagonal dominance condition, guaranteeing that the iteration $x = q(x)$ is a maximum norm contraction mapping for sufficient small $\gamma$. However, this condition is even stronger than the strong convexity condition.

HOGWILD! [Niu et al. [2011]] is a lock-free, asynchronous parallel version of the stochastic gradient method. All processors share the same memory storing $x$ and update it simultaneously. Unlike Bertsekas and Tsitsiklis [1989], inconsistent reads of $x$ are not permitted by the analysis. When the updates satisfy a certain sparsity property, the convergence of HOGWILD! approximately matches
the 1/t rate of serial stochastic gradient, as described and analyzed by Nemirovski et al. [2009]. Recent work by Avron et al. [2013] concerned an asynchronous linear solver for \(Ax = b\) (for symmetric positive definite \(A\)) using the same asynchronous scheme as HOGWILD!, proving a linear convergence rate.

Liu et al. [2013] followed the model of HOGWILD! to propose an asynchronous parallel stochastic coordinate descent (AsySCD) algorithm and proved sublinear (1/t) convergence on general convex functions and a linear convergence rate on functions that satisfy an essential strong convexity property. Richtárík and Takáč [2012] proposed a parallel coordinate descent method for minimization of a composite convex objective with separable nonsmooth part. Their method is a synchronous parallel approach, in contrast to AsySCD, which is asynchronous. Another distinction between the two approaches is found in the convexity assumptions, which are slightly weaker in Liu et al. [2013].

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 1 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 \(k(j) \leq j\), 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,

\[
k(j) \geq j - \tau,
\]

for some integer \(\tau \geq 1\). \(\tau\) can be assumed to be similar to the number of processors that are involved in the computation. Note that the step depends on \(\theta_{i(j)}\) (the cardinality of the chosen row) and a parameter \(\gamma\) which is critical to the analysis of the following sections.
Algorithm 1 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 $j \leq K$ 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 $x_{j+1} \leftarrow x_j - \gamma \theta_{i(j)} P_{i(j)} a_{i(j)} (a_{i(j)}^T x_k(j) - b_{i(j)})$;
7: $j \leftarrow j + 1$;
8: end while

4 Main Results

This section presents the convergence analysis for AsyRK. The key issue for AsyRK is to choose an appropriate steplength parameter $\gamma$. At an intuitive level, we would like $\gamma$ 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 $x_{k(j)}$ 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:

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

where $\rho$ is a user defined parameter, usually set to be slightly larger than 1. The steplength $\gamma$ depends strongly on $\rho$.

We state a result about convergence of AsyRK in Algorithm 1.

**Theorem 1.** Assume that Assumption 1 is satisfied. Let $\rho$ be any number greater than 1 and define the quantity $\psi$ as follows:

$$\psi := \mu + \frac{2\lambda_{\max}\tau \rho^\tau}{m}. \quad (4)$$

Suppose the steplength parameter $\gamma > 0$ in Algorithm 1 satisfies the following three bounds:

$$\gamma \leq \frac{1}{\psi}, \quad \gamma \leq \frac{m(\rho - 1)}{2\lambda_{\max}\rho^{\tau + 1}}, \quad \gamma \leq m\sqrt{\frac{(\rho - 1)}{\rho^\tau (m\alpha^2 + \lambda_{\max}^2 \tau \rho^\tau)}}. \quad (5)$$

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

$$\rho^{-1}E(\|Ax_j - b\|^2) \leq E(\|Ax_{j+1} - b\|^2) \leq \rho E(\|Ax_j - b\|^2) \quad (6)$$

and

$$E(\|x_j - x^*_j\|^2) \leq \left(1 - \frac{\lambda_{\min}\gamma}{m} (2 - \gamma \psi)\right)^j \|x_0 - x^*_0\|^2. \quad (7)$$

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 $\gamma$ decreases as the bound $\tau$ 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 $\tau$ in terms of $m$ and $\lambda_{\max}$.

**Corollary 2.** Suppose that Assumption 1 is satisfied and that

$$\frac{2e\lambda_{\max}(\tau + 1)}{m} \leq 1. \quad (8)$$

Then if we choose

$$\rho = 1 + \frac{2e\lambda_{\max}(\tau + 1)}{m} \quad (9)$$

and set $\gamma = 1/\psi$, where $\psi$ is defined as in (4), we have that

$$\mathbb{E}(\|x_j - x_j^*\|^2) \leq \left(1 - \frac{\lambda_{\min}}{m(\mu + 1)}\right)^j\|x_0 - x_0^*\|^2. \quad (10)$$

Over a span of $m$ iterations, (10) implies a decrease factor of approximately $1 - \lambda_{\min}/(\mu + 1)$. This rate estimate indicates that for a delay $\tau$ (and hence a number of processors) in the range implied by (5), 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 $\{\|x_j - x_j^*\|^2\}_{j=1,2,\ldots}$.

**Theorem 3.** Suppose that the assumptions of Corollary 2 hold, and that $\rho$ and $\psi$ are defined as there. For $\epsilon > 0$ and $\eta \in (0,1)$, if

$$j \geq \frac{m(\mu + 1)}{\lambda_{\min}} \log \frac{\|x_0 - x_0^*\|^2}{\eta \epsilon} \quad (11)$$

we have that

$$\mathbb{P}(\|x_j - x_j^*\|^2 \leq \epsilon) \geq 1 - \eta. \quad (12)$$

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

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 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. 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.
Table 1: Comparison among RK, AsySCD, and AsyRK. The quantity $\delta$ is the fraction of nonzero entries in $A$, while $L_{\text{res}}$ is the maximal row norm of $A^T A$ and $L_{\text{max}}$ is the maximal diagonal entry of $A^T A$. (We assume that the nonzeros are roughly evenly distributed in $A$, so that $\mu$ is a modest multiple of $\delta n$.) 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 |
|------------|----|--------|-------|
| # operation per iteration rate (iteration) | $O(\delta n)$ | $\min\{O(\delta^2 mn), O(n)\}$ | $O(\delta n)$ |
| # processors | $1 - \frac{\lambda_{\min}}{m}$ | $1 - \frac{\lambda_{\min}}{2n L_{\text{max}}}$ | $1 - \frac{\lambda_{\min}}{m(\mu+1)}$ |
| rate (running time) | $1 - O\left(\frac{\lambda_{\min}}{\delta mn}\right)$ | $1 - O\left(\frac{\lambda_{\min}}{n^{1.5} L_{\text{res}} \min(\delta^2 mn, 1)}\right)$ | $1 - O\left(\frac{\lambda_{\min}}{\delta^2 n^2 \lambda_{\text{max}}}\right)$ |

In reporting statistics for AsySCD, we consider two alternative implementations: (1) randomly choose a coordinate $i$ and compute it by $(a_i^T A)x$, 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 $Q_i x$, 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 $\lambda_{\text{max}}$ 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 1 to updated all components in 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 1, we see that AsyRK improves the rate of RK if $\delta mn \gg \delta^2 n \lambda_{\text{max}}$, or equivalently $m \gg \delta n \lambda_{\text{max}}$. Assuming the Gaussian ensemble for $A$ and that $m$ and $n$ are comparable (so that $\lambda_{\text{max}} = O(1)$), 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 $\lambda_{\text{max}} = O(L_{\text{res}})$ under the same scenario for $m$, $n$, and $A$. Comparing the rates (running time) in the last row of Table 1 we find that when the mild condition $\delta < O(n^{-1/4})$ holds, AsyRK converges much faster than AsySCD. Overall, AsyRK has a clear advantage in complexity when applied to sparse problems.

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 $\delta$, we select $\delta mn$ 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 $a_{ij}$ are updated, not just the $t(j)$ component. This scheme makes a single thread behave like $|a_{ij}|$ 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 1 and 2 we choose $m = 80000$ and $n = 100000$, with $\delta = 0.001$, and set the steplength $\gamma$ as 1 in Figure 1 and $\delta = 0.003$ in Figure 2. The left-hand graph in 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 2 which indicates that provided it is below a certain threshold, the value of $\tau$ 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 1), while for $\delta = 0.003$ there is a dropoff for larger numbers of cores (Figure 2). This can perhaps be explained by the difference between our implementation from the analyzed version, in that the nonzeros in the full row $a_{ij}$ 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 $\tau$. In this case, a matrix that is three times more dense could be modeled by a value of $\tau$ that is
three times larger. The effect may be to raise $\tau$ above the threshold for which linear speedup can be expected, thus explaining the (graceful) degradation in speedup for larger numbers of cores in Figure 2.

Next, we compare AsyRK to AsySCD [Liu et al., 2013] on sparse synthetic data sets, on 10 cores (single socket) of the Intel Xeon. Various values of $m$, $n$, and $\delta$ are chosen for comparison in Table 2. A similar number of epochs is required by both algorithms, reflecting the similarity of their theoretical convergence rates; see Section 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 1, the per-iteration complexity of AsySCD is much higher than AsyRK, for these values of the parameters.

7 Extension to Inconsistent Systems

Although this paper 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 =
which can be stated as the following square, consistent system of linear equations:

\[ Ax - \zeta y = 0, \quad \phi A^T y = \phi A^T b, \quad (13) \]

for any positive values of \( \zeta \) and \( \phi \). We can choose \( \zeta \) and \( \phi \) to maximize the critical quantity in the analysis of Algorithm \( \Pi \) which is the ratio of the minimum nonzero eigenvalue of \( A^T A \) to its squared Frobenius norm. (In Theorem \( \Pi \), this ratio appears as \( \lambda_{\text{min}}/m \), because of the normalization of the rows of \( A \).) To show how this quantity depends on \( \zeta \) and \( \phi \), we note first that the coefficient matrix in \( (13) \) is

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

Denoting the nonzero singular values of \( A \) by \( \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0 \) 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:

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

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

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

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

\[ \min \left\{ \zeta, \quad -\frac{\zeta}{2} + \frac{1}{2} \sqrt{\zeta^2 + 4 \phi \sigma_r^2} \right\}^2 \frac{\zeta^2}{(1 + \phi^2)\| A \|_F^2 + m \zeta^2} = \min \left\{ \zeta, \quad -\frac{\zeta}{2} + \frac{1}{2} \sqrt{\zeta^2 + 4 \phi \sigma_r^2} \right\} \frac{\zeta^2}{(1 + \phi^2)\| A \|_F^2 + m \zeta^2} \right\}. \quad (14) \]

For fixed \( \phi \), the first term is monotonically increasing with respect to \( \zeta \) while the second term is monotonically decreasing with respect to \( \zeta \). We can thus express the optimal \( \zeta \) value as \( \zeta^* = \sigma_r \sqrt{\phi/2} \), which is the value for which these two terms are equal. By substituting this value into \( (14) \), we obtain

\[ \frac{\sigma_r^2 \phi}{2(1 + \phi^2)\| A \|_F^2 + m \sigma_r^2 \phi} = 2(1/\phi + \phi)\| A \|_F^2 + m \sigma_r^2 \phi. \quad (15) \]
It is clear from the last expression that the optimal value for $\phi$ is $\phi^* = 1$, giving the following maximal value for (15):

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

We conclude that by normalizing the rows of $A$, estimating its minimum singular value $\sigma_r$, and setting $\phi = 1$ and $\zeta = \sigma_r/\sqrt{2}$, we obtain an optimally conditioned system (13), to which the approach of this section can be applied.

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.

Acknowledgements

The authors acknowledge support of National Science Foundation Grants DMS-0914524 and DMS-1216318, ONR Award N00014-13-1-0129, AFOSR Award FA9550-13-1-0138, and a Wisconsin Alumni Research Foundation 2011-12 Fall Competition Award. The authors would like to sincerely thank Yijun Huang for her implementation of algorithms AsyRK and AsySCD used in this paper.

A Proofs

This section provides proofs for our main results in Section 4.

In Algorithm 1, 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 $E$, 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 $E_{i(j)}$ 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 $E_{t(j)}$.

Proof of Theorem 1

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

$$\mathbb{E}_{i(j)}(P_{i(j)}a_{i(j)}) = \frac{1}{\theta_{i(j)}}P_{\text{supp}(a_{i(j)})}a_{i(j)} = \frac{1}{\theta_{i(j)}}a_{i(j)}.$$  

(16)

We prove each of the two inequalities in (6) by induction. We start from the right-hand in-
equality. First we consider the expansion of $\|Ax_{j+1} - b\|^2$ for any values of $j$:

$$
\|Ax_{j+1} - b\|^2 = \|Ax_j - b\|^2 + 2\gamma \langle Ax_j - b, A\theta_{\ell(j)} P_{t(j)} a_{i(j)} (a_i^T x_{k(j)} - b_i(j)) \rangle
$$

$$
= \|Ax_j - b\|^2 + 2\gamma \left( \langle Ax_{k(j)} - b, A\theta_{\ell(j)} P_{t(j)} a_{i(j)} (a_i^T x_{k(j)} - b_i(j)) \rangle \right) +
$$

$$
2\gamma \left( \langle A(x_{k(j)} - x_j), A\theta_{\ell(j)} P_{t(j)} a_{i(j)} (a_i^T x_{k(j)} - b_i(j)) \rangle \right). \tag{17}
$$

Next we consider the expectation of three terms $T_1$, $T_2$, and $T_3$ in (17). For $T_1$, we have

$$
E(T_1) = \mathbb{E}(\|A\theta_{\ell(j)} P_{t(j)} a_{i(j)} (a_i^T x_{k(j)} - b_i(j))\|^2)
$$

$$
\leq \alpha^2 \mathbb{E}(\|a_i^T x_{k(j)} - b\|^2)
$$

$$
= \frac{\alpha^2}{m} \mathbb{E}(\|Ax_{k(j)} - b\|^2). \tag{18}
$$

For $T_2$, we have

$$
E(T_2) = \mathbb{E}(\langle Ax_{k(j)} - b, A\theta_{\ell(j)} P_{t(j)} a_{i(j)} (a_i^T x_{k(j)} - b_i(j)) \rangle)
$$

$$
= \mathbb{E}(\langle Ax_{k(j)} - b, A\theta_{\ell(j)} P_{t(j)} a_{i(j)} (a_i^T x_{k(j)} - b_i(j)) \rangle)
$$

$$
= \mathbb{E}(\langle Ax_{k(j)} - b, Aa_{i(j)} (a_i^T x_{k(j)} - b_i(j)) \rangle) \quad \text{(by (16))}
$$

$$
= \mathbb{E}(\langle A^T (Ax_{k(j)} - b), a_{i(j)} (a_i^T x_{k(j)} - b_i(j)) \rangle)
$$

$$
= \frac{1}{m} \mathbb{E}(\|A^T (Ax_{k(j)} - b)\|^2). \tag{19}
$$
For $T_3$, we have

$$E(T_3) = E\left(\langle A(x_k(j) - x_j), A\theta_i(j) P_{t(j)} a_i(j)(a_i^T x_k(j) - b_i(j))\rangle\right)$$

$$= \gamma E\left(\sum_{d=k(j)}^{j-1} \theta_{i(d)} P_{t(d)} a_i(d)(a_i^T x_k(d) - b_i(d)), A\theta_{i(j)} P_{t(j)} a_i(j)(a_i^T x_k(j) - b_i(j))\right)$$

$$= \gamma \sum_{d=k(j)}^{j-1} E\left(\langle A\theta_{i(d)} \theta_{t(d)} P_{t(d)} a_i(d)(a_i^T x_k(d) - b_i(d)), A\theta_{i(j)} \theta_{t(j)} P_{t(j)} a_i(j)(a_i^T x_k(j) - b_i(j))\rangle\right)$$

$$= \gamma \sum_{d=k(j)}^{j-1} E\left(\langle A a_i(d)(a_i^T x_k(d) - b_i(d)), A a_i(j)(a_i^T x_k(j) - b_i(j))\rangle\right)$$

$$= \gamma \sum_{d=k(j)}^{j-1} E\left(\langle A e_{i(d)}(a_i^T x_k(d) - b_i(d)), A e_{i(j)}(a_i^T x_k(j) - b_i(j))\rangle\right)$$

$$= \gamma \sum_{d=k(j)}^{j-1} E\left(\langle A A^T (x_k(d) - b), A A^T (x_k(j) - b)\rangle\right)$$

$$\leq \frac{\gamma}{2m^2} \sum_{d=k(j)}^{j-1} E\left(\|A A^T (x_k(d) - b)\|^2 + \|A A^T (x_k(j) - b)\|^2\right)$$

$$\leq \frac{\gamma^2 \lambda_{max}^2}{2m^2} \sum_{d=k(j)}^{j-1} \left[ E\left(\|x_k(d) - b\|^2 + \|x_k(j) - b\|^2\right)\right], \quad (20)$$

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 (18), (19), (20), and (17), we obtain

$$E(\|A x_{j+1} - b\|^2) \leq E(\|A x_j - b\|^2) + \frac{\alpha^2 \gamma^2}{m} E(\|A x_k(j) - b\|^2) - \frac{2\gamma}{m} E(\|A A^T (x_k(j) - b)\|^2) +$$

$$\frac{\gamma^2 \lambda_{max}^2}{m^2} \sum_{d=k(j)}^{j-1} \left[ E(\|x_k(d) - b\|^2 + \|x_k(j) - b\|^2)\right]. \quad (21)$$

We can use this bound to show that the right-hand inequality in (6) holds for $j = 0$. By setting $j = 0$ in (21) and noting that $k(0) = 0$ and that the last summation is vacuous, we obtain

$$E(\|A x_1 - b\|^2) \leq E(\|A x_0 - b\|^2) + \frac{\alpha^2 \gamma^2}{m} E(\|A x_0 - b\|^2) - \frac{2\gamma}{m} E(\|A A^T (A x_0 - b)\|^2)$$

$$\leq \|A x_0 - b\|^2 + \frac{\gamma^2 \alpha^2}{m} \|A x_0 - b\|^2$$

$$\leq \left(1 + \frac{\gamma^2 \alpha^2}{m}\right) \|A x_0 - b\|^2. \quad (22)$$

From the third bound in (5), we have

$$1 + \frac{\gamma^2 \alpha^2}{m} \leq 1 + \frac{m(\rho - 1)\alpha^2}{\rho^2(m \alpha^2 + \lambda_{max}^2 \rho^2)} \leq 1 + \frac{m(\rho - 1)\alpha^2}{\rho^2 m \alpha^2} \leq 1 + (\rho - 1) = \rho,$$
where the third inequality follows from $\rho > 1$. By substituting into (22), we obtain $\mathbb{E}(\|x_0 - x_0^*\|^2) \leq \rho \mathbb{E}(\|x_1 - x_1^*\|^2)$.

For the inductive step, we use (21) again, assuming that the right-hand inequality in (6) holds up to stage $j$, and thus that

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

provided that $0 \leq j - k(j) \leq \tau$ and $0 \leq j - k(d) \leq 2\tau$, as assumed. By substituting into the right-hand side of (21) again, we obtain

$$
\mathbb{E}(\|Ax_{j+1} - b\|^2) \leq \mathbb{E}(\|Ax_j - b\|^2) + \frac{\alpha^2 \gamma^2}{m} \mathbb{E}(\|Ax_{k(j)} - b\|^2) + \frac{\gamma^2 \lambda_{\max}^2}{m^2} \sum_{d=k(j)}^{j-1} \mathbb{E}(\|Ax_{k(d)} - b\|^2 + \|Ax_{k(j)} - b\|^2)
$$

$$
\leq \mathbb{E}(\|Ax_j - b\|^2) + \frac{\rho^j \alpha^2 \gamma^2}{m} \mathbb{E}(\|Ax_j - b\|^2) + \frac{\gamma^2 \lambda_{\max}^2}{m^2} (2\tau \rho^{2\tau}) \mathbb{E}(\|Ax_j - b\|^2)
$$

$$
= \left(1 + \gamma^2 \left(\frac{\rho^j (m\alpha^2 + 2\tau \lambda_{\max}^2 \rho^{2\tau})}{m^2}\right)\right) \mathbb{E}(\|Ax_j - b\|^2)
$$

$$
\leq (1 + (\rho - 1)) \mathbb{E}(\|Ax_j - b\|^2)
$$

$$
= \rho \mathbb{E}(\|Ax_j - b\|^2),
$$

where the last inequality uses the third bound on $\gamma$ from (5). We conclude that the right-hand side inequality in (6) holds for all $j$.

We now work on the left-hand inequality in (6). For all $j$, we have the following:

$$
\mathbb{E}(\|Ax_{j+1} - b\|^2) = \mathbb{E}(\|Ax_j - b\|^2) + \gamma^2 \mathbb{E}(\|A\theta_{i(j)}P_{i(j)}a_{i(j)}(a_{i(j)}^T x_{k(j)} - b_{i(j)})\|^2) - 2\gamma \mathbb{E}(\langle Ax_j - b, A\theta_{i(j)}P_{i(j)}a_{i(j)}(a_{i(j)}^T x_{k(j)} - b_{i(j)})\rangle)
$$

$$
\geq \mathbb{E}(\|Ax_j - b\|^2) - 2\gamma \mathbb{E}(\langle Ax_j - b, A\theta_{i(j)}P_{i(j)}a_{i(j)}(a_{i(j)}^T x_{k(j)} - b_{i(j)})\rangle)
$$

$$
\geq \mathbb{E}(\|Ax_j - b\|^2) - 2\gamma \mathbb{E}(\langle Ax_j - b, A\theta_{i(j)}P_{i(j)}a_{i(j)}(a_{i(j)}^T x_{k(j)} - b_{i(j)})\rangle)
$$

$$
= \mathbb{E}(\|Ax_j - b\|^2) - 2\gamma \mathbb{E}(\langle A^T (Ax_j - b), a_{i(j)}(a_{i(j)}^T x_{k(j)} - b_{i(j)})\rangle)
$$

$$
= \mathbb{E}(\|Ax_j - b\|^2) - \frac{2\gamma}{m} \mathbb{E}(A^T (Ax_j - b), A^T (Ax_{k(j)} - b))
$$

$$
\geq E(\|Ax_j - b\|^2) - \frac{\gamma}{m} \mathbb{E}(\|A^T (Ax_j - b)\|^2 + \|A^T (Ax_{k(j)} - b)\|^2)
$$

$$
\geq E(\|Ax_j - b\|^2) - \frac{\gamma \lambda_{\max}}{m} \mathbb{E}(\|Ax_j - b\|^2 + \|Ax_{k(j)} - b\|^2)
$$

$$
= \left(1 - \frac{\gamma \lambda_{\max}}{m}\right) \mathbb{E}(\|Ax_j - b\|^2) - \frac{\gamma \lambda_{\max}}{m} \mathbb{E}(\|Ax_{k(j)} - b\|^2).
$$

(23)
We can use this bound to show that the left-hand inequality in (21) holds for \( j = 0 \). By setting \( j = 0 \) in (21) and noting that \( k(0) = 0 \), we obtain

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

From the second bound in (5), we have

\[
1 - \frac{2\gamma \lambda_{\max}}{m} \geq 1 - \frac{\rho - 1}{\rho^r + 1} = 1 - \frac{1}{\rho^r} \geq \rho^{-1},
\]

where the last inequality follows from \( \rho > 1 \). By substituting into (23), we obtain \( \rho^{-1} \mathbb{E}(\|Ax_0 - b\|^2) \leq \mathbb{E}(\|Ax_1 - b\|^2) \). For the inductive step, we use (23) again, assuming that the left-hand inequality in (5) holds up to stage \( j \), and thus that

\[
\mathbb{E}(\|Ax_j - b\|^2) \geq \rho^{-\tau} \mathbb{E}(\|Ax_{k(j)} - b\|^2),
\]

provided that \( 0 \leq j - k(j) \leq \tau \), as assumed (3). By substituting into the left-hand side of (23) again, we obtain

\[
\mathbb{E}(\|Ax_{j+1} - b\|^2) \geq \left(1 - \frac{\gamma \lambda_{\max}}{m}\right) \mathbb{E}(\|Ax_j - b\|^2) - \frac{\gamma \rho^\tau \lambda_{\max}}{m} \mathbb{E}(\|Ax_j - b\|^2)
\]

\[
\geq \left(1 - \frac{2\gamma \rho^\tau \lambda_{\max}}{m}\right) \mathbb{E}(\|Ax_j - b\|^2).
\]

From the second bound in (5), we have

\[
1 - \frac{2\gamma \rho^\tau \lambda_{\max}}{m} \geq 1 - \frac{\rho - 1}{\rho} = \rho^{-1}.
\]

We conclude that the left-hand side inequality in (5) holds for all \( j \).

At this point, we have shown that both inequalities in (5) are satisfied for all \( j \). We next prove (7). Consider the expansion of \( \|x_{j+1} - x_{j+1}^*\|^2 \):

\[
\|x_{j+1} - x_{j+1}^*\|^2 = \|x_j - \gamma \theta_{i(j)} P_{i(j)} a_i(j) (a_i^T(k(j)) x_{k(j)} - b_{i(j)}) - x_{j+1}^*\|^2
\]

\[
\leq \|x_j - \gamma \theta_{i(j)} P_{i(j)} a_i(j) (a_i^T(k(j)) x_{k(j)} - b_{i(j)}) - x_{j+1}^*\|^2
\]

\[
= \|x_j - x_j^*\|^2 + \gamma^2 \theta_{i(j)}^2 \|P_{i(j)} a_i(j) (a_i^T(k(j)) x_{k(j)} - b_{i(j)})\|^2 - 2 \gamma \langle x_j - x_j^*, \theta_{i(j)} P_{i(j)} a_i(j) (a_i^T(k(j)) x_{k(j)} - b_{i(j)}) \rangle
\]

\[
= \|x_j - x_j^*\|^2 + \gamma^2 \theta_{i(j)}^2 \|P_{i(j)} a_i(j) (a_i^T(k(j)) x_{k(j)} - b_{i(j)})\|^2 - \underbrace{2 \gamma \langle x_{k(j)} - x_j^*, \theta_{i(j)} P_{i(j)} a_i(j) (a_i^T(k(j)) x_{k(j)} - b_{i(j)}) \rangle}_{T_4} + \underbrace{2 \gamma \langle x_{k(j)} - x_j, \theta_{i(j)} P_{i(j)} a_i(j) (a_i^T(k(j)) x_{k(j)} - b_{i(j)}) \rangle}_{T_5},
\]

\[
(25)
\]
Next, we estimate the expectations of \( T_4, T_5, \) and \( T_6 \). For \( T_4 \), we have

\[
\mathbb{E}(T_4) = \mathbb{E}(\theta_{i(j)}^2 || P_t(i(j)a_{i(j)}(a_{i(j)}^T x_{k(j)} - b_{i(j)})||^2) \\
= \mathbb{E}(\theta_{i(j)}^2 E_{i(j)}(P_t(i(j)a_{i(j)}(a_{i(j)}^T x_{k(j)} - b_{i(j)}))||^2) \\
= \theta_{i(j)}^2 \mathbb{E} \left( \frac{1}{\theta_{i(j)}} \sum_{t \in \text{supp}(a_{i(j)})} (a_{i(j)}^T x_{k(j)} - b_{i(j)})^2 a_{i(j)}^T P_t a_{i(j)} \right) \\
= \mathbb{E}(\theta_{i(j)}^2 a_{i(j)}(a_{i(j)}^T x_{k(j)} - b_{i(j)})^2) \\
\leq \mu \mathbb{E}(a_{i(j)}^T x_{k(j)} - b_{i(j)})^2 \\
= \frac{\mu}{m} \mathbb{E}(\|Ax_{k(j)} - b\|^2), \tag{26}
\]

For \( T_5 \), we have

\[
\mathbb{E}(T_5) = \mathbb{E}((x_{k(j)} - x_j^*, \theta_{i(j)} P_t(i(j)a_{i(j)}(a_{i(j)}^T x_{k(j)} - b_{i(j)}))) \\
= \mathbb{E}((x_{k(j)} - x_j^*, E_{i(j)}(\theta_{i(j)} P_t(i(j)a_{i(j)}(a_{i(j)}^T x_{k(j)} - b_{i(j)}))) \\
= \mathbb{E}((x_{k(j)} - x_j^*, a_{i(j)}(a_{i(j)}^T x_{k(j)} - b_{i(j)}))) \\
= \frac{1}{m} \mathbb{E}(\|Ax_{k(j)} - b\|^2). \tag{27}
\]

By following a derivation similar to (26) for \( T_6 \), we obtain

\[
\mathbb{E}(T_6) = \mathbb{E}((x_{k(j)} - x_j, \theta_{i(j)} P_t(i(j)a_{i(j)}(a_{i(j)}^T x_{k(j)} - b_{i(j)}))) \\
= \mathbb{E} \left( \gamma \sum_{d=k(j)}^{j-1} (\theta_{i(d)} P_t(d) a_{i(d)}(a_{i(d)}^T x_{k(d)} - b_{i(d)}), \theta_{i(j)} P_t(i(j)a_{i(j)}(a_{i(j)}^T x_{k(j)} - b_{i(j)}))) \right) \\
= \frac{\gamma}{m^2} \mathbb{E} \left( \sum_{d=k(j)}^{j-1} \langle A^T (Ax_{k(d)} - b), A^T (Ax_{k(j)} - b) \rangle \right) \\
\leq \frac{\gamma}{m^2} \mathbb{E} \left( \sum_{d=k(j)}^{j-1} \|A^T (Ax_{k(d)} - b)\|^2 + \|A^T (Ax_{k(j)} - b)\|^2 \right) \\
\leq \frac{\gamma \lambda_{\text{max}}}{2m^2} \mathbb{E} \left( \sum_{d=k(j)}^{j-1} \|Ax_{k(d)} - b\|^2 + \|Ax_{k(j)} - b\|^2 \right). \tag{28}
\]

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

\[
k(j) - \tau \leq k(d) \leq j - 2 \leq k(j) + \tau - 1,
\]

it follows from (26) that

\[
\|Ax_{k(d)} - b\|^2 \leq \rho^\tau \|Ax_{k(j)} - b\|^2
\]
Thus from (28), we have
\[ E(T_6) \leq \gamma \tau \lambda_{\text{max}} (1 + \rho^\tau) \mathbb{E}(\|Ax_{k(j)} - b\|^2) \leq \frac{\gamma \tau \lambda_{\text{max}} \rho^\tau}{m^2} \mathbb{E}(\|Ax_{k(j)} - b\|^2). \] (29)

By substituting (26), (27), and (29) into (25), we obtain
\[ E(\|x_{j+1} - x_j^*\|^2) \leq E(\|x_j - x_j^*\|^2) - \left( \frac{2\gamma - \mu \gamma}{m} - \frac{2\gamma^2 \tau \lambda_{\text{max}} \rho^\tau}{m^2} \right) E(\|Ax_{k(j)} - b\|^2). \]

Since by (4) and (5), we have
\[ \frac{2\gamma - \mu \gamma}{m} - \frac{2\gamma^2 \tau \lambda_{\text{max}} \rho^\tau}{m^2} = \frac{\gamma}{m}(2 - \psi \gamma) > 0, \]
we have from the bound above that
\[ E(\|x_{j+1} - x_j^*\|^2) \leq E(\|x_j - x_j^*\|^2) - \frac{\gamma}{m}(2 - \psi \gamma) E(\|Ax_{k(j)} - b\|^2) \leq \frac{\gamma}{m}(2 - \psi \gamma) \lambda_{\text{min}} E(\|x_{k(j)} - x_{k(j)}^*\|^2) \leq \left( 1 - \frac{\lambda_{\text{min}} \gamma}{m}(2 - \psi \gamma) \right) E(\|x_j - x_j^*\|^2), \]
where the second line implies that \( E(\|x_j - x_j^*\|^2) \) is monotonically decreasing, and the last line is obtained by the implication from the second line. This completes the proof of (7).

Proof of Corollary 2

Proof. Note first that for \( \rho \) defined by (9), and using (8), we have
\[ \rho^\tau \leq \rho^{\tau + 1} = \left[ 1 + \frac{2e \lambda_{\text{max}}}{m} \right] \left( \frac{m}{2e \lambda_{\text{max}}} \right)^{\frac{2e \lambda_{\text{max}}(\tau + 1)}{m}} \leq e^{\frac{2e \lambda_{\text{max}}(\tau + 1)}{m}} \leq e. \]

Thus from the definition of \( \psi \) (4), and using (8) again, we have
\[ \psi = \mu + \frac{2e \lambda_{\text{max}} \tau \rho^\tau}{m} \leq \mu + \frac{2e \tau \lambda_{\text{max}}}{m} \leq \mu + 1. \] (30)

We show now that the steplength parameter choice \( \gamma = 1/\psi \) satisfies all the bounds in (5), by showing that the second and third bounds are implied by the first. For the second bound, we have
\[ \frac{m(\rho - 1)}{2e \lambda_{\text{max}} \rho^{\tau + 1}} \geq \frac{m(\rho - 1)}{2e \lambda_{\text{max}}} \geq \tau + 1 \geq 1 \geq \frac{1}{\psi}, \]
where the second inequality follows from (9) and the final inequality follows from the definition of \( \psi \) in (4) and the fact that \( \psi > \mu \geq 1. \)
For the third bound in (8), we have (by taking squares) that

\[
m^2 \frac{(\rho - 1)}{\rho^r(m\alpha^2 + \lambda_{\text{max}}^2 \tau^2)} = \frac{2 me \lambda_{\text{max}}^2 (\tau + 1)}{\rho^r(m\alpha^2 + \lambda_{\text{max}}^2 \tau^2)} = \frac{1}{\frac{\mu^2}{2(\tau + 1)} + \frac{\lambda_{\text{max}} \tau e}{2m(\tau + 1)}} \geq \frac{1}{\frac{\mu^2}{2(\tau + 1)} + \frac{\tau}{4(\tau + 1)^2}} = \frac{1}{\mu^2} \geq \frac{1}{\psi^2}. \tag{from the lower bound of \( m \) in (8)}
\]

We can thus set \( \gamma = 1/\psi \), and by substituting this choice into (7) and using (30), we obtain (10).

**Proof of Theorem 3**

**Proof.** From Markov’s inequality, we have

\[
\mathbb{P}(\|x_j - x_j^*\|^2 \geq \epsilon) \leq \epsilon^{-1} \mathbb{E}(\|x_j - x_j^*\|^2) \\
\leq \epsilon^{-1} \left(1 - \frac{\lambda_{\text{min}}}{m(\mu + 1)}\right)^j \|x_0 - x_0^*\|^2 \\
\leq \epsilon^{-1}(1 - c)^{(1/c)} \left|\log \frac{\|x_0 - x_0^*\|^2}{\eta^2}\right| \|x_0 - x_0^*\|^2 \quad \text{(with } c = \frac{\lambda_{\text{min}}}{m(\mu + 1)})
\]

\[
\leq \epsilon^{-1} \|x_0 - x_0^*\|^2 e^{-\left|\log \frac{\|x_0 - x_0^*\|^2}{\eta^2}\right|} \\
= \eta e^{\log \frac{\|x_0 - x_0^*\|^2}{\eta^2}} e^{-\left|\log \frac{\|x_0 - x_0^*\|^2}{\eta^2}\right|} \\
\leq \eta,
\]

where the second inequality applies (10), the third inequality uses the definition of \( j \) (11), and the second last inequality uses the inequality \((1 - c)^{1/c} \leq e^{-1} \forall c \in (0, 1)\), which completes the proof. 

**References**

H. Avron, A. Druinsky, and A. Gupta. A randomized asynchronous linear solver with provable convergence rate. *arXiv:1304.6475*, 2013.

D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and distributed computation: numerical methods*. Pentice Hall, 1989.

Y. Censor. Row-action methods for huge and sparse systems and their applications. *SIAM Review*, 23(4):444–466, 1981.
Y. Censor, D. Gordon, and R. Gordon. Component averaging: An efficient iterative parallel algorithm for large and sparse unstructured problems. *Parallel Comput.*, 27(6):777–808, 2001.

Y. C. Eldar and D. Needell. Acceleration of randomized Kaczmarz method via the Johnson-Lindenstrauss lemma. *Numerical Algorithms*, 58(2):163–177, 2011.

M. C. Ferris and O. L. Mangasarian. Parallel variable distribution. *SIAM Journal on Optimization*, 4(4):815–832, 1994.

A. Galantai. On the rate of convergence of the alternating projection method in finite dimensional spaces. *Journal of Mathematical Analysis and Applications*, 310:30–44, 2005.

A. J. Hoffman. On approximate solutions of systems of linear inequalities. *Journal of Research of the National Bureau of Standards*, 49(4):263–265, 1952.

S. Kaczmarz. Angenaherte auflösung von systemen linearer gleichungen. *Bulletin International de l’Académie Polonaise des Sciences et des Letters*, 35:355–357, 1937.

D. Leventhal and A. S. Lewis. Randomized methods for linear constraints: Convergence rates and conditioning. *Mathematics of Operations Research*, 35(3):641–654, 2010.

J. Liu and S. J. Wright. An accelerated randomized Kaczmarz algorithm. *arXiv:1310.2887*, 2013.

J. Liu, S. J. Wright, C. Ré, and V. Bittorf. An asynchronous parallel stochastic coordinate descent algorithm. *arXiv: 1311.1873*, 2013.

D. Needell. Randomized Kaczmarz solver for noisy linear systems. *BIT Numerical Mathematics*, 50(2):1422–1436, 2010.

A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19:1574–1609, 2009.

F. Niu, B. Recht, C. Ré, and S. J. Wright. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. *Advances in Neural Information Processing Systems 24*, pages 693–701, 2011.

C. Popa. 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, 1999.

P. Richtárik and M. Takáč. Parallel coordinate descent methods for big data optimization. *ArXiv: 1212.0873*, 2012.

T. Strohmer and R. Vershynin. A randomized Kaczmarz algorithm with exponential convergence. *Journal of Fourier Analysis and Applications*, 15:262–278, 2009.

R. Vershynin. Introduction to the non-asymptotic analysis of random matrices. *ArXiv: 1011.3027*, 2011.

A. Zouzias and N. M. Freris. Randomized extended Kaczmarz for solving least-squares. *arXiv:1205.5770v2*, 2012.