2014

Block Kaczmarz Method with Inequalities

Jonathan Briskman
Claremont McKenna College

CLAREMONT MCKENNA COLLEGE

Block Kaczmarz Method with Inequalities

SUBMITTED TO

Professor Deanna Needell

AND

DEAN NICHOLAS WARNER

BY

Jonathan Briskman

for

SENIOR THESIS

Spring 2014

April 28, 2014

Abstract

The Kaczmarz method is an iterative algorithm that solves overdetermined systems of linear equalities. This paper studies a system of linear equalities and inequalities. We use the block version of the Kaczmarz method applied towards the equalities with the simple randomized Kaczmarz scheme for the inequalities. This primarily involves combining Needell and Tropp's work on the block Kaczmarz method with the application of a randomized Kaczmarz approach towards a system of equalities and inequalities performed by Leventhal and Lewis. We give an expected linear rate of convergence for this kind of system and find that using the block Kaczmarz scheme for the equalities can improve the rate compared to the simple Kaczmarz method.

Contents

Chapter 1

Introduction

The Kaczmarz method [16] is an iterative algorithm for solving linear systems of equations. Each iteration projects from one constraint to the next in a cyclic manner, coverging to the solution at a linear rate.[1] This method was rediscovered for computer tomography, referred to as "algebraic reconstruction technique" (ART) [11], and is also used as a special case of Projection onto Convex Sets (POCS) for signal and image processing [9, 27]. The simple Kaczmarz method has been modified for systems of linear equalities and inequalities [17]. It has also been adapted in the form of the block Kaczmarz algorithm. Unlike the simple Kaczmarz algorithm, which enforces a single constraint at each iteration, the block Kaczmarz approach enforces multiple constraints simultaneously [22]. This paper attempts to demonstrate convergence for a system of linear equalities and inequalities by combining a randomized block Kaczmarz method for the equalities with a randomized Kaczmarz algorithm for the inequalities. These results indicate that the block Kaczmarz method can be used for a system of equalities and inequalities, and in some cases may quicken convergence.

[1]Mathematicians often use the term *exponential convergence* for the concept numerical analysts call *linear convergence*.

1.1 Model and Notation

Consider a linear system

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \tag{1.1.1}$$

where $\boldsymbol{A}$ is a real or complex $n \times d$ matrix.

The ℓ_p vector norm for $p \in [1,\infty]$ is denoted $\|\cdot\|_p$, while $\|\cdot\|$ is the spectral norm and $\|\cdot\|_{\mathrm{F}}$ refers to the Frobenius norm. $\lambda_{\min}$ and $\lambda_{\max}$ are the algebraic minimum and maximum eigenvalues for a Hermitian matrix. For $n \times d$ matrix $\boldsymbol{A}$, the singular values are arranged such that

$$\sigma_{\max}(\boldsymbol{A}) := \sigma_1(\boldsymbol{A}) \geq \sigma_2(\boldsymbol{A}) \geq \cdots \geq \sigma_{\min\{n,d\}}(\boldsymbol{A}) =: \sigma_{\min}(\boldsymbol{A}).$$

In this paper we assume that each row $\boldsymbol{a_i}$ of $\boldsymbol{A}$ has

$$\|\boldsymbol{a}_i\|_2 = 1 \quad \text{for each } i = 1,\ldots,n. \tag{1.1.2}$$

$\boldsymbol{A}$ is called standardized when (1.1.2) is true.

The condition number $\kappa(\boldsymbol{A}) := \sigma_{\max}(\boldsymbol{A})/\sigma_{\min}(\boldsymbol{A})$. The Moore-Penrose pseudoinverse of matrix $\boldsymbol{A}$ is denoted $\boldsymbol{A}^\dagger$. For matrix $\boldsymbol{A}$ with full row rank, the pseudoinverse $\boldsymbol{A}^\dagger := \boldsymbol{A}^*(\boldsymbol{A}\boldsymbol{A}^*)^{-1}$.

Now consider a system of linear equalities and inequalities with S as the nonempty set of feasible solutions to the linear system. Each row i is set so that

$$\boldsymbol{a}_i^T\boldsymbol{x} \leq \boldsymbol{b}_i \;\; (i \in I_\leq) \tag{1.1.3}$$

$$\boldsymbol{a}_i^T\boldsymbol{x} = \boldsymbol{b}_i \;\; (i \in I_=),$$

where the set of rows $\{1,2,...,n\}$ are partitioned such that the first n_e rows are equalities

($\{1,2,...,n_e\} \in I_=$), and the remaining n_{in} rows are inequalities ($\{n_e+1, n_e+2,...,n\} \in I_\leq$). $\boldsymbol{A}_=$ is the submatrix of rows that are equalities in $\boldsymbol{A}$, and $\boldsymbol{A}_\leq$ represents the submatrix of rows of the inequalities. The rows can be arranged such that

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_= \\ \boldsymbol{A}_\leq \end{bmatrix}. \tag{1.1.4}$$

The expected value for function f over these submatrices are denoted $\mathbb{E}_=\left[f\right] := \mathbb{E}\left[f|i \in I_=\right]$ and $\mathbb{E}_\leq\left[f\right] := \mathbb{E}\left[f|i \in I_\leq\right]$.

The error bound for this system of linear inequalities uses function $e : \mathbf{R}^n \to \mathbf{R}^n$ defined as

$$e(y)_i = \begin{cases} y_i^+ & : i \in I_\leq \\ y_i & : i \in I_= \end{cases}, \tag{1.1.5}$$

where for vector $\boldsymbol{x} \in \mathbf{R}^m$, $\boldsymbol{x}^+$ is defined as $(\boldsymbol{x}^+)_i := \max(\boldsymbol{x_i}, 0)$ [17]. In addition, the distance between vector $\boldsymbol{x}$ and set S is denoted

$$d(\boldsymbol{x}, S) := \min_{\boldsymbol{w} \in S} \|\boldsymbol{x} - \boldsymbol{w}\|_2 = \|\boldsymbol{x} - P_S(\boldsymbol{x})\|.$$

$P_S(\boldsymbol{x})$ is the projection of $\boldsymbol{x}$ onto S.

1.2 Details of the Kaczmarz Method

The simple Kaczmarz method is an iterative algorithm that approximates a solution to the system in (1.1.1). It takes an arbitrary initial approximation $\boldsymbol{x}_0$, where at each iteration j the current iterate is projected orthogonally onto the solution hyperplane

$\langle \boldsymbol{a}_i, \boldsymbol{x} \rangle = \boldsymbol{b}_i$, using the algorithm

$$\boldsymbol{x}_{j+1} = \boldsymbol{x}_j + \frac{\boldsymbol{b}_i - \langle \boldsymbol{a}_i, \boldsymbol{x}_j \rangle}{\|\boldsymbol{a}_i\|_2^2} \boldsymbol{a}_i, \tag{1.2.1}$$

where $i = j \bmod n + 1$.

To demonstrate convergence for this method, consider the extremely simple example of a 2×2 matrix $\boldsymbol{A}$. Let us examine a system of two equations and two unknowns: $2x+3y = 9$ and $x-2y = 1$. Basic algebra can give us the solution $x = 3$ and $y = 1$. Matrices can also be used to give us the desired result:

$$\boldsymbol{A} = \begin{bmatrix} 2 & 3 \\ 1 & -2 \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 9 \\ 1 \end{bmatrix}$$

and

$$\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b},$$

where we can solve for $\boldsymbol{x}$ given $\boldsymbol{A}$ and $\boldsymbol{b}$ by multiplying $\boldsymbol{A}^{-1}$ on both sides.

$$\boldsymbol{x} = \boldsymbol{A}^{-1} \cdot \boldsymbol{b}.$$

For large and non-square matrices where the linear algebra approach is computationally extensive (inverting large matrices is challenging), the Kaczmarz approach may be desirable. This involves taking an arbitrary initial guess $\boldsymbol{x}_0$ and use of update rule (1.2.1). To demonstrate this using the simple example above, we can set arbitrary initial approximation $\boldsymbol{x}_0 = [-1 \;\; 1]^T$. Using (1.2.1), we project onto the equation represented in the first row of $\boldsymbol{A}$ and $\boldsymbol{b}$. This calculation gives us $\boldsymbol{x}_1 = [3/13 \;\; 37/13]^T$, which satisfies the first equation $(2x+3y=9)$. Next, we project onto the second equation. The process continues, projecting from one equation to the other. As shown in Figure 1-1, approximation

$\boldsymbol{x}_k$ converges to $\boldsymbol{x} = [3 \ \ 1]^T$.

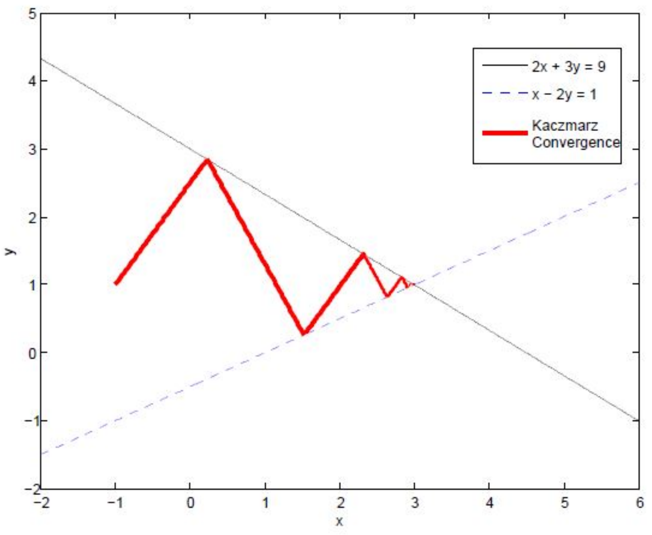

Figure 1-1 (Convergence Using Simple Kaczmarz Method).

The randomized Kaczmarz method uses a random selection method for the selection of the rows, such that each row i is selected with probability proportional to $\|\boldsymbol{a}_i\|_2^2$. Randomization provides an algorithm that is both simple to analyze and enforce [22]. Numerical simulations have shown an improved convergence rate for a randomized Kaczmarz method compared to the simple method in which the rows are selected sequentially [28]. To demonstrate how a randomized selection mechanism improves convergence, consider another example:

$$\boldsymbol{A} = \begin{bmatrix} 2 & 3 \\ 4 & 5 \\ -6 & 1 \\ 1 & -2 \\ 1 & -5 \end{bmatrix}, \ \boldsymbol{x} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}, \ \boldsymbol{b} = \begin{bmatrix} 9 \\ 17 \\ -17 \\ 1 \\ -2 \end{bmatrix}$$

and

$$\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b}.$$

If the rows are selected sequentially, it is possible that convergence will be quite slow. See Figure 1-2 [Left Panel] for convergence using the cyclic pattern. Compared to the randomized method, displayed in Figure 1-2 [Right Panel], convergence is noticeably slower. Thus in practice a randomized Kaczmarz method is preferred to the cyclic selection method.

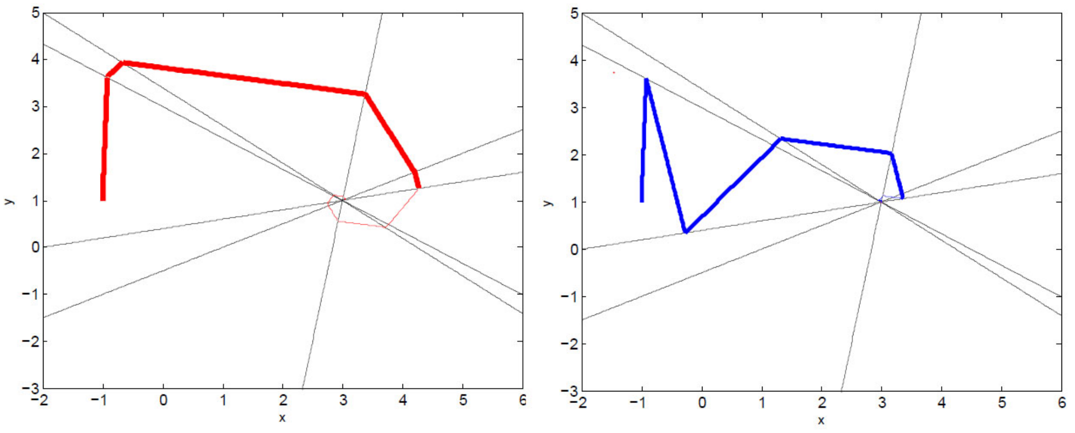

Figure 1-2 (Convergence Using Cyclic vs Randomized Kaczmarz Method). **[Left]** The simple Kaczmarz method with cyclic selection mechanism. Convergence is shown in red. The first 5 projections are in bold. **[Right]** The simple Kaczmarz method with a uniform random selection mechanism. Convergence is shown in blue. Again, the first 5 projections are in bold. Note how this algorithm has progressed much closer to the solution through 5 projections than with the cyclic selection approach.

For non-standardizd matrices, rows are selected with probability proportional to their norms [28]. In this paper we assume standardized matrices where each row has unit norm, so each row is selected uniformly at random from {1,...,n} with our simple randomized Kaczmarz approach. Strohmer and Vershynin [28] find a linear rate of convergence that depends on the scaled condition number of $\boldsymbol{A}$, and not on the number of equations n.

$$\mathbb{E}\|\boldsymbol{x}_j - \boldsymbol{x}_\star\|_2^2 \leq \left[1 - \frac{1}{\kappa(\boldsymbol{A})^2}\right]^j \|\boldsymbol{x}_0 - \boldsymbol{x}_\star\|_2^2. \tag{1.2.2}$$

In the case of a linear system of equalities and inequalities described in (1.1.3), the randomized Kaczmarz method can be adapted to fit this system. Leventhal and Lewis [17] apply the Kaczmarz method to a consistent system of linear equalities and inequalities. At each iteration j, the previous iterate only projects onto the solution hyperplane if the inequality is not already satisfied. If the inequality is satisfied for row i selected at iteration j ($\boldsymbol{a}_i^T\boldsymbol{x} \leq \boldsymbol{b}_i$), the approximation $\boldsymbol{x}_j$ is set equal to $\boldsymbol{x}_{j-1}$ [17]. The update rule for this algorithm is

$$\boldsymbol{x}_{j+1} = \boldsymbol{x}_j - \frac{e(\boldsymbol{a}_i^T\boldsymbol{x}_j - \boldsymbol{b}_i)}{\|\boldsymbol{a}_i\|_2^2}\boldsymbol{a}_i. \tag{1.2.3}$$

This algorithm converges linearly in expectation, with

$$\mathbb{E}\left[d(\boldsymbol{x}_j, S)^2 | \boldsymbol{x}_{j-1}\right] \leq d(\boldsymbol{x}_{j-1}, S)^2 - \frac{\|e(\boldsymbol{A}\boldsymbol{x}_{j-1} - \boldsymbol{b})\|_2^2}{\|\boldsymbol{A}\|_F^2}$$

or equivalently for a standardized matrix $\boldsymbol{A}$

$$\mathbb{E}\left[d(\boldsymbol{x}_j, S)^2 | \boldsymbol{x}_{j-1}\right] \leq \left[1 - \frac{\sigma_{\min}^2(\boldsymbol{A})}{n}\right] \cdot d(\boldsymbol{x}_{j-1}, S)^2. \tag{1.2.4}$$

for iteration j [17]. This results in expected error bound

$$\mathbb{E}\left[d(\boldsymbol{x}_j, S)^2\right] \leq \left[1 - \frac{\sigma_{\min}^2(\boldsymbol{A})}{n}\right]^j \cdot d(\boldsymbol{x}_0, S)^2. \tag{1.2.5}$$

The randomized Kaczmarz method has been modified by Elfving [8] and Eggermont et al. [7] in the form of the block Kaczmarz method to improve the convergence rate in certain cases. The block Kaczmarz scheme first partitions the rows $\{1, ..., n\}$ into m blocks, denoted $\tau_{\{1,...,m\}}$. Instead of selecting one row per iteration as done with the simple Kaczmarz method, the block Kaczmarz algorithm chooses a block uniformly at random at each iteration. Thus the block Kaczmarz method enforces multiple constraints simultaneously. At each iteration, the previous iterate $\boldsymbol{x}_{j-1}$ is projected onto the solution space $\boldsymbol{A}_\tau \boldsymbol{x} = \boldsymbol{b}_\tau$, which enforces the set of equations in block τ [22]. $\boldsymbol{A}_\tau$ and $\boldsymbol{b}_\tau$ are written as the row submatrix of $\boldsymbol{A}$ and the subvector of $\boldsymbol{b}$ indexed by τ respectively, giving equation to update approximation $\boldsymbol{x}_j$ as

$$\boldsymbol{x}_j = \boldsymbol{x}_{j-1} + (\boldsymbol{A}_\tau)^\dagger (\boldsymbol{b}_\tau - \boldsymbol{A}_\tau \boldsymbol{x}_{j-1}). \tag{1.2.6}$$

1.3 Pavings for the Block Kaczmarz Method

Needell and Tropp [22] define an $(m,\ \alpha,\ \beta)$ row paving of matrix $\boldsymbol{A}$ as a partition $T = \{\tau_1, ... \tau_m\}$ of the row indices such that

$$\alpha \le \lambda_{\min}(\boldsymbol{A}_\tau \boldsymbol{A}_\tau^*) \quad \text{and} \quad \lambda_{\max}(\boldsymbol{A}_\tau \boldsymbol{A}_\tau^*) \le \beta \quad \text{for each } \tau \in T.$$

The size of the paving, or number of blocks, is m, and α and β are the lower and upper paving bounds respectively. Needell and Tropp show that these values determine the performance of the algorithm, with expected error bound for matrix $\boldsymbol{A}$ with full column rank and paving T

$$\mathbb{E}\|\boldsymbol{x}_j - \boldsymbol{x}_\star\|_2^2 \le \left[1 - \frac{\sigma_{\min}^2(\boldsymbol{A})}{\beta m}\right]^j \|\boldsymbol{x}_0 - \boldsymbol{x}_\star\|_2^2. \tag{1.3.1}$$

Therefore the convergence rate depends on size m and upper bound β, with the algorithm's performance improving with low values of m and β, and large $\sigma_{\min}^2(\boldsymbol{A})$ [22].

Vershynin [30, Cor. 1.5] and Tropp [29, Thm. 1.2] built on work from [2, 3] to demonstrate that every standardized matrix has a good row paving.

Proposition 1.3.1 (Existence of Good Row Pavings) *For fixed $\delta \in (0,1)$ and standardized*

$n \times d$ matrix $\boldsymbol{A}$, $\boldsymbol{A}$ admits a row paving satisfying

$$m \leq \mathrm{C}_{\text{pave}} \cdot \delta^{-2} \|\boldsymbol{A}\|^2 \log(1+n) \quad \textit{and} \quad 1-\delta \leq \alpha \leq \beta \leq 1+\delta.$$

where C_{pave} *is a absolute constant.*

This result is important because the existence of a good row paving enables the use of the block Kaczmarz method for any standardized matrix $\boldsymbol{A}$.

A paving can be chosen such that convergence will depend only on the condition number $\kappa(\boldsymbol{A})$ and a universal paving constant [22, Cor. 1.4]. Given row pavings described in Proposition 1.3.1 and $\delta = 1/2$, Needell and Tropp [22] find expected convergence

$$\mathbb{E}\|\boldsymbol{x}_j - \boldsymbol{x}_\star\|^2 \leq \left[1 - \frac{1}{6\mathrm{C}_{\text{pave}}\kappa^2(\boldsymbol{A})\log(1+n)}\right]^j \|\boldsymbol{x}_0 - \boldsymbol{x}_\star\|_2^2.$$

Depending on the characteristics of the submatrix $\boldsymbol{A}_\tau$, the block method can provide better convergence than the simple method. Note how the error bounds given in (1.2.2) and (1.3.1) can be written in the form

$$\mathbb{E}\|\boldsymbol{x}_j - \boldsymbol{x}_\star\|_2^2 \leq e^{-j\rho} \cdot \|\boldsymbol{x}_0 - \boldsymbol{x}_\star\|_2^2$$

for some per iteration convergence rate ρ. Needell and Tropp [22] give convergence rates

$$\rho_{\text{simp}} \geq \frac{\sigma^2_{\min}(\boldsymbol{A})}{n} \text{ and } \rho_{\text{block}} \geq \frac{\sigma^2_{\min}(\boldsymbol{A})}{\beta m} \tag{1.3.2}$$

for the simple and block methods respectively.

If we assume that the submatrices $\boldsymbol{A}_\tau$ are well conditioned and admit a fast multiply, then the computational cost of the block algorithm (1.2.6) is similar to the cost of the simple update rule in (1.2.1) [22]. In this case, switching to the block method can improve the convergence rate by approximately $n/\beta m$, as the computational cost per iteration is about equal. Therefore when the paving gives small values of β and m relative to n, the block method can greatly increase the convergence rate. When the block Kaczmarz scheme is constructed so that it is the same as the simple Kaczmarz method ($m = n$), $\beta = 1$ so that the convergence rates are equivalent as anticipated.

Next, consider the case when the submatrices $\boldsymbol{A}_\tau$ are unstructured and dense. The block algorithm for this system might require much more arithmetic per iteration, so it will have a higher computational cost. To account for this, Needell and Tropp [22] compare the convergence rate per epoch, not per iteration. An epoch is the minimum number of iterations it could take to touch every row of $\boldsymbol{A}$ once. This assumes that the computational cost of each algorithm is similar per row touched, not per iteration. An epoch is n iterations for the simple Kaczmarz approach and m iterations for the block method. The convergence rates can be compared as

$$n \cdot \rho_{\text{simp}} \geq \sigma^2_{\min}(\boldsymbol{A}) \text{ and } m \cdot \rho_{\text{block}} \geq \frac{\sigma^2_{\min}(\boldsymbol{A})}{\beta} \tag{1.3.3}$$

The per-epoch convergence rate for the block method is worse or equal to that of the simple method. However, Needell and Tropp [22] argue that the block Kaczmarz algorithm can still be preferable to the simple method in some applications, such as signal processing and computer architectures. They found improved per epoch convergence for the block process in numerical simulations as well.

1.4 Contribution

This paper analyzes the system with matrix described in (1.1.1) and (1.1.4) using an algorithm with the block Kaczmarz approach for the equalities contained in $\boldsymbol{A}_=$ and the random Kaczmarz method for the inequalities contained in $\boldsymbol{A}_\leq$. A paving is created for the equality rows with the inequalities excluded. At each iteration, there is given a

Algorithm 1.4.1 Block Kaczmarz Method for a System with Inequalities

Input:

- Matrix $\boldsymbol{A}$ with dimension $n \times d$
- Right-hand side $\boldsymbol{b}$ with dimension n
- Number of rows representing equalities n_e
- Partition $T = \{\tau_1, \dots, \tau_m\}$ of the row indices $\{1, \dots, n_e\}$
- Initial iterate $\boldsymbol{x}_0$ with dimension d
- Convergence tolerance $\varepsilon > 0$

Output: An estimate $\hat{\boldsymbol{x}}$ for the solution to $\min_{\boldsymbol{x}} \| e(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) \|_2^2$

```
$j \leftarrow 0$
repeat
    $j \leftarrow j + 1$
    Draw uniformly at random $q$ from $[0,1]$
    if $q \le \frac{\beta m}{n_e + \beta m}$
        Choose a block $\tau$ uniformly at random from $T$
        $\boldsymbol{x}_j \leftarrow \boldsymbol{x}_{j-1} + (\boldsymbol{A}_\tau)^\dagger(\boldsymbol{b}_\tau - \boldsymbol{A}_\tau \boldsymbol{x}_{j-1})$        { Solve least-squares problem }
    else
        Choose a row $i$ uniformly at random from $\{n_e + 1, \dots, n\}$
        $\boldsymbol{x}_j = \boldsymbol{x}_{j-1} - \frac{e(\boldsymbol{a}_i^T \boldsymbol{x}_{j-1} - \boldsymbol{b}_i)}{\|\boldsymbol{a}_i\|_2^2} \boldsymbol{a}_i$
until $\| e(\boldsymbol{A}\boldsymbol{x}_j - \boldsymbol{b}) \|_2^2 \le \varepsilon^2$
$\hat{\boldsymbol{x}} \leftarrow \boldsymbol{x}_j$
```

probability p of choosing uniformly at random from the blocks of equalities ($p \in [0,1]$), and a $1 - p$ chance of choosing an inequality row uniformly at random. In the case of a block of equalities being selected, the algorithm proceeds by updating $\boldsymbol{x}_j$ using (1.2.6). When an inequality row is selected, $\boldsymbol{x}_j$ is updated using the rule (1.2.3).

The most arithmetically expensive step of Algorithm 1.4.1 is taking the pseudoinverse $\boldsymbol{A}_\tau^\dagger$ when applying update rule (1.2.6) for the blocks of rows representing equalities. This can be performed using an iterative least-squares solver such as `CGLS` [1] to approximate the pseudoinverse when each row of submatrix $\boldsymbol{A}_\tau$ is well-conditioned or there is a fast-multiply method for structured blocks [22].

We propose a selection method that demonstrates an expected linear convergence rate for the system proposed in (1.1.1) and (1.1.4) using Algorithm 1.4.1. As expected, introducing a block Kaczmarz scheme for the equalities tends to increase the per iteration convergence rate relative to a simple randomized Kaczmarz method. This comparison is most applicable when the per iteration computational cost of the two methods are approximately equivalent. For situations where the paving results in submatrix $\boldsymbol{A}_\tau$ which admits a fast multiply and is well conditioned, the computational costs of the two methods are comparable.

In the case of unstructured or dense submatrix $\boldsymbol{A}_\tau$, the block Kaczmarz approach is significantly more computationally expensive per iteration. To account for this, we can compare the convergence rates per epoch instead of per iteration. Theoretical results suggest that Algorithm 1.4.1 slows down the per epoch convergence rate relative to the simple Kaczmarz scheme, or at best results in approximately equivalent convergence. Needell and Tropp [22] ran numerical experiments that suggest results favorable to this theory's predictions for a system with no inequalities. Their experiments yielded faster per epoch convergence when the equalities were blocked. Our numerical simulations indicate that the inequalities (where for $i \in I_\leq$, $\boldsymbol{b}_i = \boldsymbol{A}_i\boldsymbol{x} + \gamma$ for some $\gamma \geq 0$) make convergence less precise for large γ. This results in the linear convergence rate stops much earlier compared to a system without inequalities, such that both our block Kaczmarz approach and the simple Kaczmarz scheme cease to converge after only a few epochs. Thus the methods converge at approximately the same per epoch rate in our experiments. Still, this suggests the experimental results may perform better than theory predicts in some cases.

1.5 Organization

Section 2 lays out our main result, Theorem 2.1, and provides a proof. It also includes analysis of this result. Section 3 explains numerical experiments and compares convergence using simulated results. In Section 4, we discuss an alternative block Kaczmarz approach where the inequalities are blocked as well as the equalities, and give results of experimental simulations for this method. We highlight related works in Section 5. The Appendix includes the MATLAB code for the numerical experiments in Section 4 and Section 5, as well as the examples in Section 1.

Chapter 2

Analysis of the Block Kaczmarz Algorithm for a System with Inequalities

2.1 Main Result and Proof

This section begins with our main result, which gives linear convergence for the method described by Algorithm 1.4.1. The proof combines work on the convergence of the block Kaczmarz algorithm for a system of linear equalities with the application of the randomized Kaczmarz method to a system that includes equalities and inequalities. This primarily builds on the Needell and Tropp's [22] paper on convergence using the block Kaczmarz approach, and the Kaczmarz convergence demonstrated by Leventhal and Lewis [17] for a system with inequalities. The proof methods used in these papers can be combined to give the desired result.

Theorem 2.1.1 (Convergence) *Let $\boldsymbol{A}$ be an $n \times d$ matrix that admits an (m, α, β) row paving for $\boldsymbol{A}_=$. The first n_e rows of $\boldsymbol{A}$ represent equalities and the remaining $n_{in} = n - n_e$ rows are inequalities ($\boldsymbol{a}_i^T \boldsymbol{x} \leq \boldsymbol{b}_i$ $\forall$ row $i \in I_\leq$). Set the probability p of selecting from $\boldsymbol{A}_=$, $p = \frac{\beta m}{n_{in} + \beta m}$ (and the probability of selecting from $\boldsymbol{A}_\leq$ as $1 - p$). Let $\boldsymbol{x}_0$ be an arbitrary ini-*

tial estimate. S is the nonempty feasible region. Then the Algorithm 1.4.1 satisfies for each iteration j = 1,2,3,...,

$$\mathbb{E}\left[d(\boldsymbol{x}_j, S)^2 | \boldsymbol{x}_{j-1}\right] \le \left[1 - \frac{\sigma^2_{\min}(\boldsymbol{A})}{(n_{in} + \beta m)}\right] \cdot d(\boldsymbol{x}_{j-1}, S)^2. \tag{2.1.1}$$

Proof. Let row index i be chosen during iteration j.

If $i \in I_=$, then we set $\boldsymbol{b}_\tau = \boldsymbol{A}_\tau P_S \boldsymbol{x}_{j-1}$ since $P_S \boldsymbol{x}_{j-1} \in S$. Starting with update rule (1.2.6) we have

$$\boldsymbol{x}_j = \boldsymbol{x}_{j-1} + \boldsymbol{A}_\tau^\dagger (\boldsymbol{b}_\tau - \boldsymbol{A}_\tau \boldsymbol{x}_{j-1}).$$

Substituting in our value for $\boldsymbol{b}_\tau$ gives us

$$\boldsymbol{x}_j = \boldsymbol{x}_{j-1} + \boldsymbol{A}_\tau^\dagger \boldsymbol{A}_\tau (P_S \boldsymbol{x}_{j-1} - \boldsymbol{x}_{j-1}).$$

We can subtract $P_S \boldsymbol{x}_{j-1}$ from both sides to obtain

$$\begin{aligned} \boldsymbol{x}_j - P_S \boldsymbol{x}_{j-1} &= \boldsymbol{x}_{j-1} - P_S \boldsymbol{x}_{j-1} - \boldsymbol{A}_\tau^\dagger \boldsymbol{A}_\tau (\boldsymbol{x}_{j-1} - P_S \boldsymbol{x}_{j-1}) \\ &= (\mathbf{I} - \boldsymbol{A}_\tau^\dagger \boldsymbol{A}_\tau)(\boldsymbol{x}_{j-1} - P_S \boldsymbol{x}_{j-1}). \end{aligned}$$

We take the expected value of the norm of both sides as follows:

$$\mathbb{E}\|\boldsymbol{x}_j - P_S \boldsymbol{x}_{j-1}\|_2^2 \le \mathbb{E}\|(I - \boldsymbol{A}_\tau^\dagger \boldsymbol{A}_\tau)(\boldsymbol{x}_{j-1} - P_S \boldsymbol{x}_{j-1})\|_2^2.$$

Analyzing the right-hand side, we can use the following facts.

$$\mathbb{E}\|(\mathbf{I} - \boldsymbol{A}_\tau^\dagger \boldsymbol{A}_\tau)\boldsymbol{u}\|_2^2 = \|\boldsymbol{u}\|_2^2 - \mathbb{E}\|\boldsymbol{A}_\tau^\dagger \boldsymbol{A}_\tau \boldsymbol{u}\|_2^2 \quad \text{and} \quad \mathbb{E}\|\boldsymbol{A}_\tau^\dagger \boldsymbol{A}_\tau \boldsymbol{u}\|_2^2 \ge \frac{1}{\beta m}\|\boldsymbol{A}\boldsymbol{u}\|_2^2.$$

These can be used to applied to our case, resulting in

$$\begin{aligned}
\mathbb{E}\|\boldsymbol{x}_j - P_S\boldsymbol{x}_{j-1}\|_2^2 &\le \mathbb{E}\|(I - \boldsymbol{A}_\tau^\dagger \boldsymbol{A}_\tau)(\boldsymbol{x}_{j-1} - P_S\boldsymbol{x}_{j-1})\|_2^2 \\
&= \|\boldsymbol{x}_{j-1} - P_S\boldsymbol{x}_{j-1}\|_2^2 - \mathbb{E}\|\boldsymbol{A}_\tau^\dagger \boldsymbol{A}_\tau(\boldsymbol{x}_{j-1} - P_S\boldsymbol{x}_{j-1})\|_2^2 \\
&\le \|\boldsymbol{x}_{j-1} - P_S\boldsymbol{x}_{j-1}\|_2^2 - \frac{1}{\beta m}\|\boldsymbol{A}_=(\boldsymbol{x}_{j-1} - P_S\boldsymbol{x}_{j-1})\|_2^2.
\end{aligned}$$

We can substitute our definition of distance from $\boldsymbol{x}_j$ to S, such that

$$\mathbb{E}\left[d(\boldsymbol{x}_j, S)^2 | \boldsymbol{x}_{j-1}\right] \le d(\boldsymbol{x}_{j-1}, S)^2 - \frac{1}{\beta m}\|\boldsymbol{A}_=(\boldsymbol{x}_{j-1} - P_S\boldsymbol{x}_{j-1})\|_2^2.$$

Furthermore, we rewrite the norm on the right-hand side using the error bound in (1.1.5).

$$\mathbb{E}\left[d(\boldsymbol{x}_j, S)^2 | \boldsymbol{x}_{j-1}\right] \le d(\boldsymbol{x}_{j-1}, S)^2 - \frac{1}{\beta m}\sum_{i \in I_=} e(\boldsymbol{A}_=\boldsymbol{x}_{j-1} - \boldsymbol{b}_=)_i^2.$$

Next, consider the alternative case where $i \in I_\le$. For the inequality case, the error bound is given by

$$d(\boldsymbol{x}_j, S)^2 \le d(\boldsymbol{x}_{j-1}, S)^2 - \frac{e(\boldsymbol{A}_\le \boldsymbol{x}_{j-1} - \boldsymbol{b}_\le)_i^2}{\|\boldsymbol{a}_i\|_2^2},$$

and as each row has unit norm, $\|\boldsymbol{a}_i\|_2^2 = 1$, we find

$$\mathbb{E}\left[d(\boldsymbol{x}_j, S)^2 | \boldsymbol{x}_{j-1}\right] \le d(\boldsymbol{x}_{j-1}, S)^2 - \mathbb{E}\left[e(\boldsymbol{A}_\le \boldsymbol{x}_{j-1} - \boldsymbol{b}_\le)_i^2\right]$$

when we take the expectation of both sides. The right term on the right-hand side can be treated similarly to a probability distribution, satisfying

$$\mathbb{E}\left[d(\boldsymbol{x}_j, S)^2 | \boldsymbol{x}_{j-1}\right] \le d(\boldsymbol{x}_{j-1}, S)^2 - \frac{1}{n_{in}}\sum_{i \in I_\le} e(\boldsymbol{A}_\le \boldsymbol{x}_{j-1} - \boldsymbol{b}_\le)_i^2.$$

Combining the results for $i \in I_=$ and $i \in I_\leq$, we can find the expected error bound for the system as

$$\mathbb{E}\left[d(\boldsymbol{x}_j, S)^2 | \boldsymbol{x}_{j-1}\right] = p \cdot \mathbb{E}_= \left[d(\boldsymbol{x}_j, S)\right]^2 + (1-p) \cdot \mathbb{E}_\leq \left[d(\boldsymbol{x}_j, S)\right]^2.$$

Substituting in the error bounds for each case gives us

$$\mathbb{E}\left[d(\boldsymbol{x}_j, S)^2 | \boldsymbol{x}_{j-1}\right] \leq d(\boldsymbol{x}_{j-1}, S)^2 - p \cdot \frac{1}{\beta m} \sum_{i \in I_=} e(\boldsymbol{A}_= \boldsymbol{x}_{j-1} - \boldsymbol{b}_=)_i^2 - (1-p) \cdot \frac{1}{n_{in}} \sum_{i \in I_\leq} e(\boldsymbol{A}_\leq \boldsymbol{x}_{j-1} - \boldsymbol{b}_\leq)_i^2.$$

If we set $p = \frac{\beta m}{n_{in} + \beta m}$, combining the terms on the right-hand side finds

$$\begin{aligned} \mathbb{E}\left[d(\boldsymbol{x}_j, S)^2 | \boldsymbol{x}_{j-1}\right] &\leq d(\boldsymbol{x}_{j-1}, S)^2 - \frac{1}{n_{in} + \beta m} \left[\sum_{i \in I_=} e(\boldsymbol{A}_= \boldsymbol{x}_{j-1} - \boldsymbol{b}_=)_i^2 + \sum_{i \in I_\leq} e(\boldsymbol{A}_\leq \boldsymbol{x}_{j-1} - \boldsymbol{b}_\leq)_i^2\right] \\ &= d(\boldsymbol{x}_{j-1}, S)^2 - \frac{1}{n_{in} + \beta m} \left[\sum_i e(\boldsymbol{A}\boldsymbol{x}_{j-1} - \boldsymbol{b})_i^2\right]. \end{aligned}$$

Substituting back to the norm using the definition of the error bound, we have

$$\mathbb{E}\left[d(\boldsymbol{x}_j, S)^2 | \boldsymbol{x}_{j-1}\right] \leq d(\boldsymbol{x}_{j-1}, S)^2 - \frac{1}{n_{in} + \beta m} \| e(\boldsymbol{A}\boldsymbol{x}_{j-1} - \boldsymbol{b}) \|_2^2,$$

and we reach our proposed error bound

$$\mathbb{E}\left[d(\boldsymbol{x}_j, S)^2 | \boldsymbol{x}_{j-1}\right] \leq \left[1 - \frac{\sigma_{\min}^2(\boldsymbol{A})}{(n_{in} + \beta m)}\right] d(\boldsymbol{x}_{j-1}, S)^2,$$

which concludes the proof.

2.2 Analysis of the Main Result

Theorem 2.1.1 gives linear convergence in expectation for Algorithm 1.4.1. In addition, probability p of selecting from $\boldsymbol{A}_=$ can be chosen so that the expected convergence rate only depends on the number of inequalities n_{in}, paving size m, and upper bound β.

The result from Theorem 2.1.1 gives us error bound

$$\mathbb{E}\left[d(\boldsymbol{x}_j, S)^2\right] \le \left[1 - \frac{\sigma^2_{\min}(\boldsymbol{A})}{(n_{in} + \beta m)}\right]^j \cdot d(\boldsymbol{x}_0, S)^2.$$

This is in the same form as (1.2.5),

$$\mathbb{E}\left[d(\boldsymbol{x}_j, S)^2\right] \le e^{-j\rho} \cdot d(\boldsymbol{x}_0, S)^2.$$

for convergence rate ρ.

The convergence rates for the simple and block Kaczmarz methods for system of equalities and inequalites are

$$\rho_{\text{simp}} \ge \frac{\sigma^2_{\min}(\boldsymbol{A})}{n} \text{ and } \rho_{\text{block}} \ge \frac{\sigma^2_{\min}(\boldsymbol{A})}{n_{in} + \beta m},$$

respectively. It is evident that the expected convergence rate will be faster per iteration than the method used by Leventhal and Lewis [17] for $n_{in} + \beta m < n$, or equivalently $\beta m < n_e$, since the block method utilizes multiple rows at once. This result is similar to that found by [22] given in (1.3.2) when comparing convergence rates for a system of equalities. Both systems give a speed-up in convergence that depends on paving values β and m relative to the number of equalities. In our system, this is different than the total number of rows n, as it excludes the number of inequalities. We provide no evidence that our selection of p is most efficient, or any more efficient than selecting p proportional to the number of equality rows n_e.

In this paper, we count an iteration j towards an epoch only if $\boldsymbol{x}_j \neq \boldsymbol{x}_{j-1}$. In the case of choosing a row that represents an inequality that is already satisfied, no arithmetic is required to update the approximation $\boldsymbol{x}_j$, so that iteration is not counted towards the epoch. The simple method [17] will have approximately n iterations per epoch, compared to $n_{in}+m$ iterations per epoch with the block method. The approximate per epoch convergence rates can be compared as

$$n \cdot \rho_{\text{simp}} \geq \sigma^2_{\min}(\boldsymbol{A}) \quad \text{and} \quad (n_{in}+m)\cdot \rho_{\text{block}} \geq \sigma^2_{\min}(\boldsymbol{A}) \frac{n_{in}+m}{n_{in}+\beta m}$$

This result is similar to that found by Needell and Tropp [22] in (1.3.3), with block convergence rate at best equal to that of the simple convergence rate when $\beta = 1$. The difference in convergence rates between the two methods, however, appears to be dampened by the number of inequalities. This is not surprising, as the inequalities are treated the same by both approaches and thus should result in similar convergence rates.

Overall, our results seem consistent with previous work. The convergence rates for the block Kaczmarz scheme depend on the paving T, as well as the number of inequalities. The number of inequalities simply seem to dampen the difference in rates between the simple and block Kaczmarz methods. Therefore for a system with few inequalities, the change in convergence from implementing a block Kaczmarz approach instead of the simple Kaczmarz method should be similar to that of a system with only equalities.

Chapter 3

Experiments

We use MATLAB to run some experiments using random matrices to test the convergence of the block Kaczmarz method applied to a system of inequalities. In each experiment, we create a random 500 by d matrix $\boldsymbol{A}_{\text{rand}}$, where each element is an independent standard Gaussian random variable. Each entry is then divided by the norm of its row so that the matrix is standardized. The first 400 rows of matrix $\boldsymbol{A}$ compose $\boldsymbol{A}_{=}$, and the remaining 100 rows are set as inequalities of $\boldsymbol{A}_{\leq}$ in the method described by (1.1.3). The experiments are run using the following procedure: For each of 100 trials,

1. Create matrix $\boldsymbol{A}_{\text{rand}}$ in the manner described above.

2. Create $\boldsymbol{b}_{in}$. The first 400 components are set as 0, and the remaining 100 take independent random uniform values from $\left[0, 1 \times 10^{-9}\right]$.

3. Create $\boldsymbol{x}_{\star}$ where each entry is selected independently from a unit normal distribution. Set $\boldsymbol{b} = \boldsymbol{A}_{\text{rand}}\boldsymbol{x}_{\star} + \boldsymbol{b}_{in}$.

4. Pave submatrix $\boldsymbol{A}_{=}$ into 16 blocks with 25 equalities per block without replacement.

5. Draw uniformly at random q from $[0, 1]$.

(a) If $q \le \frac{n_e}{n}$, choose block $\{1, \ldots, 16\}$ uniformly at random and update iterate $\{\boldsymbol{x}_j^{\text{mixed}} : j \ge 0\}$ using (1.2.6).

(b) Else, choose row uniformly at random from $\{401, \ldots, 500\}$ and update iterate $\{\boldsymbol{x}_j^{\text{mixed}} : j \ge 0\}$ using (1.2.3).

(c) Update iterate $\{\boldsymbol{x}_j^{\text{simp}} : j \ge 0\}$ using (1.2.3).

For both the simple and block algorithms, the median, minimum, and maximum values of the residual $\|e(\boldsymbol{A}\boldsymbol{x}_j - \boldsymbol{b})\|_2^2$ of the 100 trials are recorded for each iteration j. See the appendix for the MATLAB code for the simple randomized Kaczmarz method and block Kaczmarz method applied to the inequalities.

Figure 3-1 demonstrates convergence for the block Kaczmarz approach for a $500 \times d$ matrix, with $d = 50, 100, 150, 200$. The results are similar to those found by Leventhal and Lewis [17].

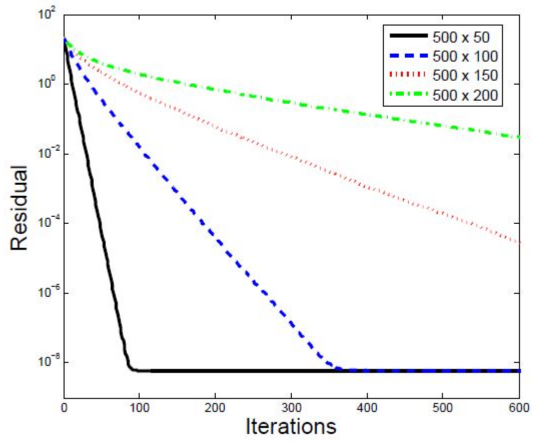

Figure 3-1 (Convergence for a Random Matrix with Block Method). The matrix $\boldsymbol{A}$ is a 500 by d matrices for d = 50, 100, 150, 200. The first 400 rows represent equalities and the last 100 rows are inequalities. The equalities are partitioned uniformly at random into 16 blocks with 25 rows in each block. The approximation error $\|e(\boldsymbol{A}\boldsymbol{x}_j - \boldsymbol{b})\|$ is plotted against iterations j. The plots represent the median of 100 trials.

Figure 3-2 compares the performance of Algorithm 1.4.1 and the simple Kaczmarz method described by Leventhal and Lewis [17]. Matrix $\boldsymbol{A}$ is a 500×50 matrix. Figure 3-2 [left panel] compares convergence per iteration. As the block Kaczmarz method enforces multiple equalities per iteration, it is unsurprising that it performs better in this experiment.

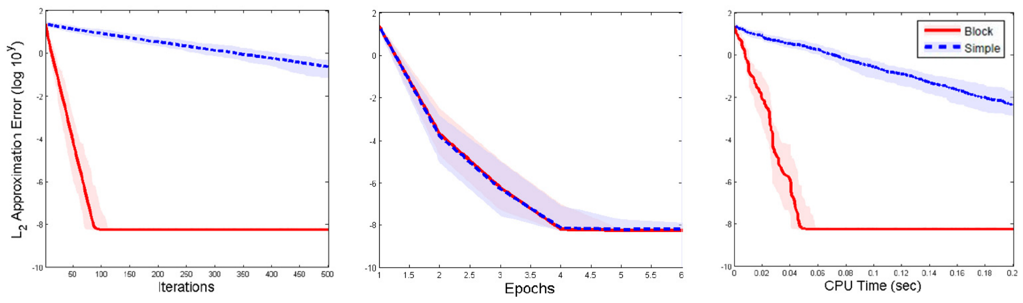

Figure 3-2 (Block Kaczmarz Method vs. Simple Kaczmarz Method). The matrix $\boldsymbol{A}$ is a 500 by 50 matrix. The approximation error $\|e(\boldsymbol{A}\boldsymbol{x}_j - \boldsymbol{b})\|$ is plotted against iterations, epochs, and CPU Time (in seconds). Convergence for the simple Kaczmarz method is displayed in blue and the block Kaczmarz method in red. The lines represent the median of 100 trials, with the shaded region containing the area between the minimum and maximum value of the 100 trials.**[Left]** Approximation error as a function of the number of iterations. **[Center]** Approximation error as a function of epochs. **[Right]** Approximation error as a function of CPU time.

Figure 3-2 [middle panel] displays the convergence of the two methods per epoch. The block Kaczmarz algorithm has a epoch of $m + n_{in}$, which is the minimum number of iterations required to enforce each equality and inequality. The epoch for the simple Kaczmarz method is again n in the inequalities case. In this paper we only count an iteration towards an epoch if the estimated solution $\boldsymbol{x}_j \neq \boldsymbol{x}_{j-1}$. Thus in the case where a chosen inequality is already satisfied for iteration j, this iteration does not count towards an epoch. The numerical experiments suggest that block Kaczmarz method is no better than the simple Kaczmarz method. This result differs from experiments run by Needell and Tropp [22], in which the block Kaczmarz method improved convergence per epoch. However, the results still indicate that blocking the equalities does not slow convergence per epoch, and actually suggests that the block Kaczmarz method may outperform the results that are predicted by theory.

Finally, Figure 3-2 [right panel] compares the rate of convergence of the two algorithms by plotting the residual against the CPU time expended in the simulation. While the block Kaczmarz approach takes more computing time to create the paving, the increased convergence rate quickly gives the method improved convergence per second relative to the simple Kaczmarz algorithm.

Chapter 4

Blocking Inequalities

We considered an adjusted block Kaczmarz strategy in which the inequalities are blocked together as well as the equalities. If a block of inequalities is selected, the algorithm proceeds by selecting each row from the block that is not already satisfied, and projecting onto those rows. Therefore unlike for a block of equalities in which every row in the block is used, the inequality block updates only using some of the rows in the given block. For consistency, the update only counts toward an iteration if any of the inequalities in the selected block are not satisfied, and will count each unsatisfied row towards an epoch. While we do not provide theoretical evidence of convergence using this approach, numerical simulations suggest that this method results in quicker convergence than the method where the block strategy is applied to only the equalities. The experiment procedure, similar to that given in Section 3, runs as follows:

For each of 100 trials,

1. Create matrix $\boldsymbol{A}_{\text{rand}}$ in the manner described in Section 3.
2. Create $\boldsymbol{b}_{in}$. The first 400 components are set as 0, and the remaining 100 take independent random uniform values from $[0, 1 \times 10^{-9}]$.
3. Create $\boldsymbol{x}_{\star}$ where each entry is selected independently from a unit normal distri-

bution. Set $\boldsymbol{b} = \boldsymbol{A}_{\text{rand}}\boldsymbol{x}_\star + \boldsymbol{b}_{in}$.

4. Pave submatrix $\boldsymbol{A}_=$ into 16 blocks with 25 equalities per block without replacement.

5. Pave submatrix $\boldsymbol{A}_\leq$ into 4 blocks with 25 inequalities per block without replacement.

6. Draw uniformly at random q from $[0,1]$.

   (a) If $q \leq \frac{n_e}{n}$, choose equality block $\{1,\ldots,16\}$ uniformly at random and update iterate $\{\boldsymbol{x}_j^{\text{block}} : j \geq 0\}$ using (1.2.6).

   (b) Else, choose inequality block uniformly at random from $\{17,\ldots,20\}$

      i. For each row i in the block: If $\boldsymbol{A}_i \cdot \boldsymbol{x}_{j-1} \leq \boldsymbol{b}_i$, remove row i from the block submatrix for this projection.

      ii. Update iterate $\{\boldsymbol{x}_j^{\text{block}} : j \geq 0\}$ using (1.2.6) for the remaining inequalities.

The MATLAB code for the block method was modified to block the inequalities as well (see Appendix). Figure 4-1 illustrates the per iteration convergence of the previous block method with the new block algorithm including blocking inequalities, and Figure 4-2 displays the per epoch convergence. The plots display the median of 100 trials for each method, with shaded regions representing the area between the minimum and maximum values of the 100 trials. The numerical experiments suggest improved convergence when the inequalities are blocked.

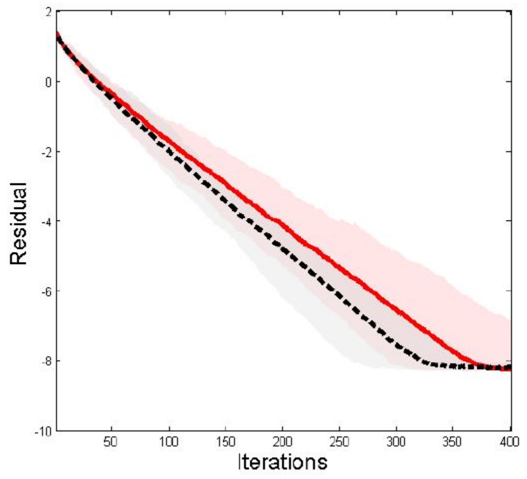

Figure 4-1 (Block Kaczmarz Method Applied to Equalities vs. Block Kaczmarz Method Applied to Equalities and Inequalities). Per iteration convergence of the two variants for matrix A with dimensions 500×100. The red line represents the block Kaczmarz algorithm applied to equalities and the simple randomized Kaczmarz algorithm applied to inequalities (Section 3). The black line shows convergence for the block Kaczmarz method applied to both the equalities and inequalities in the method described in Section 4.

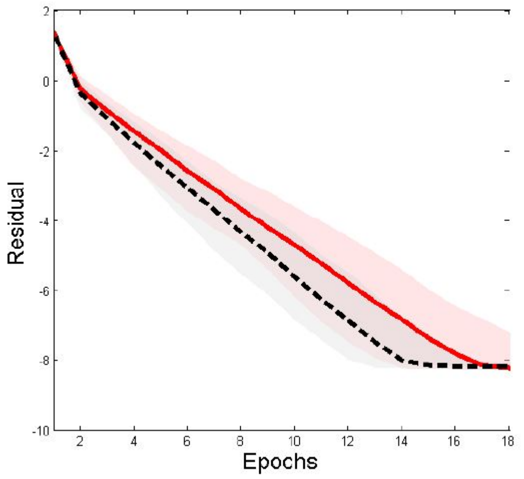

Figure 4-2 (Block Kaczmarz Method Applied to Equalities vs. Block Kaczmarz Method Applied to Equalities and Inequalities). Per epoch convergence of the two variants for matrix A with dimensions 500×100. The red line represents the block Kaczmarz algorithm applied to equalities and the simple randomized Kaczmarz algorithm applied to inequalities described in Section 3. The black line shows convergence for the block Kaczmarz method applied to both the equalities and inequalities in the method described in Section 4.

Chapter 5

Related Works

The Kaczmarz algorithm was first proposed in [16]. This method is also called "algebraic reconstruction technique" when used in computer tomography [11, 4, 19, 13]. Kaczmarz [16] demonstrated that the method converged to the solution of linear system $\boldsymbol{Ax} = \boldsymbol{b}$ for square, non-singular matrix $\boldsymbol{A}$. Many have analyzed convergence results of the Kaczmarz method [5, 6, 31, 10].Empirical results suggested that randomized selection improved convergence over the cyclic scheme [12, 14, 9, 19]. Strohmer and Vershynin [28] were the first to prove an expected linear convergence rate using a randomized Kaczmarz algorithm. This result was extended by Needell [20] to apply to an inconsistent system. Needell found a linear convergence rate to a fixed radius around the least-squares solution. Zouzias and Freris [32] also extend [28] in the inconsistent case, using a variant of a Kaczmarz method described in [24] to reduce the residual and thus prove convergence to the least-squares solution, unlike the standard methods. Needell, Srebro, and Ward [21] use a connection between stochastic gradient descent and the randomized Kaczmarz algorithm to present a tradeoff between the convergence rate and the radius of convergence (the residual described in [20]). Liu, Wright, and Sridhar [18] discuss applying a parallelized variant of the randomized Kaczmarz method, demonstrating that the linear convergence rate can be increased almost linearly by bounding the number of processors by a multiple of the number of rows of $\boldsymbol{A}$.

The Block Kaczmarz updating method was introduced by Elfving [8], and focused to the case used in this paper by Eggermont et al. [7]. Xu and Zikatanov [31] give estimates of the convergence rates for algorithms including the block Kaczmarz approach. Needell and Tropp [22] extend this result, giving an expected linear convergence rate which depends on the properties of matrix $\boldsymbol{A}$ and of the submatrices $\boldsymbol{A_\tau}$ resulting from the paving, connecting pavings and the block Kaczmarz scheme. Popa's paving literature used blocks with orthogonal rows that give results favorable to the block Kaczmarz method [24, 25, 26]. Needell, Zhao, and Zouzias [23] expand on the results from [22] and [31] to demonstrate convergence to the least-squares solution for an inconsistent system using the block Kaczmarz method. Again the block approach can yield faster convergence than the simple method.

The Kaczmarz method was first analyzed for a system of equalities and inequalities by Leventhal and Lewis [17]. They give a linear convergence rate to the feasible solution space S, using $\|\boldsymbol{A}\|_F^2$ and the Hoffman constant [15]. We apply the block Kaczmarz scheme to the system described in [17], combing their result with that of Needell and Tropp [22].

Chapter 6

Appendix

6.1 MATLAB Code: Kaczmarz Convergence Example 1

```
A = [2 3; 1 -2];
a1 = [2 3];
a2 = [1 -2];
b1 = 9;
b2 = 1;
x = [3; 1];
b = [9; 1];

x0 = [-1 1];
xk = zeros(100,2);


xk(1,:) = x0 + ((b1 - a1*transpose(x0))/(norm(a1)^2))*a1;
for i = 2:100
    if rem(i,2) == 0
        xk(i,:) = xk(i-1,:) + ((b2 - a2*transpose(xk(i-1,:)))/(norm(a2)^2))*a2;
    else
        xk(i,:) = xk(i-1,:) + ((b1 - a1*transpose(xk(i-1,:)))/(norm(a1)^2))*a1;
    end

end
xk


x = linspace(-2,6,1000);
y = 3 - x*2/3;
z = .5*x - .5;
```

```

plot(x,y, 'black')
hold all
plot(x,z,'blue')

x = linspace(-1,xk(1,1),100);
y = ((xk(1,2)-1)/(xk(1,1)+1))*(x+1)+1;
hold all
plot(x,y,'red')

for j = 2:10
    x = linspace(xk(j-1,1),xk(j,1),100);
    y = ((xk(j,2)-xk(j-1,2))/(xk(j,1)-xk(j-1,1)))*(x-xk(j-1,1))+xk(j-1,2);
    hold all
    plot(x,y,'red')
end
```

6.2 MATLAB Code: Kaczmarz Convergence Example 2

```
A = [2 3; 4 5; -6 1; 1 -2; 1 -5];
a1 = [2 3];
a2 = [4 5];
a3 = [-6 1];
a4 = [1 -2];
a5 = [1 -5];
b1 = 9;
b2 = 17;
b3 = -17;
b4 = 1;
b5 = -2;
x = [3; 1];
b = [9; 17; -17; 1; -2];

x0 = [-1 1];
xk = zeros(100,2);


xk(1,:) = x0 + ((b1 - a1*transpose(x0))/(norm(a1)^2))*a1;
for i = 2:100
    %For cyclic selection:
    j = i;
    j = randi([1,5],1,1);
    %For random selection:
    if rem(j,5) == 2
        xk(i,:) = xk(i-1,:) + ((b2 - a2*transpose(xk(i-1,:)))/(norm(a2)^2))*a2;
    end
    if rem(j,5) == 3
        xk(i,:) = xk(i-1,:) + ((b3 - a3*transpose(xk(i-1,:)))/(norm(a3)^2))*a3;
    end
    if rem(j,5) == 4
```

```
        xk(i,:) = xk(i-1,:) + ((b4 - a4*transpose(xk(i-1,:)))/(norm(a4)^2))*a4;
    end
    if rem(j,5) == 0
        xk(i,:) = xk(i-1,:) + ((b5 - a5*transpose(xk(i-1,:)))/(norm(a5)^2))*a5;
    end
    if rem(j,5) == 1
        xk(i,:) = xk(i-1,:) + ((b1 - a1*transpose(xk(i-1,:)))/(norm(a1)^2))*a1;
    end


end

x = linspace(-2,6,1000);
y = 3 - x*2/3;
z = .5*x - .5;
w = -.8*x + 17/5;
q = 6*x -17;
r = .2*x + .4;

plot(x,y, 'black')
hold all
plot(x,z,'black')
hold all
plot(x,w,'black')
hold all
plot(x,q,'black')
hold all
plot(x,r,'black')

x = linspace(-1,xk(1,1),100);
y = ((xk(1,2)-1)/(xk(1,1)+1))*(x+1)+1;
hold all
plot(x,y,'red')

for j = 2:15
    x = linspace(xk(j-1,1),xk(j,1),100);
    y = ((xk(j,2)-xk(j-1,2))/(xk(j,1)-xk(j-1,1)))*(x-xk(j-1,1))+xk(j-1,2);
    hold all
    plot(x,y,'red')
end
```

6.3 MATLAB Code: Simple Kaczmarz

```
function [resid, cpu] = Simple (A, b, x, n, d, its, epoch)

x0 = zeros(d,1);
xk = x0;
% eq = # of equalities. n Ű eq = # of inequalities
eq = 400;
ineq = n - eq;
```

```
%Number of equalities in a block
blocksize = 25;
% m is size of paving (number of blocks)
m = eq/blocksize;

%probability of choosing equality
se = eq/n;
j = 1;
time = cputime;

resid(1) = err(A,xk,b,eq,m);
cpu(1) = cputime-time;

while j <= its*epoch
    %Choose equality or inequality
    choice = rand;
    %If select equality
    if choice<se
        row = randi([1,eq]);
        ai = A(row,:);
        bi = b(row,:);
        xk = xk + ((bi - ai*xk)*(ai)/(norm(ai)^2))';
    %If select inequality
    else
        row = floor(rand*(n-eq))+1+eq;
        ai = A(row,:);
        bi =b(row,:);
        beta = -(bi - ai*xk);
        if beta < 0
            beta = 0;
            %Only count iteration if inequality not satisfied
            if j > 0
                j = j-1;
            end
        end
        xk = xk + ((beta)*(ai)/(norm (ai)^2))';
    end
    j = j + 1;
    if rem(j, epoch) == 0
        resid(j/epoch) = err(A,xk,b,eq,m);
        cpu(j/epoch) = cputime-time;
    end
end
```

6.4 MATLAB Code: Simple Kaczmarz for Inequalities, Block Kaczmarz for Equalities

```
function [resid, cpu] = Block (A, b, x, n, d, its, epoch)
% eq = # of equalities. n Ű eq = # of inequalities
```

```
eq = 400;
ineq = n - eq;
%Number of equalities in a block
blocksize = 25;
% m is size of paving (number of blocks)
m = eq/blocksize;

%Block just equalities:
x0 = zeros(d,1);
xk = x0;
%empty matrix for paving
t = zeros(m, blocksize);
%keep track of rows used
used = zeros(eq, 1);


%First step: pave
%Make t an m by eq/m matrix with each row k represents paving k
for h = 1:m
    for j = 1:blocksize
        i = floor(rand*eq)+1;
        while any(used == i)
            i = floor(rand*eq)+1;
        end
        t(h,j) = i;
        used((h-1)*m+j) = i;
    end
end

%disp(t);

At = zeros(eq/m, d, m);
Attemp = zeros(eq/m, d);
bttemp = zeros(eq/m, 1);
bt = zeros(eq/m, 1, m);

for q = 1:m
    for r = 1:eq/m
        Attemp(r,:) = A(t(q,r),:);
        bttemp(r,:) = b(t(q,r),:);
    end
    At(:,:,q) = Attemp;
    bt(:,:,q) = bttemp;
end

%probability of choosing equality
se = eq/n;
%counts rows touched
tot = 1;
values = 1;
resid(values) = err(A,xk,b,eq,m);
time = cputime;
```

```
cpu(values) = cputime - time;
values = values + 1;


while tot <= its*epoch
    %Choose equality or inequality
    choice = rand;
    %If select equality
    if choice<se
        part = randi([1,m]);
        Atd = At(:,:,part)'*(At(:,:,part)*At(:,:,part)')^(-1);
        xk = xk + ((Atd)*(bt(:,:,part)-At(:,:,part)*xk));
        %Increase counter depending if by iteration or epoch
        if epoch == 1
            tot = tot + 1;
        else
            tot = tot + blocksize;
        end
    %If select inequality
    else
        row = floor(rand*ineq)+1+eq;
        ai = A(row,:);
        bi =b(row,:);
        beta = -(bi - ai*xk);
        if beta < 0
            beta = 0;
            tot = tot-1;
        end
        xk = xk + ((beta)*(ai)/(norm (ai)^2))';
        tot = tot + 1;
    end
    if tot/epoch >= values
        resid(values) = err(A,xk,b,eq,m);
        cpu(values) = cputime - time;
        values = values + 1;
    end
end
```

6.5 MATLAB Code: Block Kaczmarz for Inequalities and Equalities

```
function [resid, cpu] = AllBlock (A, b, x, n, d, its, epoch)
% eq = # of equalities. n Ű eq = # of inequalities
eq = 400;
ineq = n - eq;
%Number of equalities in a block
blocksize = 25;
%Number of equality blocks
m = eq/blocksize;
```

```
%Number of inequalities in a block
blocksizei = 25;
%Number of inequality blocks
mi = ineq/blocksizei;
x0 = zeros(d,1);
xk = x0;

%empty matrix for paving
t = zeros(m, blocksize);
%keep track of rows used
used = zeros(eq, 1);
ti = zeros(mi,blocksize);
usedi = zeros(ineq,1);

%Pave equalities
for h = 1:m
    for j = 1:blocksize
        i = floor(rand*eq)+1;
        while any(used == i)
            i = floor(rand*eq)+1;
        end
        t(h,j) = i;
        used((h-1)*m+j) = i;
    end
end

%Pave inequalities
for h = 1:mi
    for j = 1:blocksizei
        i = floor(rand*ineq)+eq+1;
        while any(usedi == i)
            i = floor(rand*ineq)+eq+1;
        end
        ti(h,j) = i;
        usedi((h-1)*mi+j) = i;
    end
end

%Get paving matrices
At = zeros(eq/m, d, m);
Attemp = zeros(eq/m, d);
bttemp = zeros(eq/m, 1);
bt = zeros(eq/m, 1, m);

for q = 1:m
    for r = 1:eq/m
        Attemp(r,:) = A(t(q,r),:);
        bttemp(r,:) = b(t(q,r),:);
    end
    At(:,:,q) = Attemp;
    bt(:,:,q) = bttemp;
end
```

```
for q = 1:mi
    for r = 1:ineq/mi
        Attempi(r,:) = A(ti(q,r),:);
        bttempi(r,:) = b(ti(q,r),:);
    end
    Ati(:,:,q) = Attempi;
    bti(:,:,q) = bttempi;
end

%probability of choosing equality
se = eq / n;
%counts rows seen
tot = 1;
values = 1;
resid(values) = err(A,xk,b,eq,m);
time = cputime;
cpu(values) = cputime - time;
values = values + 1;

while tot <= its*epoch
    %choose equality or inequality
    choice = rand;
    %If select equality
    if choice<se
        part = randi([1,m]);
        Atd = At(:,:,part)'*(At(:,:,part)*At(:,:,part)')^(-1);
        xk = xk + ((Atd)*(bt(:,:,part)-At(:,:,part)*xk));
        if epoch == 1
            tot = tot + 1;
        else
            tot = tot + blocksize;
        end
    %If select inequality
    else
        part = randi([1,mi]);
        Atik = Ati(:,:,part);
        btik = bti(:,:,part);
        passed = 1;
        Attemp = zeros(passed,d);
        bttemp = zeros(passed,1);
        for v = 1:ineq/mi
            %check to see if inequality satisfied- if not, add it to
            %temporary matrices
            if Atik(v,:)*xk < bti(v)
                Attemp(passed,:) = Atik(v,:);
                bttemp(passed) = btik(v);
                passed = passed + 1;
                %only if by per epoch, not per iteration, update by number
                %of inequalities not satisfied
                if epoch > 1
                    tot = tot + 1;
                end

            end
        end
        %if not all inequalities already satisfied, update xk
        if passed > 1
            Attemp = Attemp(1:passed-1,:);
            bttemp = bttemp(1:passed-1);
            Atdi = Attemp'*(Attemp*Attemp')^(-1);
            xk = xk + ((Atdi)*(transpose(bttemp)-Attemp*xk));
        end
        %in iteration case, update by 1 if any inequality not satisfied
        if epoch == 1
            if passed > 1
                tot = tot + 1;
            end
        end
    end
    if tot/epoch >= values
        resid(values) = err(A,xk,b,eq,m);
        cpu(values) = cputime - time;
        values = values + 1;
    end

end
```

6.6 MATLAB Code: Residual Calculation

```
function residual = err(A, xk, b, eq, m)
    resid = A*xk - b;
    for i = eq+1:m
        if resid(i) < 0
            resid(i) = 0;
        end
    end
    residual = norm(resid);
end
```

References

[1] Åke Björck. *Numerical methods for least squares problems.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1996.

[2] J. Bourgain and L. Tzafriri. Invertibility of "large" submatrices with applications to the geometry of Banach spaces and harmonic analysis. *Israel J. Math.*, 57(2):137–224, 1987.

[3] J. Bourgain and L. Tzafriri. On a problem of Kadison and Singer. *J. Reine Angew. Math.*, 420:1–43, 1991.

[4] Charles L. Byrne. *Applied iterative methods.* A K Peters Ltd., Wellesley, MA, 2008.

[5] Frank Deutsch. Rate of convergence of the method of alternating projections. In *Parametric optimization and approximation*, pages 96–107. Springer, 1985.

[6] Frank Deutsch and Hein Hundal. The rate of convergence for the method of alternating projections, ii. *Journal of Mathematical Analysis and Applications*, 205(2):381–405, 1997.

[7] P. P. B. Eggermont, G. T. Herman, and A. Lent. Iterative algorithms for large partitioned linear systems, with applications to image reconstruction. *Linear Algebra Appl.*, 40:37–67, 1981.

[8] Tommy Elfving. Block-iterative methods for consistent and inconsistent linear equations. *Numer. Math.*, 35(1):1–12, 1980.

[9] Hans G Feichtinger, C Cenker, M Mayer, H Steier, and Thomas Strohmer. New variants of the pocs method using affine subspaces of finite codimension with applications to irregular sampling. In *Applications in Optical Science and Engineering*, pages 299–310. International Society for Optics and Photonics, 1992.

[10] A Galántai. On the rate of convergence of the alternating projection method in finite dimensional spaces. *Journal of mathematical analysis and applications*, 310(1):30–44, 2005.

[11] R. Gordon, R. Bender, and G. T. Herman. Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography. *J. Theoret. Biol.*, 29:471–481, 1970.

[12] C. Hamaker and D. C. Solmon. The angles between the null spaces of X-rays. *J. Math. Anal. Appl.*, 62(1):1–23, 1978.

[13] Gabor T Herman. *Fundamentals of computerized tomography: image reconstruction from projections.* Springer, 2009.

[14] G.T. Herman and L.B. Meyer. Algebraic reconstruction techniques can be made computationally efficient. *IEEE Trans. Medical Imaging*, 12(3):600–609, 1993.

[15] Alan J. Hoffman. On approximate solutions of systems of linear inequalities. *J. Research Nat. Bur. Standards*, 49:263–265, 1952.

[16] S. Kaczmarz. Angenäherte auflösung von systemen linearer gleichungen. *Bull. Int. Acad. Polon. Sci. Lett. Ser. A*, pages 335–357, 1937.

[17] D. Leventhal and A. S. Lewis. Randomized methods for linear constraints: convergence rates and conditioning. *Math. Oper. Res.*, 35(3):641–654, 2010.

[18] Ji Liu, Stephen J Wright, and Sridhar Srikrishna. An asynchronous parallel randomized kaczmarz algorithm. Available at `arXiv:1401.4780`, January 2014.

[19] F. Natterer. *The mathematics of computerized tomography*, volume 32 of *Classics in Applied Mathematics.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2001. Reprint of the 1986 original.

[20] Deanna Needell. Randomized Kaczmarz solver for noisy linear systems. *BIT*, 50(2):395–403, 2010.

[21] Deanna Needell, Nathan Srebro, and Rachel Ward. Stochastic gradient descent and the randomized kaczmarz algorithm. *arXiv preprint arXiv:1310.5715*, 2013.

[22] Deanna Needell and Joel A Tropp. Paved with good intentions: Analysis of a randomized block kaczmarz method. *Linear Algebra and its Applications*, 441:199–221, 2014.

[23] Deanna Needell, Ran Zhao, and Anastasios Zouzias. Randomized block kaczmarz method with projection for solving least squares. *arXiv preprint arXiv:1403.4192*, 2014.

[24] Constantin Popa. Block-projections algorithms with blocks containing mutually orthogonal rows and columns. *BIT*, 39(2):323–338, 1999.

[25] Constantin Popa. A fast Kaczmarz-Kovarik algorithm for consistent least-squares problems. *Korean J. Comput. Appl. Math.*, 8(1):9–26, 2001.

[26] Constantin Popa. A Kaczmarz-Kovarik algorithm for symmetric ill-conditioned matrices. *An. Ştiinţ. Univ. Ovidius Constanţa Ser. Mat.*, 12(2):135–146, 2004.

[27] M Ibrahim Sezan and Henry Stark. Applications of convex projection theory to image recovery in tomography and related areas. *Image Recovery: Theory and Application*, pages 155–270, 1987.

[28] Thomas Strohmer and Roman Vershynin. A randomized Kaczmarz algorithm with exponential convergence. *J. Fourier Anal. Appl.*, 15(2):262–278, 2009.

[29] Joel A. Tropp. Column subset selection, matrix factorization, and eigenvalue optimization. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 978–986, Philadelphia, PA, 2009. SIAM.

[30] R. Vershynin. John's decompositions: selecting a large part. *Israel J. Math.*, 122:253–277, 2001.

[31] Jinchao Xu and Ludmil Zikatanov. The method of alternating projections and the method of subspace corrections in Hilbert space. *J. Amer. Math. Soc.*, 15(3):573–597, 2002.

[32] Anastasios Zouzias and Nikolaos M Freris. Randomized extended kaczmarz for solving least squares. *SIAM Journal on Matrix Analysis and Applications*, 34(2):773–793, 2013.