Computational Sciences and Engineering 4(2) (2024) 331-349

University of Guilan

Computational Sciences and Engineering

journal homepage: https://cse.guilan.ac.ir/

Acceleration of randomized Kaczmarz method via the Johnson–Lindenstrauss Lemma

Somayeh Aghaei Khomami [a,*]

[a] Rasht Metropolitan Municipality, Iran

ARTICLE INFO

Article history:
Received 4 August 2025
Received in revised form 4 October 2025
Accepted 4 October 2025
Available online 5 December 2025

Keywords:
Randomized Kaczmarz Method,
Dimensionality Reduction,
Johnson–Lindenstrauss Lemma,
Monte Carlo Technique,
Iterative Algorithms

ABSTRACT

In this paper, we propose an accelerated variant of the randomized Kaczmarz method for solving large-scale linear systems, including both standard and inequality-constrained systems. The key innovation lies in integrating the Johnson–Lindenstrauss (JL) lemma into the row-selection process, which allows high-dimensional rows to be projected onto lower-dimensional spaces while approximately preserving pairwise distances. This enables near-optimal row selection with reduced computational cost, improving both convergence rate and stability, particularly for ill-conditioned systems. Furthermore, Monte Carlo techniques are employed to efficiently construct the projection matrices, enhancing the overall computational performance. Numerical experiments demonstrate that the proposed method achieves faster convergence and higher accuracy compared to traditional randomized Kaczmarz and other conventional techniques, making it highly suitable for large-scale problems in applied mathematics and engineering.

1. Introduction

Solving systems of linear equations is of fundamental importance across various scientific and engineering disciplines. Several methods have been developed to address these systems. For instance, the Monte Carlo approach rewrites a system $Ax=b$ as $x=Cx+bx$, relying on Markov chains and requiring that the spectral radius $\rho(C)<1$ to ensure convergence. While effective in certain contexts, this method may face challenges when applied to large-scale or ill-conditioned systems.

An alternative class of methods, known as Kaczmarz methods, iteratively solves linear systems by selecting and projecting onto rows of the coefficient matrix. The convergence rate of these methods strongly depends on the row selection strategy. Traditional randomized Kaczmarz algorithms

* Corresponding author.
E-mail addresses: somayehaghaye@yahoo.com (S. Aghaei Khomami)

https://doi.org/10.22124/cse.2025.31590.1119

choose rows based on their Euclidean norms, which can be suboptimal for large or high-dimensional systems.

In this paper, we propose an accelerated variant of the randomized Kaczmarz method that integrates the Johnson–Lindenstrauss (JL) lemma into the row-selection process. By projecting high-dimensional rows onto a lower-dimensional space, the method preserves pairwise distances approximately while enabling the selection of nearly optimal rows with reduced computational cost. This modification improves both the convergence rate and the stability of the algorithm, particularly for ill-conditioned systems. Furthermore, the efficiency is enhanced using Monte Carlo techniques to construct projection matrices. Numerical experiments confirm that the proposed approach outperforms traditional randomized Kaczmarz algorithms in terms of both convergence speed and accuracy, making it suitable for large-scale computational problems in practical applications.

2. History of the Kaczmarz Method

The Kaczmarz method is an iterative algorithm for solving consistent underdetermined systems $Ax=b$ through repeated projections onto solution subspaces. The randomized variant proposed by Strohmer and Vershynin (to be discussed later) achieves exponential convergence in expectation, outperforming gradient methods for large underdetermined systems.

Originally discovered by mathematician Stefan Kaczmarz, the method was later revitalized in the 1970s by Richard Gordon, Robert Bender, and Gabor Herman in the context of image reconstruction, where it was termed the Algebraic Reconstruction Technique (ART)**.** This technique incorporates positivity constraints that render it nonlinear. While applicable to any linear system of equations, its computational efficiency depends critically on the sparsity and structure of the system. In biomedical imaging applications, it has been shown to outperform alternatives such as filtered back-projection, with wide-ranging applications from computed tomography to signal processing [1, 3].

3. The Kaczmarz Method

The Kaczmarz method is a popular iterative algorithm for solving consistent underdetermined linear systems $Ax=b$. Due to its speed and simplicity, it is widely used in applications ranging from tomography to digital signal processing. The method employs a sequence of cyclic projections to iteratively converge to the solution, making it computationally feasible even for very large systems.

Given an initial estimate x0 and denoting the rows of an m×n matrix A as ai (i=1,…,m), each iteration orthogonally projects the current estimate onto the hyperplane defined by $\langle a_i,x\rangle=bi$, selected cyclically. The algorithm is expressed by the recurrence relation:

$$x_{k+1} = x_k + \frac{b_i - (a_i, x_k)}{\|a_i\|^2} a_i,$$

where x_k is the k-th iterate, b_i is the i-th coordinate of b, and $i=k \bmod m+1$.

Although long used in practice, theoretical convergence guarantees were historically elusive. Crucially, the convergence rate depends on the row selection strategy in A, where poorly chosen rows can decelerate convergence. To address this, rows may be selected randomly. This randomized

variant significantly accelerates convergence, with recent theoretical results formalizing its efficacy [3].

We next present the randomized Kaczmarz method and its accelerated version, which leverages the Johnson-Lindenstrauss Lemma (JL) to enhance convergence rates [1].

4. Johnson-Lindenstrauss Lemma

For any set of nn points in ℓ_2^{d}, there exists a distribution over mappings such that a randomly chosen JL embedding projects the data onto $\ell_2^{\mathrm{o(log\,n)}}$with high probability, approximately preserving distances. Simply put, JL embeddings enable dimensionality reduction of high-dimensional vectors. This lemma is pivotal in metric space embedding theory.

Consider points projected first onto a plane, then onto a line: the result illustrates the power of dimensionality reduction. Formally, given a set of high-dimensional points, we ask whether they can be embedded into a lower-dimensional space with minimal distortion. For n points in R^d (represented as an $n \times d$ matrix A, one common embedding is via the singular value decomposition (SVD**)** of A: to embed into $\mathrm{R^k}$, project onto the k-dimensional space spanned by the singular vectors corresponding to A's largest singular values [17].

For given $\varepsilon > 0$ and integer n, let k be a positive integer satisfying $k \geq k_0 = O(\varepsilon^{-2} \log n)$. For any set P of n points in $\mathbb{R}^{\mathbf{d}}$, there exists a mapping f: $\mathbb{R}^{\mathbf{d}} \rightarrow \mathbb{R}^{\mathbf{k}}$ such that for all $u, v \in P$**:**

$$(1-\varepsilon)\|u-v\|^2 \leq \|f(u)-f(v)\|^2 \leq (1+\varepsilon)\|u-v\|^2 . \quad \textbf{(1)}$$

In recent years, such embeddings have accelerated solutions for diverse problems. The JL method reduces dimensionality, dramatically speeding up algorithms—particularly those with runtime exponentially dependent on ambient space dimension. The pairwise distance

guarantee often enables proofs that solutions in the low-dimensional space well-approximate those in the original space.

Theorem 1: Let P be an arbitrary point set in $\mathbb{R}^{\mathbf{d}}$, represented as an n × d matrix A. For any ε, β > 0, define $\boldsymbol{k_0 = \frac{4+2\beta}{\frac{\varepsilon^2}{2}-\frac{\varepsilon^3}{3}} log\, n}$.

For integer $k \geq k_0$**,** let R be a $d \times k$ random matrix with $R(i,j) = r_{ij}$**,** where **{**r_{ij}**}** are *i.i.d.* random variables generated as:

$$r_{ij} = \begin{cases} +1 & P = 1/2 \\ -1 & P = 1/2 \end{cases} \quad \textbf{(2)}$$

$$r_{ij} = \sqrt{3} \times \begin{cases} +1 & P = \frac{1}{2} \\ 0 & P = \frac{2}{3} \\ -1 & P = \frac{1}{2} \end{cases} \quad \textbf{(3)}$$

Then $E = \frac{1}{\sqrt{k}} AR$, and f**:** $\mathbb{R}^{\mathbf{d}} \rightarrow \mathbb{R}^{\mathbf{k}}$ maps the *i*-th row of *A* to the **i**-th row of *E*. With probability at least 1 - $n^{-\beta}$**,** for all u, ν ∈ *P***:**

$$(1-\varepsilon)\|u-v\|^2 \leq \|f(u)-f(v)\|^2 \leq (1+\varepsilon)\|u-v\|^2 .$$

We observe that Theorem 1 requires only simple probability distributions to construct the projection matrix, while the embedding computation itself reduces dimensionality. Though distribution (2) appears challenging, distribution (3) triples processing speed by requiring only one-third of all features per coordinate [4].

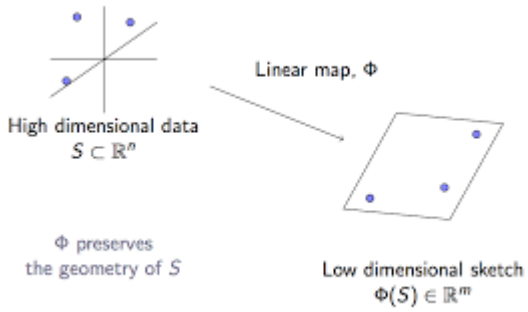

Figure 1. Point projection and dimensionality reduction

Definition 1. Let u be an arbitrary vector in R^d, and R be a $d\times\ k$ random matrix where each entry $R_{ij} \sim\ N(0,1)$ is i.i.d. We define the projection:

$g = \frac{1}{\sqrt{k}} uR$. Thus, $g \epsilon R^k$ with components: $g_i = \frac{1}{\sqrt{k}} \sum_j u_j R_{ji}$

Definition 2. *Let* $X = \frac{\sqrt{k}}{\|u\|} g$. Then for *every* $i = 1, \dots, k$:

$$If\, g = \frac{1}{\sqrt{k}} uR\,, then\ \sqrt{k} g = uR\,, which\ implies\ X = \frac{\sqrt{k}}{\|u\|}\ g = \frac{uR}{\|u\|}\,, so\ finally\ x_i = \frac{1}{\|u\|}\ u.R_i\,.$$

$$g_i = \frac{1}{\sqrt{k}} \sum_j u_j R_{ji}$$

$$x = \sum_{i=1}^{k} x_i^2 = \|X\|_2^2 = k\ \frac{\|g\|_2^2}{\|u\|_2^2}\ . \qquad \textbf{(4)}$$

Remark 1: Under above assumptions we have

$$g_i \sim N\left(0, \frac{\|u\|_2^2}{k}\right) \quad i.i.d \quad , i = 1, \dots, k\,, \quad X \sim N_k(0, I).$$

Note:

$$E\left(e^{\lambda t^2}\right) = \int_{-\infty}^{\infty} e^{\lambda t^2}\ \frac{1}{\sqrt{2\,\pi}}\ e^{-\frac{t^2}{2}}\, dt = \frac{1}{\sqrt{1-2\,\lambda}} \int_{-\infty}^{\infty} \frac{\sqrt{1-2\,\lambda}}{\sqrt{2\,\pi}} e^{-\frac{t^2}{2}(1-2\,\lambda)}\, d = \frac{1}{\sqrt{1-2\,\lambda}}.$$

Lemma 1: Prove that under above assumptions we have that

$$P\,[\,\|g\|_2^2\ \leq (1+\varepsilon)\|u\|_2^2\,]\ > 1 -\ n^{-2}.$$

Proof:

$P\left[\ \|g\|_2^2\ \geq (1+\varepsilon)\|u\|_2^2\ \right]$

From Equation (4), we derive the following relationship:

$$= P\left[\frac{\|u\|_2^2\, x}{k}\ \geq (1+\varepsilon)\|u\|_2^2\right] = P\left[\ x \geq (1+\epsilon)\ k\right]$$

$$= P\left[e^{\lambda x} \geq e^{\lambda(1+\varepsilon)k}\right], \qquad \forall\ \lambda \geq 0$$

Using Markov's inequality, we have:

$$\mathrm{P}\left[\mathrm{e}^{\lambda \mathrm{x}} \geq \mathrm{e}^{\lambda(1+\varepsilon)\mathrm{k}}\right] \ \leq\ \frac{\mathrm{E}\left(\mathrm{e}^{\lambda \mathrm{x}}\right)}{\mathrm{e}^{\lambda(1+\varepsilon)\mathrm{k}}} = \prod_{\mathrm{i}=1}^{\mathrm{k}} \frac{\mathrm{E}\left(\mathrm{e}^{\lambda \mathrm{x}_\mathrm{i}^2}\right)}{\mathrm{e}^{\lambda(1+\varepsilon)\,\mathrm{k}}} \qquad \text{"The } \boldsymbol{x_i} \text{ are i.i.d."}$$

$$\leq \left[\frac{\mathrm{E}\left(\mathrm{e}^{\lambda \mathrm{x}_\mathrm{i}^2}\right)}{\mathrm{e}^{\lambda(1+\varepsilon)}}\right]^{\mathrm{k}} = \left[\frac{1}{\sqrt{1-2\,\lambda}\mathrm{e}^{\lambda(1+\varepsilon)}}\right]^{\mathrm{k}}, \qquad 0 < \lambda < \frac{1}{2}$$

We will arrange $\lambda = \frac{\varepsilon}{2(1+\varepsilon)}$

$$P\left[\ \|g\|_2^2\ \geq (1+\varepsilon)\|u\|_2^2\ \right] \leq \left[\frac{1}{\sqrt{\frac{1}{1+\varepsilon}}\ e^{\frac{\varepsilon}{2}}}\right]^k = \left[(1+\varepsilon)e^{-\varepsilon}\right]^{\frac{k}{2}}. \tag{5}$$

Using the inequality $log(1+x) < x - \frac{x^2}{2} + \frac{x^3}{3}$ we set

$$log(1+\varepsilon) < \varepsilon - \frac{\varepsilon^2}{2} + \frac{\varepsilon^3}{3} \qquad then \quad (1+\varepsilon) < e^{\varepsilon - \frac{\varepsilon^2}{2} + \frac{\varepsilon^3}{3}}. \tag{6}$$

$$(1+\varepsilon)e^{-\varepsilon} < e^{-\left(\frac{\varepsilon^2}{2} - \frac{\varepsilon^3}{3}\right)},$$

Therefore,

$$P\left[\ \|g\|_2^2\ \geq (1+\varepsilon)\|u\|_2^2\ \right] \leq\ e^{-\left(\frac{\varepsilon^2}{2} - \frac{\varepsilon^3}{3}\right)\frac{k}{2}}.$$

On the other hand

$$-\left(\frac{\varepsilon^2}{2} - \frac{\varepsilon^3}{3}\right)\frac{k}{2} = -\left(\frac{3\varepsilon^2 - 2\varepsilon^3}{12}\right)k \tag{7}$$

$$\mathrm{k} \geq \frac{24}{3\ \varepsilon^2 - 2\varepsilon^3}\log(\mathrm{n}) \quad \text{then} \quad \left(\frac{3\varepsilon^2 - 2\varepsilon^3}{12}\right) k \geq 2\ log\ n \tag{8}$$

Therefore, continuing with equation (5) and using equation (36), (7), and (8), we will have

$$e^{-\left(\frac{\varepsilon^2}{2} - \frac{\varepsilon^3}{3}\right)\frac{k}{2}}\ \leq\ e^{-2\ log\ n}\ =\ n^{-2}.$$

And as a result it was proven

$$P\left[\ \|g\|_2^2\ \leq (1+\varepsilon)\|u\|_2^2\ \right]\ > 1 -\ n^{-2}.$$

Lemma 2: $P\left[\ \|g\|_2^2\ \geq (1-\varepsilon)\|u\|_2^2\ \right]\ > 1 -\ n^{-2}.$

According to equation (3), we have the following

$$P\left[\ \|g\|_2^2\ \le (1-\varepsilon)\|u\|_2^2\ \right] =$$

$$= P\left[\frac{\|u\|_2^2\, x}{k}\ \le (1-\varepsilon)\|u\|_2^2\right] = P\left[\ x \le (1-\epsilon)\ k\right]$$

$$= P\left[e^{-\lambda x} \ge e^{-\lambda(1-\varepsilon)k}\right], \qquad \forall\, \lambda \ge 0$$

$$\mathrm{P}\left[\mathrm{e}^{-\lambda \mathrm{x}} \ge \mathrm{e}^{-\lambda(1-\varepsilon)\mathrm{k}}\right] \le \frac{\mathrm{E}\left(\mathrm{e}^{-\lambda \mathrm{x}}\right)}{\mathrm{e}^{-\lambda(1-\varepsilon)\mathrm{k}}} = \prod_{\mathrm{i}=1}^{\mathrm{k}} \frac{\mathrm{E}\left(\mathrm{e}^{-\lambda \mathrm{x}_\mathrm{i}^2}\right)}{\mathrm{e}^{-\lambda(1-\varepsilon)\mathrm{k}}}$$ The random variables x_i are i. i. d

$$\le \left[\frac{1}{\sqrt{1+2\,\lambda}e^{-\lambda(1-\varepsilon)}}\right]^k. \qquad 0 < \lambda < \frac{1}{2}$$

We will arrange $\lambda = \frac{\varepsilon}{2(1-\varepsilon)}$

$$= [(1-\varepsilon)e^{\varepsilon}]^{\frac{k}{2}}\ .$$

According to inequality $log(1-x) < -x - \frac{x^2}{2}$ We will arrange

$$\le e^{-\frac{\varepsilon}{2}\frac{k}{2}}\ \le\ e^{-\left(\frac{\varepsilon^2}{2}-\frac{\varepsilon^3}{3}\right)\frac{k}{2}} \le\ n^{-2},$$

And finally, $P\left[\ \|g\|_2^2\ \ge (1-\varepsilon)\|u\|_2^2\ \right] > 1 -\ n^{-2}$.

So, combining the above results, we have

$$P(\ \|g\|_2^2\ \notin [\ (1-\varepsilon)\|u\|_2^2\ , (1+\varepsilon)\|u\|_2^2\ ]\ ) \le \frac{2}{n^2}\ .$$

Let *u* be an arbitrary vector in $\mathbb{R}^d$. The following probability holds for $u = x_i - x_j$, where x_i and x_j are vectors in the set *A*, and the function **f** is defined by a random matrix. In the following, we consider a matrix and apply the Johnson–Lindenstrauss dimensionality reduction method using MATLAB software.

Assume matrix *A* has dimensions *10×10*. Using the Johnson–Lindenstrauss lemma and Theorem 1, we reduce the dimensionality of this matrix. We set the value of *k* to **2**, and generate the random projection matrix *R* using Equation (3).

$$A = \begin{bmatrix} 1 & 5 & 7 & 6 & 4 & 0 & 2 & 6 & 4 & 2 & 0 & 2 & 3 & 4 & 1 \\ 10 & 9 & 7 & 8 & 3 & 0 & 1 & 3 & 2 & 14 & 2 & 1 & 1 & 3 & 2 \\ 0 & 7 & 9 & 5 & 6 & 7 & 8 & 9 & 1 & 3 & 9 & 1 & 0 & 0 & 1 \\ 0 & 1 & 4 & 7 & 3 & 11 & 0 & 7 & 8 & 9 & 6 & 7 & 0 & 3 & 7 \\ 3 & 1 & 0 & 2 & 0 & 7 & 7 & 3 & 3 & 9 & 1 & 0 & 8 & 9 & 2 \\ 7 & 8 & 0 & 0 & 1 & 10 & 3 & 7 & 9 & 8 & 7 & 1 & 0 & 2 & 3 \\ 0 & 0 & 1 & 1 & 5 & 7 & 5 & 4 & 6 & 2 & 5 & 6 & 1 & 0 & 9 \\ 3 & 1 & 2 & 7 & 0 & 2 & 4 & 5 & 1 & 0 & 3 & 1 & 2 & 4 & 5 \\ 9 & 0 & 1 & 0 & 3 & 1 & 1 & 0 & 2 & 3 & 2 & 4 & 1 & 2 & 3 \\ 1 & 2 & 5 & 4 & 0 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \end{bmatrix}$$

As previously stated, the goal was to reduce the dimensionality of matrix A, which was achieved using Lemma 1. The dimensionality of the matrix was reduced from 15 to 11, which contributes to faster computational performance. Moreover, the norm of the original matrix A decreased from 73 to 58.49 after the transformation.

$$A' = \begin{bmatrix} 6.2 & 3.1 & 5.2 & -4.1 & -2.0 & 2.0 & 4.1 & 5.2 & 2.6 & 2.0 & -4.7 \\ 9.4 & 0 & -2.0 & 6.7 & 6.7 & 3.6 & 8.8 & 6.2 & 7.8 & 1 & -5.7 \\ 6.7 & 1.5 & 6.7 & -12.0 & -1.5 & -1.0 & -4.1 & 9.9 & 1.5 & -1 & -6.7 \\ 6.2 & 6.2 & -0.5 & -0.5 & 1.5 & 0 & 4.7 & 1.0 & 4.7 & 3.6 & 6.7 \\ 4.7 & 4.1 & 3.6 & 4.1 & -3.6 & 5.7 & 5.2 & 1.0 & 8.8 & 11.4 & 3.1 \\ 5.7 & 2.0 & -0.5 & 1.0 & 6.7 & 0 & 0.5 & 1.0 & 4.1 & -0.5 & -3.6 \\ 4.1 & 5.7 & -3.13 & -1.0 & -2.6 & -5.2 & -1.5 & 1.5 & 1.5 & 3.1 & 3.6 \\ 2.0 & 3.13 & 0 & -3.1 & -6.2 & 5.2 & -1.5 & 4.7 & 1.0 & 4.7 & 2.0 \\ 6.2 & -2.0 & -5.7 & 6.7 & 0.5 & -2.6 & 2.6 & 2.0 & 2.0 & 1.5 & 1.5 \\ 5.7 & 3.1 & 2.0 & -3.6 & -3.6 & 1.0 & -3.1 & 3.6 & 3.6 & 1.0 & -1.0 \end{bmatrix}$$

As observed, we successfully reduced the dimensionality of the matrix using the Johnson–Lindenstrauss lemma. In the following section, we introduce another method the Kaczmarz algorithm whose convergence rate can be improved by applying this lemma.

5. Applying the Monte Carlo Method in the Johnson–Lindenstrauss Lemma

Since the Monte Carlo method is based on the generation of random numbers, and the Johnson–Lindenstrauss lemma also requires the construction of a random projection matrix for dimensionality reduction, it is therefore possible to apply Monte Carlo techniques within this lemma. In this section, we employ the Monte Carlo method to reduce the dimensionality of matrix *A*, as introduced in the previous section. We will observe that the norm of the projected matrix obtained using the Monte Carlo technique is lower than that of the projected matrix obtained without applying this technique. In fact, this approach significantly accelerates computations in large-scale problems [5].

In the following, we project matrix *A* multiple times using the Johnson–Lindenstrauss lemma. We then compute the norm of each projected matrix and compare the results.

Table 1: Image matrix software

Number	Norms of Projected Matrices Without Using the Monte Carlo Technique	Norms of Projected Matrices Using the Monte Carlo Technique
1	31/2181	15/3744
2	55/3833	22/6157
3	42/9643	13/0978
4	57/9023	13/8904
5	42/8283	15/1807
6	48/1435	18/1330
7	56/5722	25/3172

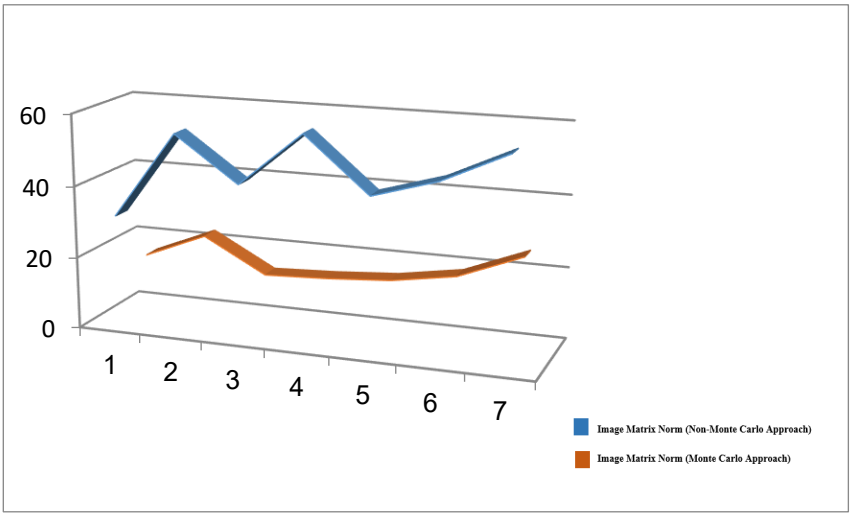

Figure 2: Comparison of image matrix software based on Monte Carlo technique

As shown in Figure 2, the Monte Carlo method demonstrates a significant reduction in the projection matrix norm. The norm decreases progressively as the number of Monte Carlo samples increases. This approach proves particularly advantageous for large-scale problems, enabling faster computational processing [1].

We first introduce the randomized Kaczmarz method, then present a modified version where each iteration selects an optimal projection from a randomly chosen subset. This modification significantly improves convergence rates in most cases. By incorporating Johnson-Lindenstrauss dimensionality reduction, we maintain the original method's runtime while only increasing preprocessing time. Our experimental studies demonstrate substantial acceleration in convergence using this enhanced approach.

5.1 Randomized Kaczmarz Method

Strohmer and Vershynin [7] proposed selecting rows of matrix A randomly with probabilities proportional to their squared Euclidean norms. The randomized Kaczmarz iteration can be expressed as:"

$$x_{k+1} = x_k + \frac{b_{p(i)} - (a_{p(i)}, x_k)}{\|a_{p(i)}\|^2} a_{p(i)} \quad , \quad k = 1,2, \ldots \tag{9}$$

where p(i) exists 1,2,…,m with probability $\frac{\|a_{p(i)}\|^2}{\|A\|^2}$ is. In general, $\|A\|_F$ Frobenius soft representation of a matrix A and $\|.\|_2$

The standard Euclidean norm and the spectral norm are used to represent the norms of vectors and matrices, respectively [1]. In general, the row selection rule used in this method is not optimal. Modifying this rule based on the row norm weights serves two main purposes. First, it guarantees exponential convergence in expectation for the Kaczmarz method. Second, it provides a computationally efficient strategy, since these row norms are often known approximately or exactly and need to be computed only once. This type of selection rule is also related to the idea of preconditioning matrix A by scaling its rows. Although diagonal preconditioners may generally perform better, finding an optimal preconditioner is itself a high-complexity optimization problem.

Based on the aforementioned row selection strategy, the following exponential bound for the expected convergence of this randomized method has been established in [7].

$E\|x_k - x\|^2 \leq \left(1 - \frac{1}{R}\right)^k \|x_0 - x\|^2,$

such that $\|A^{-1}\|^2 \|A\|_F^2$ and x_0 is an arbitrary initial estimate. We always assume that matrix A has full column rank, and its norm is defined as follows:

$$\|A^{-1}\|^2 = inf\{M: M\|Ax\|_2 \geq \|x\|_2, \quad \forall x\}$$

This bound is essentially independent of the rows of matrix A. Since each iteration involves a single projection with a time complexity of $o(n)$, This indicates that the overall method has a time complexity of $o(n^2)$, which clearly outperforms other approaches such as Gaussian elimination, which requires $o(mn^2)$ time—especially when the system is large. The theoretical and empirical advantages of this method naturally raise the question: Is it still accurate in high-precision scenarios when noise is present? Consider the system $Ax \approx b + \omega$, where ω is an arbitrary error vector added to the consistent system $Ax \approx b + \omega$. As shown in [7], in this case, we still achieve exponential convergence to the solution, but with an error factor:

$E\|x_k - x\|_2 \leq \left(1 - \frac{1}{R}\right)^{\frac{k}{2}} \|x_0\|_2 + \sqrt{R}\gamma,$

where R is the same as defined above, and $\gamma = max_i \frac{|\omega[i]|}{\|a_i\|_2}$.

- ❖ The Randomized Kaczmarz method algorithm
  - Start with the initial value x_0.
  - Compute $x_{k+1} = x_k + \frac{b - (a, x_k)}{\|a_p\|^2} a_p$, where $P(p = i) = \frac{\|a_i\|^2}{\|A\|^2}$.
  - Assume that

$R = \|A^{-1}\|^2 \|A\|_F^2 (\|A^{-1}\| = inf\{M: M\|Ax\|_2 \geq \|x\|_2 \ \ \forall x\})$,

  - where

$E\|x_k - x\|_2^2 \leq \left(1 - \frac{1}{R}\right)^k \|x_0 - x\|_2^2$.

  - A converges in $O(n)$ iterations, resulting in an overall runtime of $O(n^2)$ [15]."

5.2 The Randomized Kaczmarz Method with Noise

Assume that the original consistent system $Ax = b$ is affected by additive noise, leading to an inconsistent system of the form $Ax \approx b + z$ represents the noise vector.

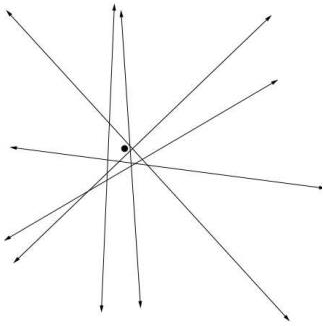

Figure 3: Image with Noise

Theorem 2:

Assume that the system $Ax = b$ is perturbed by noise, resulting in the inconsistent system $Ax \approx b + z$. Then, the expected error satisfies the following bound:

$E\|x_k - x\|_2 \leq \left(1 - \frac{1}{R}\right)^{\frac{k}{2}} \|x_0 - x\|_2 + \sqrt{R}\gamma$ Discussion In the following, we present an example to examine the convergence behavior of the Randomized Kaczmarz Method in the presence of noise.

Example 1

We solve the following system of linear equations using the Randomized Kaczmarz Method implemented in MATLAB. Then, we compare the mean of the solution vectors obtained up to the 10th iteration.

$$\begin{pmatrix} 1 & 0 & 2 & 1 \\ 3 & 1 & 4 & 2 \\ 1 & 6 & 0 & 4 \\ 2 & 2 & 5 & 3 \\ 2 & 3 & 1 & 7 \\ 5 & 2 & 3 & 1 \\ 3 & 1 & 4 & 0 \\ 8 & 1 & 9 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 5 \\ 13 \\ 21 \\ 17 \\ 23 \\ 14 \\ 9 \\ 21 \end{pmatrix}, \qquad x = \begin{pmatrix} 1 \\ 2 \\ 1 \\ 2 \end{pmatrix}$$

Table 2: Mean of the Solution Vector

Number of Iterations	Average Response per Iteration
1	0.84615
2	0.84615
3	1.72025
4	0.96435
5	1.217125
6	1.16625
7	1.648825
8	1.880975
9	1.55535
10	1.555975

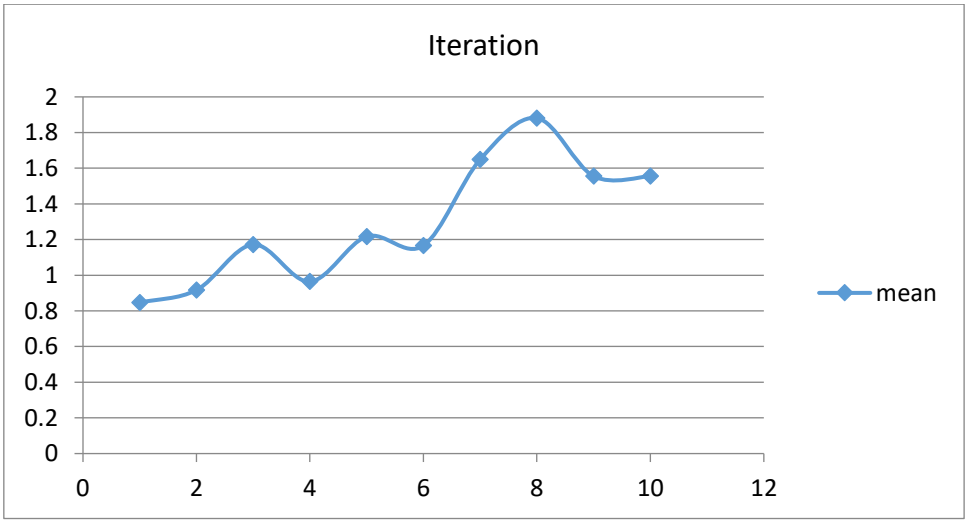

Figure 4: Convergence of the Randomized Kaczmarz Method

Figure 4 illustrates the convergence behavior of the Randomized Kaczmarz Method. As the number of iterations increases, the average of the solution vectors converges toward the true solution vector. In fact, the more iterations are performed, the closer the algorithm gets to the actual solution of the system.

It is worth noting that the use of the rand function introduces fluctuations in the solutions obtained across different runs, which will be demonstrated later. To reduce these fluctuations, one can use quasi-random sequences for generating the required random numbers. This not only smooths the convergence curve but can also accelerate the convergence rate [1].

To further analyze the previous example, we solve the same system again using the Randomized Kaczmarz Method. We then identify the first iteration in which the solution error falls below the threshold e^{-13}. After that, we compute the error at each iteration to observe the fluctuations in the solution.

These fluctuations are a result of using the rand function for random index selection, which introduces variability across different runs. This behavior is clearly visible in the error plot.

To mitigate this issue, one can use quasi-random sequences instead of purely random numbers. These sequences not only reduce the fluctuations but can also improve the convergence speed [5].

Table 3: Solution of the Linear System Using the Randomized Kaczmarz Method

Iteration Number	Final Solution			First Iteration ($\|Error\| < \varepsilon$)
100	[0.92722830849649	2.04893413316159	1.03843199475789 1.95183838096477]	-
400	[0.99947137092303	2.00052653677710	1.00045619942376 1.99961623022944]	-
700	[0.99999582437251	2.00000369573636	1.00000256831291 1.99999903900165]	-
900	[0.99999996364280	2.00000002622000	1.00000001858627 1.99999997578100]	-
1200	[0.99999999979842	2.00000000005409	1.00000000010317 1.99999999992636]	-
1500	[0.99999999999933	2.00000000000076	1.00000000000055 1.99999999999958]	-
1700	[0.99999999999996	2.00000000000008	1.00000000000002 1.99999999999997]	-
1800	[0.999999999999993	2.000000000000004	1.000000000000004 1.999999999999]	1797
1900	[0.99999999999996	2.000000000000002	1.000000000000003 1.99999999999998]	1872
2000	[0.999999999999999 2.0000000000000 1.000000000000 2.0000000000000]			1808
2100	[1.000000000000 2.000000000000 1.00000000000 2.000000000000000]			1863

Based on Table 2, we observe that at iterations 1800 and 1900, the solution approaches the true value. However, at iteration 2000, the error increases compared to the previous two iterations, indicating the presence of fluctuations.

Similarly, in Table 3, we notice that as the solution gets closer to the true value (beyond iteration 2000), the fluctuations become more pronounced. This behavior highlights the instability introduced by randomness in the selection process.

These observations suggest that using quasi-random sequences instead of purely random numbers can reduce such fluctuations and improve the stability and convergence behavior of the Randomized Kaczmarz Method.

Table 4: Error Comparison

Iteration Number	Error Norm
100	0.239197004769567
400	1.854461246280691 e-04
700	3.550709912354872 e-06
900	1.859100098818182 e-07
1200	3.248691372569895 e-10
1500	5.069066420760477 e-13
1700	4.981586918199077 e-14
1800	2.795646511717312 e-14
1900	1.158096195655681 e-14
2000	6.661338147750939 e-16
2050	1.661629672422090 e-15
2100	3.845925372767128 e-16
2150	6.280369834735101 e-16

5.3 An Important Observation on Row Selection Strategy

Another important point to consider is the method used for selecting rows randomly. In the Randomized Kaczmarz Method, there are generally two strategies for row selection:

1. Uniform Discrete Selection
2. Probability-Based Selection According to Row Norms, where the discrete probability distribution is defined as:

$$p_k = \frac{\|a_k\|^2}{\sum_{k=1}^{m}\|a_k\|^2} \ , \qquad k = 1,2,\dots,m$$

In the previous example, row selection was performed using the uniform discrete distribution. In the following example, we instead select rows at each iteration based on their norms, using the discrete probability distribution defined above.

It appears that uniform discrete selection may lead to faster convergence of the algorithm in some cases [1].

Example 2

We solve the following system of linear equations using the Randomized Kaczmarz Method. We define two strategies for row selection:

- **Method 1**: Rows are selected based on their norms, using the discrete probability distribution $p_k = \frac{\|a_k\|^2}{\sum_{k=1}^{m}\|a_k\|^2}$, $k = 1,2,\dots,m$
- **Method 2**: Rows are selected uniformly at random (uniform discrete distribution).

We then compare the error norms obtained from both methods to evaluate their convergence behavior.

$$\begin{pmatrix} 1 & 0 & 2 \\ 3 & 1 & 4 \\ 1 & 1 & 0 \\ 5 & 2 & 7 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 3 \\ 8 \\ 2 \\ 14 \\ 3 \end{pmatrix}$$

Table 5: Answers obtained by the random Kekzmarz method

Number of repetitions	Device answer
1000	[0.665021807258867 1.280904466430951 1.159011718691966]
10000	[0.999824192380030 1.00109177649869 1.00094383257159]
20000	[0.999999959283278 1.00000019885184 1.00000032193660]
30000	[0.999999999992755 1.000000000005922 1.000000000003483]
40000	[0.999999999999990 1.000000000000008 1.000000000000005]
50000	[0.999999999999999 1.000000000000001 1.000000000000000]
60000	[0.999999999999999 1.000000000000001 1.000000000000000]

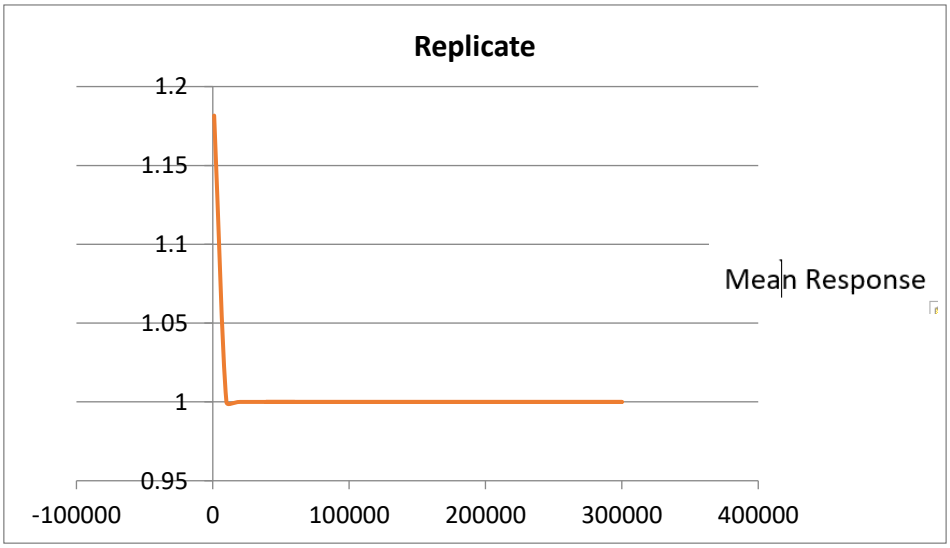

Figure 5: Comparison of average response vectors

Table 6: Error of the Kekzmarz method based on the type of row selection

Repetition	Error vector smoothing method 1	Error vector smoothing method 2
1000	0.534676082388941	0.005999195844708
10000	2.361236361225260 e-04	9.678699938266253 e-16
20000	6.149667172092788 e-04	8.950904182623619 e-16
30000	1.934703379346519 e-11	7.447602459741819 e-16
40000	2.664535259100376 e-15	6.661338147750939 e-16
50000	9.678699938266253 e-16	7.447602459741819 e-16

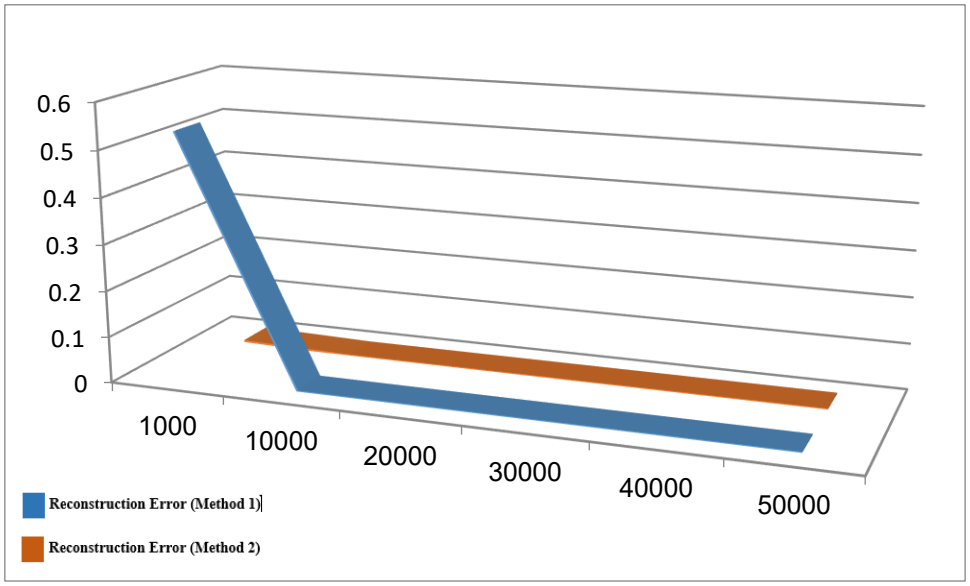

***Figure 6*:** Comparison of the error of the Kekzmarz method based on the type of row selection

As can be seen in Table 5 and Figure 6, using the discrete uniform method helps us to reach the real solution with fewer iterations. More precisely, according to Table 5, in Method Two, we almost reach the original solution in iterations 20,000 to 30,000, while in Method One, the real solution is obtained in iterations 40,000 to 50,000.

6. Modified method

In order to improve the convergence rate of the Kecsmars method, we propose a different method for selecting the rows of *A*. Although our idea should be applicable to the case where there is noise and the system is inconsistent, in this work we only consider the case without noise. Since the images in Algorithm (4-1) are vertical, we can see that the optimal image at the kth iteration is the one that maximizes $\|x_{k+1}-x_k\|^2$. By defining the recurrence relation (4-1), we can calculate these values by computing the inner product between the rows a_i of the matrix A and the current iteration x_k. Since computing an inner product requires *o(n)* iterations, we are clearly unable to do more than a fixed number of these in a given iteration. The method is to map the rows of *A* onto a lower-dimensional space where the distance between the vectors is approximately preserved. Then we perform the calculations of equation (7) with respect to these lower dimensional vectors and select the best image.

The modified algorithm will converge to the worst-case solution $Ax = b$. In practice, we expect convergence to be much faster, especially when the lower dimension d on which we plot the rows is not too small. The progress at each iteration can be measured in terms of d and the current estimate.

7. Execution and execution time

Because the images in the algorithm are orthogonal, it is easy to see that the optimal image at the kth iteration can maximize $\|x_{k+1} - x_k\|_2$ or equivalently the following expression

$$\frac{|b[i] - (a_i, x_k)|}{\|a_{p(i)}\|_2}. \quad \textbf{(10)}$$

Unfortunately, computing this expression takes o(n) time. However, if we can significantly reduce the dimension of the vectors a_i and x_k used in the computations, then most of these computations can be done in each iteration and the best computations can be selected, leading to fast convergence. Our main idea is to map the vectors onto a lower-dimensional space. This operation leads to the estimation of the inner products (a_i, x_k) and the norms $\|a_i\|_2$ from the mapped data. In order to do this, we will introduce the Johnson-Linden-Strauss lemma. The famous Johnson-Linden-Strauss lemma states that any set of n points in a d-dimensional Euclidean space can be embedded in a k -dimensional Euclidean space, where k depends on the logarithm of n and is independent of d, such that all pairwise distances are controlled by an arbitrarily small factor. This can be summarized as follows [2].

Lemma 3. Suppose $\delta > 0$ and S is a finite set of points in R^n. Then for each d

$$d \geq C\frac{log|S|}{\delta^2}, \quad \textbf{(11)}$$

There exists a mapping $f: R^n \to R^d$ such that

$$(1-\delta)\|s_i - s_j\|^2 \leq \|\Phi(s_i) - \Phi(s_j)\|^2 \leq (1+\delta)\|s_i - s_j\|^2, \quad \textbf{(12)}$$

For $all\ s_i, s_j \epsilon\ S$, where C is a constant.

Remark 2

Although this lemma only guarantees the existence of a mapping, as stated, in the proof the mapping Φ is chosen as the image onto a random d -dimensional subspace of R^n. This result has been improved over time.

8. New Algorithm

The Johnson-Lynden-Strauss lemma allows us to map the rows of A and the estimates x_k onto a space of lower dimension. This operation allows us to compute expression (10) approximately with fewer operations, so that we can decide which hyperplane maps the current estimate. We mentioned that the mapping x_k will be done at each iteration. We use a similar strategy as the randomized Kekzmarz algorithm to select the rows, namely, by using the weights of the row normals. The slow convergence rate due to this type of selection leads to the use of a modified Kekzmarz stochastic method, called the Johnson-Lynden-Strauss stochastic Kekzmarz method, which we briefly describe below [6].

New model based on the Johnson-Lynden-Strauss image theorem has been made as follow:

1. Input: matrix A with dimensions $m \times n$ and vector $b \in R^m$, parameter d, choose an integer parameter $s = \frac{8\,log\,(m)}{\varepsilon^2}$, ε is the desired accuracy. Initial value x_0 .
2. Output: Approximate solution vector of $Ax = b$.

Set $k = 0$ and generate a random matrix R of dimensions n × s and calculate$h_i =$

1. $(AR)_i = a_i R$.

Repeat the following steps $o(n)$ times:

2. Randomly sample a set of N rows (according to $\mathrm{p_i} = \frac{\|\mathrm{a_i}\|^2}{\|\mathrm{A}\|^2_{\mathrm{F}}}$ or with uniform distribution). For each selected row, calculate

$\Delta_i = \frac{|b_i - (h_i, R^T x_k)|}{\|h_i\|_2}$ and put $j = arg\, \max_i \gamma_i$.

Select a_j and an arbitrary row a_p in the set of selected rows and calculate

$$\bar{\Delta}_j = \frac{|b_j - (a_j, x_k)|}{\|a_j\|_2} \quad , \quad \overline{\Delta_p} = \frac{|b_p - (a_p, x_k)|}{\|a_p\|_2}$$

If $\bar{\Delta}_p > \bar{\Delta}_j$, then let $j = p$. Calculate the image, $x_{k+1} = x_k + \frac{b_j - (a_j, x_k)}{\|a_j\|^2} a_j^T$

3. Set $k = k + 1$ and go to step 4.

6.4 Analytical justification

In this section we will analyze how the Johnson-Lynden-Strauss lemma is used in the Keczmarcz method. We assume that the system is real-valued and homogeneous $Ax = 0$ such that the rows of A all have unit normals and for the initial value x_0 we have $\|x_0\|_2 \leq 1$. These assumptions are not necessary but they make the analysis easier. First, we state a lemma that shows that the distances and geometric locations of the vectors used in the Keczmarcz stochastic method are approximately preserved based on the Johnson-Lynden-Strauss lemma[2].

Lemma 4. Let Φ be an $n \times d$ matrix with $d = C\delta^{-2} \log n$in the $RKJL$ method. Let $\gamma_i = \langle \Phi a_i, \Phi x_k \rangle$. Then for all i and k in the first $O(n)$ iteration of $RKJL$ we have$|\gamma_i - \langle a_i, x_k \rangle| \leq 2\delta$.

Proof: We apply the Johnson-Lynden-Strauss lemma with S containing all $a_i$and x_k in the first $O(n)$ iteration of $RKJL$. Then since $|S| \precsim n^2$, we have

$\gamma\_i = \langle \Phi a\_i, \Phi x\_k$

$\gamma_i = \langle \Phi a_i, \Phi x_k \rangle$

$= \frac{1}{4}(\|\Phi a_i + \Phi x_k\|_2^2 - \|\Phi a_i - \Phi x_k\|_2^2)$

$\leq \frac{1}{4}\left((1+\delta)\|a_i + x_k\|_2^2 - (1-\delta)\|a_i - x_k\|_2^2\right)$

$= \langle a_i, x_k \rangle + \frac{1}{4}\delta(\|a_i + x_k\|_2^2 + \|a_i - x_k\|_2^2)$

$\leq \langle a_i, x_k \rangle + 2\delta.$

Similarly, we *have* $\gamma_i \geq \langle a_i, x_k \rangle - 2\delta$, which completes the claim.

This shows that the term γ_i used for selection in the algorithm is approximately equal to the true values of $\langle a_i, x_k \rangle$. It also shows that when the estimate x_k approaches the true solution $x = 0$, the error δ starts to dominate and the possibility of improvement decreases. However, this does not cause any problems because it only occurs when the estimate is close to x. Given the structure of the $RKJL$ algorithm, it is clear that the convergence of $RKJL$ is at least as fast as the standard stochastic version. Furthermore, when the error generated by using Φ is small, the $RKJL$ method will image onto the best hyperplane outside those selected in that iteration. The probability of selecting this best row

when only one row is selected is significantly lower than the probability of selecting that row when a set of rows is selected, which means that the only case where $RKJL$ cannot converge completely faster is when $\langle a_i, x_k\rangle = \langle a_j, x_k\rangle$ holds for all rows a_i and a_j selected in the K th iteration. Given the current estimate of x_k, We can explicitly compare the expected improvement in the subsequent estimate for both the $RKJL$ method and the standard stochastic methods. First, we observe that if P represents the image at the kth iteration, then $x_{k-1} - x_k$ resides in the kernel of P and is thus orthogonal to the space in which P is imaged. This space contains $x_k - x\_$ because $x\_$ The solution of all equations is $Ax = b$ and $x_k = Px_{k-1}$. Therefore $x_k - x$ and $x_{k-1} - x_k$ are perpendicular. By implication, we have

$$\|x_k - x_{k+1}\|_2^2 = \|x - x_k\|_2^2 - \|x - x_{k+1}\|_2^2.$$

Relationship 1-4 shows that a larger $\|x_k - x_{k+1}\|_2$produces a greater improvement in that iteration. We then determine an estimate of x_k and analyze the expectation$\|x_k - x_{k+1}\|_2$ for the $RKJL$ method against its standard version [8].

Theorem 3. Determine an estimate x_k and show by x_{k+1} and $x^*{}_{k+1}$ that the subsequent estimates using the $RKJL$ method and the standard Kecksmarsh randomness, respectively. Let $\gamma_j^* = |\langle a_j, x_k\rangle|^2$ and then arrange them so that $\gamma_1^* \geq \gamma_2^* \geq \cdots \geq \gamma_m^*$. Then when

$$d = C\delta^{-2} \log n$$

$$E\|x_{k+1} - x\|_2^2 \leq min\ [E\left\|x^*{}_{K+1} - x\right\|_2^2 - \sum_{j=1}^{m}\left(p_j - \frac{1}{m}\right)\gamma_j^* + 2\delta,\ \ E\|x_{k+1}^* - x_{k+1}\|_2^2,$$

where,

$$p_j = \begin{cases} \frac{\binom{m-j}{n-1}}{\binom{m}{n}}, j \leq m - n + 1 \\ 0,\ \ j > m - n + 1 \end{cases}$$

are nonnegative values such that $\sum_{j=1}^{m} p_j = 1\ and\ .p_1 \geq p_2 \geq \cdots \geq p_m = 0.$

Theorem 3 establishes a lower bound that shows the improvement in the worst case, when the error due to the Johnson-Lynden-Strauss image causes the method to select a row A in the worst case. Numerical experiments prove the significant improvement. It is also very useful to identify some interesting scenarios, for example, when one or more rows are relatively large, in which case the standard RK algorithm (each row selected independently of the previous ones) will select this row regularly with high probability. Obviously, this may slow down the convergence (especially when these rows are highly correlated) and although there is still guaranteed exponential convergence, R is much larger in this case so that the guaranteed rate is slower. In $RKJL$, although the guaranteed worst-case coefficient is the same, these large rows are unlikely to be selected when their contribution to the solution is minimal [2].

9. Numerical results

We will demonstrate improved convergence using the $RKJL$ method. In this case, we simply randomly select the best row from n rows. This experiment demonstrates improved convergence using $RKJL$ when the error δ due to Φ becomes zero. This is the best improvement that can be hoped for in RKJL. When the problem size becomes very large, the effect of δ^2 in (11) becomes minimal,

so it may be quite realistic to make δ small. For these simulations, a 60,000 × 1,000 matrix with Bernoulli entries and a homogeneous system with uniformly randomly selected initial estimates are used. In Figure 7, we see that convergence is dramatically improved in this scenario [2, 7].

We then run the simulation using the Johnson-Lynden-Strauss matrix Φ. We generate the matrix Φ and the system $Ax = b$ as above, now we run the RKJL method using different values of d. In Figure 8 we see exactly what we expect, that we have much faster convergence with higher values of d (corresponding to lower values of δ). As discussed in the previous sections, the speed of convergence using larger d must be weighed against the increase in computations per iteration.

Since there is currently no theoretical guarantee as to exactly how convergence is affected by larger d, this comparison must be made empirically. Ultimately, it is clear that as m and nbecome larger, the effect on d will diminish. Thus, for very large systems, using $d = O(\log n)$ yields convergence that looks more like Figure 7 [2].

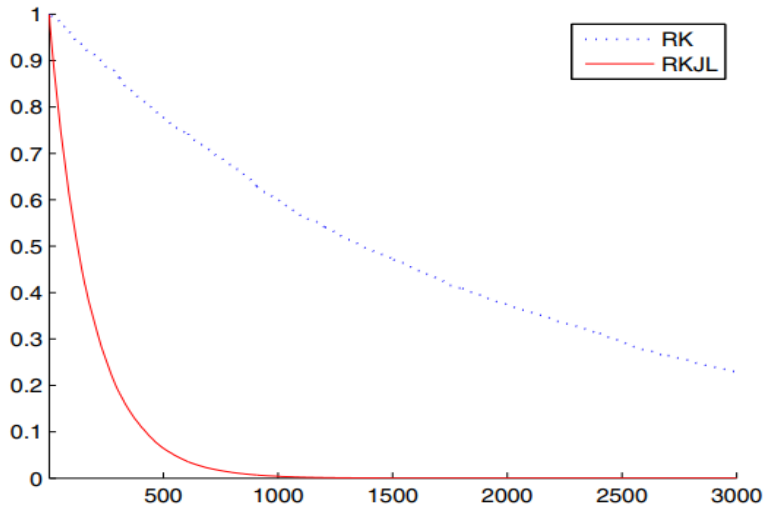

Figure 7: Error diagram for the Kecsmarz method with and without Johnson's lemma

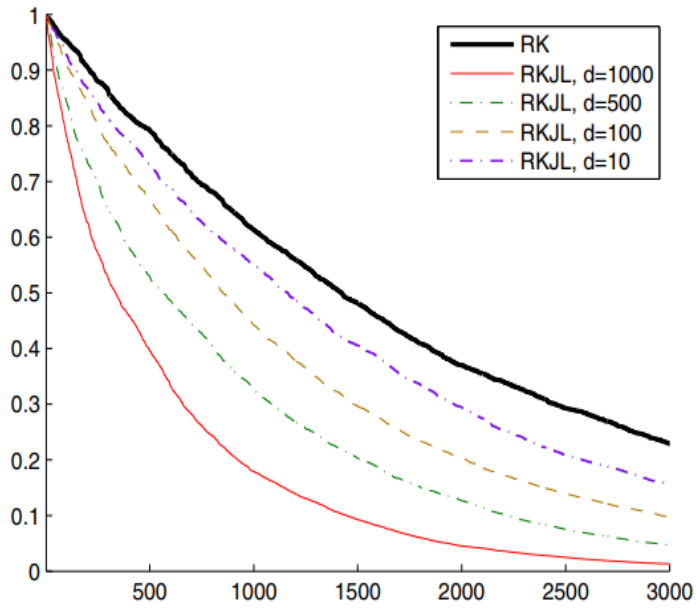

Figure 8: Error diagram for the Kecszmarz method with and without Johnson's lemma for different d

There is another version of the Keksmarz method, in which instead of one row, we select rows in a block form in each iteration, which has a higher convergence speed than other versions. In fact, with this method, we use fewer iterations to achieve the real solution [1,8].

10. Conclusion

In this paper, we have proposed an accelerated variant of the randomized Kaczmarz method that incorporates the Johnson–Lindenstrauss lemma and Monte Carlo techniques for efficient row selection and dimensionality reduction. The main contribution lies in demonstrating that projecting high-dimensional rows onto a lower-dimensional space preserves pairwise distances approximately while enabling the selection of nearly optimal rows with reduced computational cost. Numerical experiments confirm that the proposed method achieves faster convergence and improved stability compared to traditional randomized Kaczmarz algorithms, particularly for large-scale and ill-conditioned systems.

The study also highlights the effect of row selection strategies, showing that quasi-random sequences and weighted probability-based selections can reduce fluctuations and enhance convergence. While the theoretical guarantees provide bounds on the expected improvement, practical experiments demonstrate significant acceleration even in high-dimensional settings.

Future work could focus on extending this method to inconsistent systems with noise, exploring adaptive strategies for selecting the dimensionality parameter, and integrating block-wise row selection to further enhance computational efficiency. Overall, the proposed approach provides a robust and scalable solution for large linear systems, making it suitable for applications in scientific computing, image reconstruction, and signal processing.

References

[1] Yousefpanah, K., (2017), Resolved Monte Carlo Algorithms for solving linear systems, MSc dissertations, University of Guilan.
[2] Eldar, Yonina C., and Deanna Needell. "Acceleration of randomized Kaczmarz method via the Johnson–Lindenstrauss lemma." *Numerical Algorithms* 58.2 (2011): 163-177.
[3] Sabelfeld, Karl, and Nadja Loshchina. "Stochastic iterative projection methods for large linear systems." *Monte Carlo Methods and Applications*16.3-4 (2010): 343-359.
[4] Dasgupta, Sanjoy, and Anupam Gupta. "An elementary proof of the Johnson-Lindenstrauss lemma." *International Computer Science Institute, Technical Report* (1999): 99-006.
[5] Chan, T.F., Wan, W.L.: Analysis of projection methods for solving linear systems with multiple right-hand sides. SIAM J. Sci. Comput. **18**, 1698–1721 (1997).
[6] Dasgupta, S., Gupta, A.: An elementary proof of a theorem of Johnson and Lindenstrauss. Random Struct. Algorithms **22**(1), 60–65 (2003).
[7] B. Fathi Vajargah, M. Moradi, Diagonal scaling of ill-conditioned matrixes by genetic algorithm, Journal of Applied Mathematics, Statistics and Informatics (JAMSI) 8 (1), (2012).
[8] B. Fathi Vajargah , A way to obtain Monte Carlo matrix inversion with minimal error, Applied mathematics and computation 191 (1), 225-233, (2007).