Journal of Applied Analysis and Computation
Volume 13, Number 5, October 2023, 2522–2541
Website:http://www.jaac-online.com
DOI:10.11948/20220414

# THE MULTI-STEP RANDOMIZED KACZMARZ ALGORITHMS FOR SOLVING LARGE CONSISTENT LINEAR EQUATIONS*

Hai-Long Shen[1,†], Zhi-Min Xu[1] and Xin-Hui Shao[1]

**Abstract** In order to solve large scale consistent linear systems, based on the Kaczmarz algorithm, we propose two Multi-step Randomized Kaczmarz algorithms, denoted as MRK1 and MRK2, and give the corresponding convergence analysis. We consider using parameters to control the number of working rows. MRK1 algorithm uses a fixed parameter $m$, and the best result is that the parameter $m$ is the number of rows of the coefficient matrix, while MRK2 algorithm uses a randomly generated parameter $m$, which is more general. Finally, we carry out the corresponding numerical simulation experiments, the experimental results show that, compared with the RK, GRK algorithm, the new algorithm is faster and more efficient on CPU in most cases, and the maximum CPU acceleration is 12.54. And the difference between MRK1(s), MRK2 and MGRK on CPU is not very significant according to experimental results.

**Keywords** Kaczmarz algorithm, Randomized Kaczmarz algorithms, multi-step Randomized algorithm, linear system, convergence analysis.

**MSC(2010)** 65F08, 65F10.

## 1. Introduction

In many fields such as physics, aerospace, engineering and economics, when it relates to fluid mechanics, linear elasticity, electromagnetism, optimization, least squares, elliptic partial differential equation problems and so on, it is often necessary to solve a large-scale consistent linear systems. Therefore, solving the large-scale consistent linear systems has gradually become a research focus. And when dealing with large scale sparse matrices, there are a series of requirements such as fast efficiency, high accuracy, small storage, easy operation and execution, so it is particularly important to find fast and efficient methods. In this paper, we consider solving the large-scale consistent linear systems as follows:

$$Ax = b \tag{1.1}$$

†The corresponding author.
[1]Department of Mathematics, College of Sciences, Northeastern University, Shenyang 110819, China
*The authors were supported bythe Fundamental Research Funds for the Central Universities (N2224005-1), (N2005013) and the Natural Science Foundation of Liaon-Ning Province (No. 20170540323).
E-mail: hailong_shen@126.com(H. Shen), xzm0123456789@163.com(Z. Xu), xinhui1002@126.com(X. Shao)

where A is a coefficient matrix and $A \in R^{s\times t}$, b is a column vector and $b \in R^s$.

At present, the methods of solving linear systems (1.1) can be divided into direct method and iterative method. The direct methods include Gaussian elimination method, column principal element elimination method, matrix trigonometric decomposition method, square root method, catch-up method and so on, which are more convenient for solving small-scale linear systems. And classical iterative methods include Jacobi iterative method, Gauss-Seidel iterative method, etc. They are suitable for solving low-order dense matrix equations and is also one of the representative iterative methods, it uses orthogonal projection technology to speed up the computation process and has been widely concerned by experts and scholars. Due to its simplicity and efficiency, Kaczmarz algorithm has been widely used in many fields, such as wireless sensor networks, environmental monitoring, structural health monitoring and smart power grids [9], information retrieval [2], compressed sensing [4], electromagnetic field [10], image reconstruction [8], atmospheric imaging [7], etc.

In the 1930s, Polish mathematician Stefan Kaczmarz proposed the Kaczmarz algorithm to solve large sparse linear systems (1.1), and the idea of the classical Kaczmarz algorithm [6] is as follows:

If $A^{(i)}$ represents the ith row of the matrix A, $b^{(i)}$ represents the ith entry of the right-hand side vector b, the initial vector is denoted as $x_0$, and the index i is the working row selected for the kth iteration, then its iteration formula is as follows:

$$x_{k+1} = x_k + \frac{b^{(i)} - A^{(i)}x_k}{\left\|A^{(i)}\right\|_2^2}\left(A^{(i)}\right)^*, \quad k = 0, 1, 2, \cdots, \tag{1.2}$$

where $i = (k \ mod \ m) + 1$, $(\cdot)^*$ represents the conjugate transpose of the vector or matrix, $\|\cdot\|_2^2$ represents the Euclidean norm, $r_k^{(i)}$ represents the ith entry of the residual vector $r_k$. The geometric meaning of the formula (1.2) is to project the current iterative solution $x_k$ onto the hyperplane $b^{(i)} - A^{(i)}x_k = 0$ formed by the selected working row $A^{(i)}$.

In order to reduce the error, many scholars have begun to further research and exploration, it not only improves the convergence theory of Kaczmarz algorithm, but also made a lot of improvements. The greedy residual Kaczmarz algorithm [12] was proposed in 1984 by Ansorge. The greedy idea is mainly reflected in the idea of choosing the one with the largest entries of the residual vector in each iteration and finding the corresponding row for iteration. In 2009, Strohmer and Vershynin consider a random strategy to select the working rows, put forward the Randomized Kaczmarz algorithm(RK) [11], and prove that the RK algorithm converges with expected exponential rate. The RK algorithm benefits from the strategy $P\left(row = i_k\right) = \frac{\left\|A^{(i_k)}\right\|_2^2}{\|A\|_F^2}$, which makes the selection of working rows more random, so its convergence speed is faster. The disadvantage of the RK algorithm is that the probability criterion is invalid when the coefficient matrix is a unitary matrix or an orthogonal matrix. That is when the number of iteration steps or the number of equations is very large, the results of the RK algorithm and the classical Kaczmarz algorithm are roughly the same. Considering this situation, Bai and Wu proposed a new probability criterion for selecting working rows and gave the greedy Randomized Kaczmarz algorithm (GRK) in 2018;see [1].In the kth iteration, the index set $U_k$ is constructed by greedy strategy, that is, the corresponding row index with large distance from the current iteration solution to the hyperplane is put into

$U_k$ and then the working rows are selected from the index set $U_k$ by a random strategy $P\left(row = i_k\right) = \frac{\left\|\widetilde{r}_k^{(i_k)}\right\|_2^2}{\|\widetilde{r}_k\|_2^2}$. For the GRK algorithm, considering that the current residual vector $\widetilde{r}_k$ has the same maximum modulus margin $\widetilde{r}_k^{(i)} = \widetilde{r}_k^{(j)} = \widetilde{r}_k^{(l)}$ or their values are close enough $\widetilde{r}_k^{(i)} > \widetilde{r}_k^{(j)} > \widetilde{r}_k^{(l)}$ at three different positions, the probability of choosing row i is greatly reduced, it can be seen that the second largest residual and other larger residuals are also important. In order to make full use of the residual information calculated in each iteration, Jiang proposed the Multi-step Greedy Randomized Kaczmarz (MGRK) method [5]. That is, selects multiple working rows instead of one according to probability one time, which greatly improves the computing speed.

This paper also pays attention to the above shortcomings of the RK algorithm, and each iteration of the MGRK method needs to recalculate the probability and the index set $U_k$ by residuals, resulting in a large amount of calculation. In order to solve these problems, on the basis of the RK algorithm, we consider select multiple working rows one time, and propose two multi-step Randomized Kaczmarz algorithms, denoted as MRK1 algorithm and MRK2 algorithm, collectively known as MRK algorithm. MRK1 algorithm selects working rows according to probability and a fixed parameter $m$. It is found that the CPU of MRK1 algorithm is better when $m$ is larger (but does not exceed the number of rows of coefficient matrix). The difference is that the parameter $m$ of MRK2 algorithm is randomly generated by a function. In this paper, the convergence factor of the new algorithm is compared with that of RK, GRK and MGRK, and we give the corresponding theoretical analysis. Finally, numerical simulation experiments are carried out. Theoretical analysis and experimental results show that the two algorithms are more efficient and faster than RK, GRK and MGRK algorithm in most cases, although the improvement in the number of iterative steps is not great.

The structure of this paper is as follows: In Section 2 we introduce some knowledge about classical Kaczmarz, RK algorithm, MGRK algorithm and their convergence. In section 3, the new MRK algorithm is proposed, and two different forms of the multi-step Randomized Kaczmarz algorithm are given. In section 4, the convergence of the two new algorithms is analyzed and compared with the convergence factors of other algorithms. In section 5, the corresponding numerical simulation experiments are carried out. Finally, this paper is summarized.

## 2. Preliminary knowledge

First, the general form of the least-squares solution of the linear systems (1.1) can be written as

$$x = A^{\dagger}b + \left(E - P_{A^*}\right) y \ , \forall y \in R^t \tag{2.1}$$

where $A^{\dagger}$ is the Moore-Penrose pseudoinverse of the matrix A, E is the identity matrix, $P_{A^*}$ is the orthogonal projection on the row space of the matrix A, and $P_{A^*} = A^{\dagger}A$.When the matrix A is full rank, if $s \geq t$, then $A^{\dagger} = \left(A^*A\right)^{-1} A^*$; if s $< t$, then $A^{\dagger} = A^* \left(AA^*\right)^{-1}$. If the system of linear systems is consistent, because $\left(E - P_{A^*}\right) y \in ker\left(A^*A\right) = ker\left(A\right), A^{\dagger}b \in R_{A^*}$, and the row space $R_{A^*}$ is orthogonal to the null space $ker\left(A\right)$, it can be known as $x_\star = A^{\dagger}b$ is the least

Euclidean-norm solution of the system of linear systems (1.1).

For a matrix $Q = (q_{ij}) \in R^{s\times t}$, it is recorded $\|Q\|_F = \sqrt{\sum_{i=1}^{i=s}\sum_{j=1}^{j=t} q_{ij}^2}$ as the Frobenius norm of the matrix. In particular, if Q is a square matrix, it is recorded $\lambda_i$ as the eigenvalue of the matrix Q, and $tr(Q) = \sum_{i=1}^{i=s}\lambda_i = \sum_{i=1}^{i=s} q_{ii}$ can be defined as the trace of the matrix Q. This leads to another definition of Frobenius norm: $\|Q\|_F = \|Q^*Q\|_2$. And if Q is a Hermitian matrix, denote $\lambda_{min}(Q)$ and $\lambda_{max}(Q)$ as the minimum and maximum nonzero eigenvalues of the matrix, respectively.
For the consistent linear systems in (1.1), the following RK algorithm is proposed by Strohmer and Vershynin:

**Algorithm 1 RK**

**Input:** A,b,$T$,$x_0$
**Output:** $x_T$
1: **for** $k = 0, 1, 2\cdots T$ **do**
2: Select $i_k = \{1, 2, ..., s\}$ with probability $P(row = i_k) = \frac{\left\|A^{(i_k)}\right\|_2^2}{\|A\|_F^2}$;
3: Set $x_{k+1} = x_k + \frac{b^{(i)} - A^{(i)}x_k}{\left\|A^{(i)}\right\|_2^2}\left(A^{(i)}\right)^*, k = 0, 1, 2, \cdots$;
4: **end for**

The corresponding convergence analysis is given in [11], and the solution error satisfies the following theorem.

**Theorem 2.1** ( [11]). *If the system of linear systems (1.1) is consistent, the iteration sequence* $\{x_k\}_{k=0}^{\infty}$ *generated by the initial vector* $x_0 (x_0 \in R_{A^*})$ *and the RK algorithm converges to a unique least-norm solution* $x_\star = A^\dagger b$ *in expectation. And the solution error is expected to satisfy the following formula:*

$$E\|x_k - x_\star\|_2^2 \le \left(1 - \frac{\lambda_{\min}(A^*A)}{\|A\|_F^2}\right)^k \|x_k - x_0\|_2^2, \quad k = 1, 2, \cdots,$$

*where* $E_k(\cdot) = E\{\cdot | i_0, i_1, i_2 \cdots i_{k-1}\}$ *represents the mean error of the kth iteration.*

In order to solve the problem that the GRK algorithm may have the same maximum residual modulus margin or the residual modulus margins are very close at different locations, Jiang proposed an MGRK method in [5]. That is, select a series of working rows with largely relative residuals. The MGRK algorithm is as follows:

**Algorithm 2 MGRK**

**Input:** A,b,$T$,$x_0$
**Output:** $x_T$
1: **for** $k = 0, 1, 2\cdots T$ **do**
2: Compute $\varepsilon_k = \frac{1}{2}\left(\frac{1}{\|b - Ax_k\|_2^2}\max_{1\le i_k\le s}\left\{\frac{\left|b^{(i_k)} - A^{(i_k)}x_k\right|^2}{\left\|A^{(i_k)}\right\|_2^2}\right\} + \frac{1}{\|A\|_F^2}\right)$;
3: Define the index set of positive integers:
$$U_k = \left\{i_k \middle| \left|b^{(i_k)} - A^{(i_k)}x_k\right|^2 \ge \varepsilon_k \|b - Ax_k\|_2^2 \left\|A^{(i_k)}\right\|_2^2\right\};$$
4: Compute vector
$$\tilde{r}_k^{(i)} = \begin{cases} b^{(i)} - A^{(i)}x_k, & \text{if } i \in U_k \\ 0, & \text{otherwise} \end{cases};$$

5: Select $i_k = \{i_{k_1}, i_{k_2}, ..., i_{k_{m1}}\} \in U_k$ with probability $P\left(row = i_k\right) = \frac{\left|\tilde{r}_k^{(i)}\right|^2}{\|\tilde{r}_k\|_2^2}$;

6: Compute

$$\begin{cases} x_{k+\frac{1}{m1}} = x_k + \dfrac{b^{(i_{k_1})} - A^{(i_{k_1})} x_k}{\left\|A^{(i_{k_1})}\right\|_2^2} \left(A^{(i_{k_1})}\right)^* \\ x_{k+\frac{2}{m1}} = x_{k+\frac{1}{m1}} + \dfrac{b^{(i_{k_2})} - A^{(i_{k_2})} x_{k+\frac{1}{m1}}}{\left\|A^{(i_{k_2})}\right\|_2^2} \left(A^{(i_{k_2})}\right)^* \\ \vdots \\ x_{k+1} = x_{k+\frac{m1-1}{m1}} + \dfrac{b^{(i_{k_{m1-1}})} - A^{(i_{k_{m1-1}})} x_{k+\frac{m1-1}{m1}}}{\left\|A^{(i_{k_{m1-1}})}\right\|_2^2} \left(A^{(i_{k_{m1-1}})}\right)^* \end{cases};$$

7: **end for**

The convergence theorem of MGRK algorithm is also given in literature:

**Theorem 2.2** ( [5]). *If the system of linear equations (1.1) is consistent, the iteration sequence* $\{x_k\}_{k=0}^{\infty}$ *generated by the initial vector* $x_0 (x_0 \in R_{A^*})$ *and the MGRK algorithm converges to a unique least-norm solution* $x_\star = A^\dagger b$ *in expectation, and the solution error satisfies the following formula :*

$$\begin{aligned} & E\left\|x_k - x_\star\right\|_2^2 \\ \leq & \left(1 - \frac{1}{2}\left(\frac{1}{\gamma}\left\|A\right\|_F^2 + 1\right)\frac{\lambda_{min}\left(A^*A\right)}{\left\|A\right\|_F^2}\right)^{m1(k-1)} \cdot \left(1 - \frac{\lambda_{min}\left(A^*A\right)}{\left\|A\right\|_F^2}\right)^{m1} \left\|x_0 - x_\star\right\|_2^2. \end{aligned}$$

*In the above formula,* $k = 1, 2....$ *and* $\gamma = \max\limits_{1 \leq i \leq s} \sum\limits_{\substack{j=1 \\ j \neq i}}^{j=s} \left\|A^{(j)}\right\|_2^2$.

# 3. Multi-step Randomized Kaczmarz algorithm.

We note the probability criterion of selecting working row in each iteration of the RK algorithm, when the row norms of the coefficient matrix are very close $\left\|A^{(i_k)}\right\|_2^2 > \left\|A^{(i_l)}\right\|_2^2 > \left\|A^{(i_m)}\right\|_2^2$ or equal $\left\|A^{(i_k)}\right\|_2^2 = \left\|A^{(i_l)}\right\|_2^2 = \left\|A^{(i_m)}\right\|_2^2$, the probability of selecting the $i_k th$ row will be greatly reduced. We also notice that RK algorithm only selects one row in each iteration, this reduces the likelihood that other rows will be selected as working rows, which correspond to relatively high probabilities. And in order to avoid the calculation of the index set $U_k$ and probability in MGRK algorithm, the new algorithm adopts the random strategy of the RK algorithm.

In this paper, MRK algorithm has improved RK algorithm from the idea of improving GRK algorithm, a Multi-step Randomized Kaczmarz (MRK) method is proposed by selecting multiple working rows according to the probability. Meanwhile, two different forms of the multi-step Randomized Kaczmarz algorithm are given: One is to use the fixed parameter $m$ in each iteration to control the number

of working rows, the other is to randomly generate a parameter $m$ according to a function and select the rows of the coefficient matrix according to the probability. MRK algorithm can be regarded as a multi-step RK method, and because one time multiple rows of coefficient matrix are selected, the convergence process of iteration sequence to true solution is accelerated.

Firstly, consider the MRK1 algorithm with fixed parameter $m$ that selects the same number of working rows one time. Secondly, MRK2 algorithm is given to select different number of working rows one time, that is, random function generates parameter $m$.

**Algorithm 3 MRK1**

**Input:** A,b,$T$,$x_0$,and $m$
**Output:** $x_T$

1: **for** $k = 0, 1, 2 \cdots T$ **do**
2: Select $i_k = \{i_{k_1}, i_{k_2}, ..., i_{k_m}\}$ with probability $P\left(row = i_k\right) = \frac{\left\|A^{(i_k)}\right\|_2^2}{\|A\|_F^2}$;
3: Compute

$$\begin{cases} x_{k+\frac{1}{m}} = x_k + \dfrac{b^{(i_{k_1})} - A^{(i_{k_1})} x_k}{\left\|A^{(i_{k_1})}\right\|_2^2} \left(A^{(i_{k_1})}\right)^* \\ x_{k+\frac{2}{m}} = x_{k+\frac{1}{m}} + \dfrac{b^{(i_{k_2})} - A^{(i_{k_2})} x_{k+\frac{1}{m}}}{\left\|A^{(i_{k_2})}\right\|_2^2} \left(A^{(i_{k_2})}\right)^* \\ \vdots \\ x_{k+1} = x_{k+\frac{m-1}{m}} + \dfrac{b^{(i_{k_{m-1}})} - A^{(i_{k_{m-1}})} x_{k+\frac{m-1}{m}}}{\left\|A^{(i_{k_{m-1}})}\right\|_2^2} \left(A^{(i_{k_{m-1}})}\right)^* \end{cases} ; \qquad (3.1)$$

4: **end for**

**Algorithm 4 MRK2**

**Input:** A,b,$T$,$x_0$
**Output:** $x_T$

1: **for** $k = 0, 1, 2 \cdots T$ **do**
2: Compute $m = randi(s)$ to select the number of row indicators.
3: Select $i_k = \{i_{k_1}, i_{k_2}, ..., i_{k_m}\}$ with probability $P\left(row = i_k\right) = \frac{\left\|A^{(i_k)}\right\|_2^2}{\|A\|_F^2}$;

4: Compute

$$\begin{cases} x_{k+\frac{1}{m}} = x_k + \dfrac{b^{(i_{k_1})} - A^{(i_{k_1})} x_k}{\left\| A^{(i_{k_1})} \right\|_2^2} \left( A^{(i_{k_1})} \right)^* \\ x_{k+\frac{2}{m}} = x_{k+\frac{1}{m}} + \dfrac{b^{(i_{k_2})} - A^{(i_{k_2})} x_{k+\frac{1}{m}}}{\left\| A^{(i_{k_2})} \right\|_2^2} \left( A^{(i_{k_2})} \right)^* \\ \vdots \\ x_{k+1} = x_{k+\frac{m-1}{m}} + \dfrac{b^{(i_{k_{m-1}})} - A^{(i_{k_{m-1}})} x_{k+\frac{m-1}{m}}}{\left\| A^{(i_{k_{m-1}})} \right\|_2^2} \left( A^{(i_{k_{m-1}})} \right)^* \end{cases} ; \quad (3.2)$$

5: **end for**

# 4. Convergence analysis of MRK algorithm

This section we will prove that MRK1 and MRK2 algorithms are linear convergent, compare their convergence factors with those of RK, GRK and MGRK algorithm, and analyze the advantages and disadvantages.

Firstly,it is shown that if the MRK algorithm converges, it converges to the least-norm solution $x_\star = A^\dagger b$. This is because when the initial vector $x_0 \in R_{A^*}$, suppose the iteration sequence $\{x_k\}_{k=0}^{\infty}$ generated by RK algorithm converge to $\bar{x}$, and the limit of both sides of the iteration formula can be obtained:

$$\lim_{k\to\infty} x_{k+1} = \lim_{k\to\infty} \left( x_k + \frac{b^{(i_k)} - A^{(i_k)} x_k}{\left\| A^{(i_k)} \right\|_2^2} \left( A^{(i_k)} \right)^* \right), \quad k = 0, 1, 2, \cdots$$

then we have

$$\left( b^{(w)} - A^{(w)} \bar{x} \right) \left( A^{(w)} \right)^* = 0, \quad w = 1, 2, \ldots, s.$$

Consider the system of equations in the form of

$$A^* b - A^* A \bar{x} = 0,$$

that is

$$A^* b = A^* A \bar{x}.$$

According to the formula (2.1), we have $\bar{x} = A^\dagger b + (E - P_{A^*})\, y, \forall y \in R^t$. Considering that each step of iteration is carried out on the row space $R_{A^*}$, $\bar{x}$ can be regarded as a linear combination of matrix row vectors, that is, $\bar{x} \in R_{A^*}$, and there is $\bar{x} = x_\star = A^\dagger b$. The convergence theorem of the MRK1 algorithm is given below.

**Theorem 4.1.** *If the system of linear equations (1.1) is consistent, the sequence of iterative solutions $\{x_k\}_{k=0}^{\infty}$ generated by the initial vector $x_0 (x_0 \in R_{A^*})$ and MRK1 algorithm converges to a unique least-norm solution $x_\star = A^\dagger b$. The error of the solution satisfies the following formula in expectation:*

$$E_k \left\| x_{k+1} - x_\star \right\|_2^2 \leq \left( 1 - \frac{\lambda_{min}\left( A^* A \right)}{\|A\|_F^2} \right)^m \left\| x_k - x_\star \right\|_2^2, \quad k = 0, 1, 2, \ldots, \quad (4.1)$$

*so that it is possible to have for* $k = 1, 2, ....$

$$E\left\|x_k - x_\star\right\|_2^2 \leq \left(1 - \frac{\lambda_{min}\left(A^*A\right)}{\|A\|_F^2}\right)^{km} \left\|x_0 - x_\star\right\|_2^2. \tag{4.2}$$

**Proof.** First let's prove an orthogonal relation.From the iterative formula (3.1), we get

$$\begin{aligned}
A^{(i_{k_l})}\left(x_{k+\frac{l}{m}} - x_\star\right) &= A^{(i_{k_l})}\left(x_{k+\frac{l-1}{m}} - x_\star + \frac{b^{(i_{k_l})} - A^{(i_{k_l})}x_{k+\frac{l-1}{m}}}{\left\|A^{(i_{k_l})}\right\|_2^2}\left(A^{(i_{k_l})}\right)^*\right) \\
&= A^{(i_{k_l})}\left(x_{k+\frac{l-1}{m}} - x_\star\right) \\
&\quad + A^{(i_{k_l})}\left(\frac{A^{(i_{k_l})}x_\star - A^{(i_{k_l})}x_{k+\frac{l-1}{m}}}{\left\|A^{(i_{k_l})}\right\|_2^2}\right)\left(A^{(i_{k_l})}\right)^* \\
&= A^{(i_{k_l})}\left(x_{k+\frac{l-1}{m}} - x_\star\right) + A^{(i_{k_l})}x_\star - A^{(i_{k_l})}x_{k+\frac{l-1}{m}} \\
&= 0,
\end{aligned}$$

where $l = 1, 2, ..., m$. It can be known that $x_{k+\frac{l}{m}} - x_\star$ and $A^{(i_{k_l})}$ are perpendicular to each other, $\left(A^{(i_{k_l})}\right)^*$ and $x_{k+\frac{l}{m}} - x_{k+\frac{l-1}{m}}$ are parallel to each other. We can obtain $x_{k+\frac{l}{m}} - x_\star$ and $\left(x_{k+\frac{l}{m}} - x_{k+\frac{l-1}{m}}\right)^*$ are perpendicular to each other. According to the geometric relationship, we can get the formula:

$$\left\|x_{k+\frac{l}{m}} - x_\star\right\|_2^2 = \left\|x_{k+\frac{l-1}{m}} - x_\star\right\|_2^2 - \left\|x_{k+\frac{l}{m}} - x_{k+\frac{l-1}{m}}\right\|_2^2, l = 1, 2, \ldots, m.$$

The full proof is given below.

When $m = 1$,

$$E_k\left\|x_{k+1} - x_\star\right\|_2^2 \leq \left(1 - \frac{\lambda_{min}\left(A^*A\right)}{\|A\|_F^2}\right)\left\|x_k - x_\star\right\|_2^2$$

it is the same as the proof of RK algorithm which can refer to [11].

When $m = 2$, first for the $k = 0, 1, 2, \ldots$, we have

$$\left\|x_{k+\frac{1}{2}} - x_\star\right\|_2^2 = \left\|x_k - x_\star\right\|_2^2 - \left\|x_{k+\frac{1}{2}} - x_k\right\|_2^2, \tag{4.3}$$

$$\left\|x_{k+1} - x_\star\right\|_2^2 = \left\|x_{k+\frac{1}{2}} - x_\star\right\|_2^2 - \left\|x_{k+1} - x_{k+\frac{1}{2}}\right\|_2^2. \tag{4.4}$$

From the above formula (4.3) and (4.4), we can get

$$\left\|x_{k+1} - x_\star\right\|_2^2 = \left\|x_k - x_\star\right\|_2^2 - \left\|x_{k+\frac{1}{2}} - x_k\right\|_2^2 - \left\|x_{k+1} - x_{k+\frac{1}{2}}\right\|_2^2, \tag{4.5}$$

taking the expectation for $k$ in the formula (4.5), we get

$$E_k\left\|x_{k+1} - x_\star\right\|_2^2 = \left\|x_k - x_\star\right\|_2^2 - E_k\left\|x_{k+\frac{1}{2}} - x_k\right\|_2^2 - E_k\left\|x_{k+1} - x_{k+\frac{1}{2}}\right\|_2^2$$

$$
\begin{aligned}
&= \|x_k - x_\star\|_2^2 - \mathbb{E}_k \left\| \frac{b^{(i_{k_1})} - A^{(i_{k_1})} x_k}{\left\|A^{(i_{k_1})}\right\|_2^2} \left(A^{(i_{k_1})}\right)^* \right\|_2^2 \\
&\quad -\mathbb{E}_k \left\| \frac{b^{(i_{k_2})} - A^{(i_{k_2})} x_{k+\frac{1}{2}}}{\left\|A^{(i_{k_2})}\right\|_2^2} \left(A^{(i_{k_2})}\right)^* \right\|_2^2 \\
&= \|x_k - x_\star\|_2^2 - \sum_{i_{k_1}=1}^{s} \frac{\left|b^{(i_{k_1})} - A^{(i_{k_1})} x_k\right|^2}{\left\|A^{(i_{k_1})}\right\|_2^2} \cdot \frac{\left\|A^{(i_{k_1})}\right\|_2^2}{\|A\|_F^2} \\
&\quad - \sum_{i_{k_2}=1}^{s} \frac{\left|b^{(i_{k_2})} - A^{(i_{k_2})} x_{k+\frac{1}{2}}\right|^2}{\left\|A^{(i_{k_2})}\right\|_2^2} \cdot \frac{\left\|A^{(i_{k_2})}\right\|_2^2}{\|A\|_F^2} \\
&= \|x_k - x_\star\|_2^2 - \frac{\|b - Ax_k\|_2^2}{\|A\|_F^2} - \frac{\left\|b - Ax_{k+\frac{1}{2}}\right\|_2^2}{\|A\|_F^2} \\
&\leq \|x_k - x_\star\|_2^2 \\
&\quad - \frac{\lambda_{min}\left(A^*A\right)}{\|A\|_F^2} \|x_k - x_\star\|_2^2 - \frac{\lambda_{min}\left(A^*A\right)}{\|A\|_F^2} \left\|x_{k+\frac{1}{2}} - x_\star\right\|_2^2.
\end{aligned}
$$

It can be seen from the above formula that MRK1 algorithm converges in expected. Taking the expectation for $k$ in the formula (4.3) and (4.4), we get

$$
\begin{aligned}
E_k \left\|x_{k+\frac{1}{2}} - x_\star\right\|_2^2 &= \|x_k - x_\star\|_2^2 - E_k \left\|x_{k+\frac{1}{2}} - x_k\right\|_2^2 \\
&= \|x_k - x_\star\|_2^2 - \frac{\|b - Ax_k\|_2^2}{\|A\|_F^2} \\
&\leq \|x_k - x_\star\|_2^2 - \frac{\lambda_{min}\left(A^*A\right)}{\|A\|_F^2} \|x_k - x_\star\|_2^2 \\
&= \left(1 - \frac{\lambda_{min}\left(A^*A\right)}{\|A\|_F^2}\right) \|x_k - x_\star\|_2^2, \qquad (4.6) \\
E_k \|x_{k+1} - x_\star\|_2^2 &= \left\|x_{k+\frac{1}{2}} - x_\star\right\|_2^2 - E_k \left\|x_{k+1} - x_{k+\frac{1}{2}}\right\|_2^2 \\
&= \left\|x_{k+\frac{1}{2}} - x_\star\right\|_2^2 - \frac{\left\|b - Ax_{k+\frac{1}{2}}\right\|_2^2}{\|A\|_F^2} \\
&\leq \left\|x_{k+\frac{1}{2}} - x_\star\right\|_2^2 - \frac{\lambda_{min}\left(A^*A\right)}{\|A\|_F^2} \left\|x_{k+\frac{1}{2}} - x_\star\right\|_2^2 \\
&= \left(1 - \frac{\lambda_{min}\left(A^*A\right)}{\|A\|_F^2}\right) \left\|x_{k+\frac{1}{2}} - x_\star\right\|_2^2. \qquad (4.7)
\end{aligned}
$$

The inequality in the above formula (4.7) can be proved by the following formula:

$$
\|Ay\|_2^2 = (Ay, Ay) = y^* A^* Ay \geq \lambda_{min}\left(A^*A\right) \|y\|_2^2, \quad \forall y \in C^t.
$$

From (4.6) and (4.7), we have

$$E_k \left\| x_{k+1} - x_\star \right\|_2^2 \leq \left( 1 - \frac{\lambda_{min}\left(A^* A\right)}{\|A\|_F^2} \right)^2 \left\| x_k - x_\star \right\|_2^2 . \tag{4.8}$$

When the number of selected working rows is $m$, according to the orthogonal relationship, we can get

$$\left\| x_{k+\frac{l}{m}} - x_\star \right\|_2^2 = \left\| x_{k+\frac{l-1}{m}} - x_\star \right\|_2^2 - \left\| x_{k+\frac{l}{m}} - x_{k+\frac{l-1}{m}} \right\|_2^2, l = 1, 2, \ldots, m. \tag{4.9}$$

Further, we can get

$$E_k \left\| x_{k+1} - x_\star \right\|_2^2 \leq \left\| x_k - x_\star \right\|_2^2 - \frac{\lambda_{min}\left(A^* A\right)}{\|A\|_F^2} \sum_{l=1}^{l=m-1} \left\| x_{k+\frac{l}{m}} - x_\star \right\|_2^2 . \tag{4.10}$$

It can be seen from the above equation that the MRK1 algorithm converges in expected. According to the expectation on both sides of the equation (4.9), we have

$$\begin{aligned} E_k \left\| x_{k+\frac{l}{m}} - x_\star \right\|_2^2 &= \left\| x_{k+\frac{l-1}{m}} - x_\star \right\|_2^2 - E_k \left\| x_{k+\frac{l}{m}} - x_{k+\frac{l-1}{m}} \right\|_2^2 \\ &\leq \left( 1 - \frac{\lambda_{min}\left(A^* A\right)}{\|A\|_F^2} \right) \left\| x_{k+\frac{l-1}{m}} - x_\star \right\|_2^2, l = 1, 2, \ldots, m. \end{aligned}$$

So we have

$$E_k \left\| x_{k+1} - x_\star \right\|_2^2 \leq \left( 1 - \frac{\lambda_{min}\left(A^* A\right)}{\|A\|_F^2} \right)^m \left\| x_k - x_\star \right\|_2^2, k = 0, 1, 2, \ldots .$$

Taking the full expectation of the above formula has

$$E \left\| x_{k+1} - x_\star \right\|_2^2 \leq \left( 1 - \frac{\lambda_{min}\left(A^* A\right)}{\|A\|_F^2} \right)^m E \left\| x_k - x_\star \right\|_2^2, k = 0, 1, 2, \ldots .$$

By induction of $k$ we have

$$E \left\| x_k - x_\star \right\|_2^2 \leq \left( 1 - \frac{\lambda_{min}\left(A^* A\right)}{\|A\|_F^2} \right)^{km} \left\| x_0 - x_\star \right\|_2^2 .$$

It holds for $k = 0, 1, 2, 3, \ldots$. □

It can be seen from (4.1) that the larger the parameter $m$ is, the faster the convergence speed of MRK1 algorithm will be. Therefore, a conclusion is drawn in this paper. In order to achieve the best effect, the parameter $m$ of MRK1 algorithm is usually taken as the number of rows $s$ of coefficient matrix in linear systems (1.1).

The convergence proof of MRK2 algorithm is similar to that of MRK1 algorithm. The error formula of MRK2 algorithm is directly given below without proof.

**Theorem 4.2.** *If the system of linear equations (1.1) is consistent, the sequence of iterations* $\{x_k\}_{k=0}^{\infty}$ *produced by the initial vector* $x_0 (x_0 \in R_{A^*})$ *and the MRK2*

*algorithm converges to a unique least-norm solution $x_\star = A^\dagger b$. And the error of the solution satisfies the following formula in expectation:*

$$E_k \left\|x_{k+1} - x_\star\right\|_2^2 \leq \left(1 - \frac{\lambda_{min}\left(A^*A\right)}{\|A\|_F^2}\right)^{m_k} \left\|x_k - x_\star\right\|_2^2, k = 0, 1, 2, \ldots, \tag{4.11}$$

*then we can get for $k = 0, 1, 2, 3, \ldots$.*

$$E \left\|x_k - x_\star\right\|_2^2 \leq \left(1 - \frac{\lambda_{min}\left(A^*A\right)}{\|A\|_F^2}\right)^{\sum_{k=1}^{\infty} m_k} \left\|x_0 - x_\star\right\|_2^2, \tag{4.12}$$

*where $m_k$ is the parameter generated by the random function at the kth iteration.*

$\rho$ is denoted as the convergence factor, and the convergence factors of different algorithms are compared below. Firstly, the convergence factors of RK and MRK algorithms are compared.

$$\rho_{RK} = 1 - \frac{\lambda_{min}\left(A^*A\right)}{\|A\|_F^2} > \rho_{MRK1} = \left(1 - \frac{\lambda_{min}\left(A^*A\right)}{\|A\|_F^2}\right)^{m}, \tag{4.13}$$

$$\rho_{RK} = 1 - \frac{\lambda_{min}\left(A^*A\right)}{\|A\|_F^2} > \rho_{MRK2} = \left(1 - \frac{\lambda_{min}\left(A^*A\right)}{\|A\|_F^2}\right)^{m_k}. \tag{4.14}$$

From the two equations (4.13) and (4.14), the convergence rate of MRK algorithm is faster than that of RK algorithm for both fixed and randomly generated parameter. And by observing the following numerical examples, we can also draw the same conclusion.

Next, the convergence factors of MRK1 algorithm and MRK2 algorithm are compared. Since the parameter $m$ in each iteration of the MRK2 algorithm is uncertain, we consider the convergence factor of the previous $k$ step for comparison. We record the factor as $\rho\prime_{MRK1}$, $\rho\prime_{MRK2}$ and denote $m_k$ as the parameter generated by the $k$th iteration.

When $mk > \sum_{l=1}^{k} m_l$ , we have

$$\rho'_{MRK1} = \left(1 - \frac{\lambda_{min}\left(A^*A\right)}{\|A\|_F^2}\right)^{mk} < \rho'_{MRK2} = \left(1 - \frac{\lambda_{min}\left(A^*A\right)}{\|A\|_F^2}\right)^{\sum_{l=1}^{k} m_l}$$

the convergence speed of MRK1 is faster than that of MRK2, and vice versa.

Then, we compare the convergence factors of MRK1 algorithm and MGRK algorithm.According to the above analysis, we have

$$\rho_{MRK1} = \left(1 - \frac{\lambda_{min}\left(A^*A\right)}{\|A\|_F^2}\right)^{m}, \rho_{MGRK} = \left(1 - \frac{1}{2}\left(\frac{1}{\gamma}\|A\|_F^2 + 1\right)\frac{\lambda_{min}\left(A^*A\right)}{\|A\|_F^2}\right)^{m1}.$$

Let $\alpha = \frac{\lambda_{min}(A^*A)}{\|A\|_F^2}$ and $\beta = \frac{1}{2}\left(\frac{1}{\gamma}\|A\|_F^2 + 1\right)$, when $m > \frac{ln(1-\beta\alpha)}{ln(1-\alpha)} m1$, we have $\rho_{MRK1} < \rho_{MGRK}$, the convergence speed of the MRK1 algorithm is faster than that of the MGRK algorithm. When $m < \frac{ln(1-\beta\alpha)}{ln(1-\alpha)} m1$, we have $\rho_{MRK1} > \rho_{MGRK}$,

the convergence speed of the MRK1 algorithm is slower than that of the MGRK algorithm. And we can observed from numerical examples that MRK1 algorithm sometimes runs faster and has smaller CPU compared to MGRK algorithm. And the CPU of MGRK algorithm is similar to that of MRK1(s) algorithm, which is a good result.

Finally, the convergence factors of MRK2 algorithm and MGRK algorithm are compared. The convergence factor of the previous $k$ step is also considered for comparison.

$$\rho'_{MGRK} = \left(1 - \frac{1}{2}\left(\frac{1}{\gamma}\|A\|_F^2 + 1\right)\frac{\lambda_{min}(A^*A)}{\|A\|_F^2}\right)^{m1(k-1)}\left(1 - \frac{\lambda_{min}(A^*A)}{\|A\|_F^2}\right)^{m1},$$

$$\rho'_{MRK2} = \left(1 - \frac{\lambda_{min}(A^*A)}{\|A\|_F^2}\right)^{\sum_{l=1}^{k} m_l}.$$

When $\sum_{l=1}^{l=k} m_l > \frac{m1(k-1)ln(1-\beta\alpha)+m1ln(1-\alpha)}{ln(1-\alpha)}$, $\rho'_{MRK2} < \rho'_{MGRK}$, the convergence speed of the MRK2 algorithm is faster than that of the MGRK algorithm. And when $\sum_{l=1}^{l=k} m_l < \frac{m1(k-1)ln(1-\beta\alpha)+m1ln(1-\alpha)}{ln(1-\alpha)}$, there is $\rho'_{MGRK} < \rho'_{MRK2}$, that is, the convergence speed of the MRK2 algorithm is slower than that of the MGRK algorithm. More generally, their CPUs are very close. At the same time, according to the experimental results, the convergence rate of MRK2 is faster than that of MGRK for most practical applications matrices and full rank matrices. Other situations also need to be further improved.

## 5. Numerical experiment

In this section, coefficient matrices with different characteristics will be selected as numerical examples of RK, MRK, GRK and MGRK algorithms. Let IT and CPU represent the arithmetic mean of the running time and iteration steps of the corresponding method repeated 50 times, respectively. Assuming that the initial vector $x_0$ of all algorithms is a t-dimensional zero vector, the termination rule is that the relative solution error $RSE$ defined below reaches the precision $10^{-6}$ or the number of iteration steps exceeds 200000,

$$RSE = \frac{\|x_k - x_\star\|_2^2}{\|x_\star\|_2^2} \leq 10^{-6}.$$

For those that do not meet the accuracy requirements or the number of iteration steps exceeds 200000, the experimental results are represented by "$--$". In addition, we also report the speed-up ratio of MRK1, MRK2 against RK, namely

$$speed - up_{1,m} = \frac{CPU_{RK}}{CPU_{MRK1(m)}}, speed - up_2 = \frac{CPU_{RK}}{CPU_{MRK2}}.$$

All the numerical experiments are run on MATLAB(R2017a) on a personal laptop computer(Intel(R) Core(TM)i5-7200U CPU@2.50GHz 2.71 GHz). All methods

are executed precisely according to the procedures defined in the algorithm. For the first type matrices, we use the random function randn in the MATLAB to randomly generate the coefficient matrix$A$ and the solution vector $x_\star \in R^t$, then calculate the right-hand side vector $b = Ax_\star$. Numerical experiments were carried out on matrices of different orders, and the experimental results were shown in Tables 1–4.

**Table 1.** The experimental results of matrix ($s = 50$ with different $t$)

| $s \times t$ | | $50 \times 1000$ | $50 \times 2000$ | $50 \times 3000$ | $50 \times 4000$ | $50 \times 5000$ |
|---|---|---|---|---|---|---|
| RK | IT | 570.8 | 500.0 | 470.8 | 436.1 | 435.7 |
| | CPU | 0.0400 | 0.0401 | 0.0501 | 0.0627 | 0.0713 |
| MRK1 (2) | IT | 843.9 | 760.3 | 721.1 | 684 | 657.2 |
| | CPU | 0.0251 | 0.0293 | 0.0384 | 0.0489 | 0.0576 |
| | $speed - up_{1,2}$ | 1.59 | 1.37 | 1.30 | 1.28 | 1.24 |
| MRK1 (3) | IT | 696. 0 | 650.1 | 612.1 | 599.3 | 565.1 |
| | CPU | 0.0187 | 0.0237 | 0.0299 | 0.0413 | 0.0524 |
| | $speed - up_{1,3}$ | 2.14 | 1.69 | 1.68 | 1.52 | 1.36 |
| MRK1 ($s$) | IT | 545.2 | 496.7 | 492.4 | 451.4 | 460.81 |
| | CPU | 0.0073 | 0.0115 | 0.0188 | 0.0261 | 0.0361 |
| | $speed - up_{1,3}$ | 5.48 | 3.49 | 2.66 | 2.40 | 1.98 |
| MRK2 | IT | 572.5 | 507.8 | 503.3 | 468.5 | 441.6 |
| | CPU | 0.0087 | 0.0125 | 0.0198 | 0.0278 | 0.0333 |
| | $speed - up_2$ | 4.60 | 3.21 | 2.53 | 2.26 | 2.14 |
| GRK | IT | 135.2 | 114.4 | 105.8 | 100.8 | 97.2 |
| | CPU | 0.0141 | 0.0129 | 0.0148 | 0.0191 | 0.0223 |
| MGRK (2) | IT | 238.7 | 206.5 | 191.2 | 184.1 | 175.8 |
| | CPU | 0.0093 | 0.0099 | 0.0120 | 0.0144 | 0.0191 |
| MGRK (3) | IT | 246.3 | 210.5 | 198.9 | 189.2 | 182.9 |
| | CPU | 0.0086 | 0.0086 | 0.0107 | 0.0144 | 0.0171 |

In order to intuitively observe the acceleration effect of the new algorithm MRK on RK, GRK,MGRK, we have drawn the IT and CPU graphs of the MRK1, MRK2, RK,GRK and MGRK algorithms, as shown in Figures 1–8.

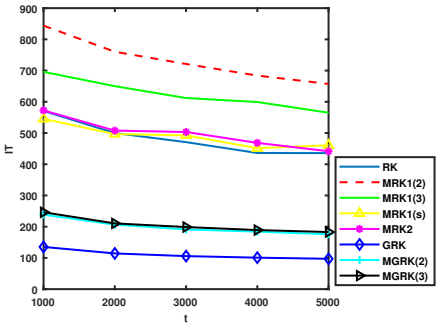


**Figure 1.** IT for $s = 50$ with different $t$.

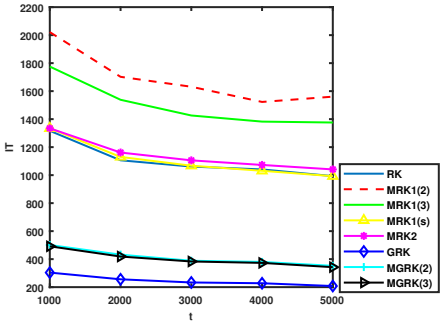


**Figure 2.** IT for $s = 100$ with different $t$.

**Table 2.** The experimental results of matrix ($s = 100$ with different $t$)

| $s \times t$ | | $100 \times 1000$ | $100 \times 2000$ | $100 \times 3000$ | $100 \times 4000$ | $100 \times 5000$ |
|---|---|---|---|---|---|---|
| RK | IT | 1316.7 | 1106.1 | 1061.2 | 1040.8 | 993.1 |
| | CPU | 0.0880 | 0.0853 | 0.1214 | 0.1580 | 0.1775 |
| MRK1 (2) | IT | 2021.6 | 1702.3 | 1631.4 | 1522.9 | 1560.6 |
| | CPU | 0.0625 | 0.0649 | 0.0924 | 0.1237 | 0.1496 |
| | $speed - up_{1,2}$ | 1.41 | 1.31 | 1.31 | 1.28 | 1.19 |
| MRK1 (3) | IT | 1775.5 | 1538.3 | 1426.1 | 1382.4 | 1375.9 |
| | CPU | 0.0460 | 0.0521 | 0.0750 | 0.1053 | 0.1419 |
| | $speed - up_{1,3}$ | 1.91 | 1.64 | 1.62 | 1.50 | 1.25 |
| MRK1 ($s$) | IT | 1334.3 | 1130.1 | 1067.3 | 1031.1 | 992.6 |
| | CPU | 0.0168 | 0.0248 | 0.0417 | 0.0729 | 0.0973 |
| | $speed - up_{1,3}$ | 5.24 | 3.44 | 2.91 | 2.17 | 1.82 |
| MRK2 | IT | 1334.5 | 1161.5 | 1105.6 | 1072.8 | 1040.9 |
| | CPU | 0.0174 | 0.0274 | 0.0463 | 0.0700 | 0.0883 |
| | $speed - up_2$ | 5.06 | 3.11 | 2.62 | 2.17 | 1.82 |
| GRK | IT | 303.9 | 256.0 | 233.4 | 227.8 | 207.9 |
| | CPU | 0.0320 | 0.0365 | 0.0518 | 0.0776 | 0.0869 |
| MGRK (2) | IT | 500.9 | 432.0 | 388.4 | 380.1 | 352.1 |
| | CPU | 0.0207 | 0.0242 | 0.0331 | 0.0489 | 0.0583 |
| MGRK (3) | IT | 491.6 | 418.6 | 383.3 | 373.2 | 342.6 |
| | CPU | 0.0169 | 0.0203 | 0.0279 | 0.0424 | 0.0490 |

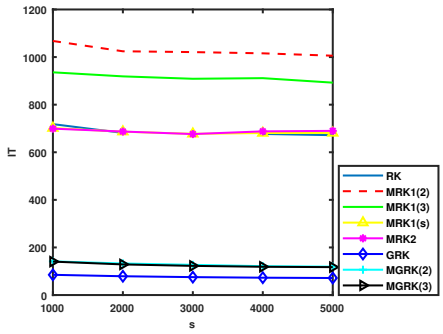


**Figure 3.** IT for $t = 50$ with different $s$.

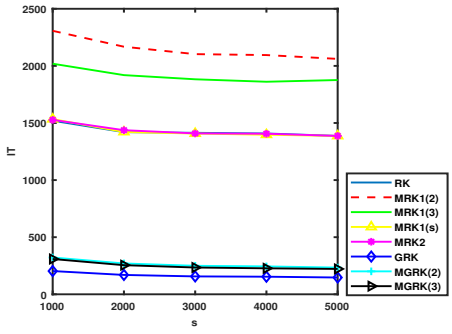


**Figure 4.** IT for $t = 100$ with different $s$.

After analysis, it is found that although MRK algorithm is not dominant in the number of iterative steps, when the algorithm parameters are selected properly, the CPU of MRK1 and MRK2 algorithms is greatly shortened compared with RK and GRK algorithms. For the randomly generated fat matrix (s<t), MRK1 algorithm can always accelerate RK algorithm regardless of the value of parameter $m$, and with the increase of parameter $m$, the acceleration effect of RK algorithm is more and more obvious. The minimum acceleration is 1.19, the maximum acceleration is 5.48, and the running speed is significantly improved. For thin matrix (s>t), although MRK1 algorithm cannot achieve the acceleration effect when $m$ is very small, it can still achieve the acceleration effect with the increase of $m$, and the larger the $m$, the better the effect, the maximum acceleration is 12.54. No matter

**Table 3.** The experimental results of matrix ($t = 50$ with different $s$)

| $s \times t$ | | $1000 \times 50$ | $2000 \times 50$ | $3000 \times 50$ | $4000 \times 50$ | $5000 \times 50$ |
|---|---|---|---|---|---|---|
| RK | IT | 718.2 | 681.3 | 676.2 | 676.9 | 671.8 |
| | CPU | 0.0544 | 0.0621 | 0.0732 | 0.1165 | 0.1160 |
| MRK1 (2) | IT | 1067.6 | 1024.3 | 1020.9 | 1015.7 | 1005.7 |
| | CPU | 0.0593 | 0.0905 | 0.1222 | 0.1644 | 0.1974 |
| | $speed - up_{1,2}$ | 0.92 | 0.69 | 0.60 | 0.71 | 0.59 |
| MRK1 (3) | IT | 935.9 | 918.9 | 908.8 | 911.4 | 892.5 |
| | CPU | 0.0434 | 0.0626 | 0.0865 | 0.1147 | 0.1345 |
| | $speed - up_{1,3}$ | 1.25 | 0.99 | 0.85 | 1.02 | 0.86 |
| MRK1 ($s$) | IT | 701.6 | 686.4 | 677.4 | 680.8 | 681.2 |
| | CPU | 0.0044 | 0.0078 | 0.0133 | 0.0320 | 0.0430 |
| | $speed - up_{1,3}$ | 12.36 | 7.96 | 5.50 | 3.64 | 2.70 |
| MRK2 | IT | 699.2 | 687.1 | 676.5 | 687.8 | 689.9 |
| | CPU | 0.0046 | 0.0064 | 0.0091 | 0.0153 | 0.0279 |
| | $speed - up_2$ | 11.83 | 9.70 | 8.04 | 7.61 | 4.16 |
| GRK | IT | 85.0 | 79.0 | 75.3 | 72.9 | 71.6 |
| | CPU | 0.0097 | 0.0115 | 0.0147 | 0.0186 | 0.0215 |
| MGRK (2) | IT | 141.5 | 132.6 | 126.9 | 121.3 | 119.3 |
| | CPU | 0.0062 | 0.0066 | 0.0081 | 0.0114 | 0.0131 |
| MGRK (3) | IT | 140.5 | 128.7 | 122.6 | 118.8 | 117.3 |
| | CPU | 0.0052 | 0.0056 | 0.0060 | 0.0085 | 0.0100 |

**Table 4.** The experimental results of matrix ($t = 100$ with different $s$)

| $s \times t$ | | $1000 \times 100$ | $2000 \times 100$ | $3000 \times 100$ | $4000 \times 100$ | $5000 \times 100$ |
|---|---|---|---|---|---|---|
| RK | IT | 1519.7 | 1419.8 | 1413.9 | 1408.1 | 1386.8 |
| | CPU | 0.1195 | 0.1332 | 0.1566 | 0.2177 | 0.2411 |
| MRK1 (2) | IT | 2307.0 | 2167.8 | 2102.9 | 2094.7 | 2061.2 |
| | CPU | 0.1309 | 0.1927 | 0.2550 | 0.3328 | 0.3975 |
| | $speed - up_{1,2}$ | 0.91 | 0.69 | 0.61 | 0.65 | 0.61 |
| MRK1 (3) | IT | 2018.9 | 1919.1 | 1882.6 | 1861.1 | 1876.1 |
| | CPU | 0.0868 | 0.1325 | 0.1728 | 0.2290 | 0.2768 |
| | $speed - up_{1,3}$ | 1.38 | 1.01 | 0.91 | 0.95 | 0.87 |
| MRK1 ($s$) | IT | 1533.7 | 1421.4 | 1407.5 | 1397.2 | 1386.6 |
| | CPU | 0.0096 | 0.0115 | 0.0166 | 0.0323 | 0.0461 |
| | $speed - up_{1,3}$ | 12.54 | 11.58 | 9.43 | 6.74 | 5.23 |
| MRK2 | IT | 1527.3 | 1437.0 | 1408.6 | 1405.8 | 1386.7 |
| | CPU | 0.0144 | 0.0162 | 0.0185 | 0.0262 | 0.0416 |
| | $speed - up_2$ | 8.30 | 8.22 | 8.46 | 8.31 | 5.80 |
| GRK | IT | 203.4 | 169.5 | 156.6 | 154.0 | 147.0 |
| | CPU | 0.0249 | 0.0275 | 0.0361 | 0.0530 | 0.0607 |
| MGRK (2) | IT | 322.1 | 270 | 249.4 | 244.1 | 233.8 |
| | CPU | 0.0146 | 0.0163 | 0.0207 | 0.0311 | 0.0342 |
| MGRK (3) | IT | 309.1 | 255.0 | 234.5 | 227.4 | 222.1 |
| | CPU | 0.0111 | 0.0120 | 0.0150 | 0.0231 | 0.0243 |

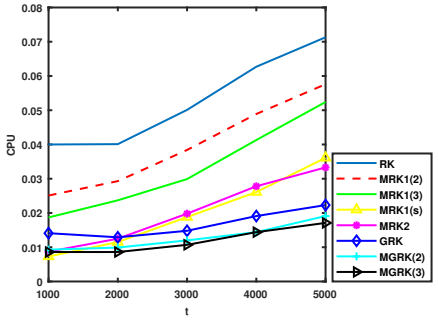


**Figure 5.** CPU for $s = 50$ with different $t$.

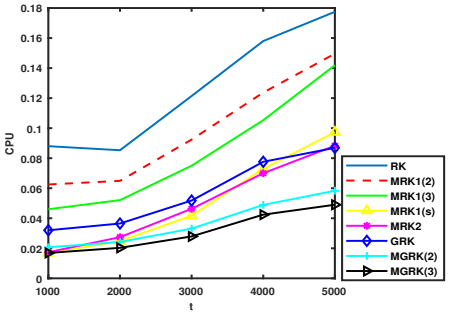


**Figure 6.** CPU for $s = 100$ with different $t$.

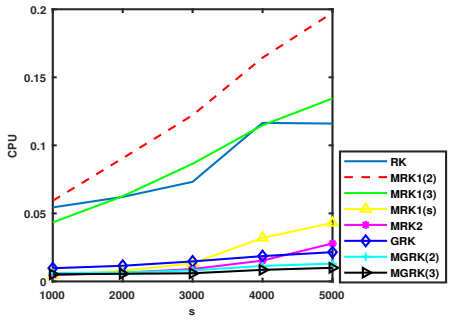


**Figure 7.** CPU for $t = 50$ with different $s$.

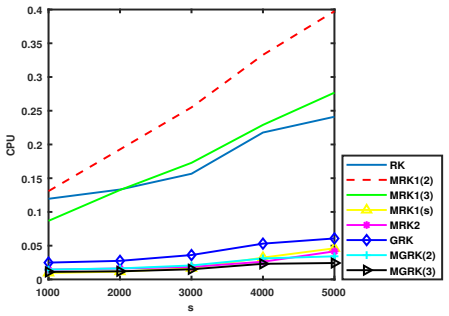


**Figure 8.** CPU for $t = 100$ with different $s$.

fat matrix or thin matrix, firstly, the CPU acceleration effect of MRK2 algorithm on RK algorithm is confirmed. Secondly, it is found that MRK1(s) and MRK2 algorithms can accelerate GRK algorithm in most cases, especially for fat matrix, the maximum acceleration is 2.59. Under the premise of similar iterative steps, when $m$ takes the number of rows as $s$, sometimes the effect of MRK2 algorithm is even better than MRK1 algorithm.

The second kind of matrices is the full-rank matrix with certain structures and properties selected from the SuiteSparse Matrix Collection website [3], such as the order, rank, condition number (cond (A)) and density of the test matrix. where density is defined as

$$\text{density} = \frac{\text{the number of nonzero elements in matrix } A}{\text{the number of all elements in matrix } A}.$$

some properties of the second type of matrices are recorded in Table 5. The experimental results are shown in Table 6.

We find that MRK1 algorithm can always accelerate RK algorithm regardless of the value of parameter $m$, and the acceleration effect becomes more and more significant with the increase of parameter $m$, the maximum acceleration is 10.65. Compared MRK1, MRK2 algorithm and GRK algorithm, we found that MRK1(s) and MRK2 algorithms can always accelerate GRK algorithm, and the maximum acceleration is 4.55. It can also be concluded that although the improvement in the number of iteration steps is not large, the MRK algorithm has a significant

**Table 5.** Full rank matrix

| Matrix Name | Order | Density/% | Rank | Condition Number |
|---|---|---|---|---|
| Stranke94 | $10 \times 10$ | 90.00 | 10 | 5.173300e + 01 |
| ash958 | $958 \times 292$ | 0.68 | 292 | 3.210358e + 00 |
| bibd_15_3 | $105 \times 455$ | 2.86 | 105 | 1.882938e + 00 |
| cari | $400 \times 1200$ | 31.83 | 400 | 3.129239e + 00 |
| can_24 | $24 \times 24$ | 27.78 | 24 | 7.775851e + 01 |

**Table 6.** Experimental results for full-rank matrices

| $s \times t$ | | Stranke94 | ash958 | bibd_15_3 | cari | can_24 |
|---|---|---|---|---|---|---|
| RK | IT | 1893.1 | 6330.4 | 1252.8 | 39008.2 | 4697 |
| | CPU | 1.0120 | 0.4794 | 0.0799 | 2.1171 | 0.3799 |
| MRK1 (2) | IT | 28528.0 | 9342.1 | 1870.7 | 60089.3 | 7086.6 |
| | CPU | 0.5705 | 0.5054 | 0.0515 | 1.2289 | 0.3237 |
| | $speed - up_{1,2}$ | 1.77 | 0.95 | 1.55 | 1.72 | 1.17 |
| MRK1 (3) | IT | 28442.3 | 9190.2 | 1842.9 | 55172.2 | 6274.8 |
| | CPU | 0.5715 | 0.5046 | 0.0561 | 1.1477 | 0.2354 |
| | $speed - up_{1,3}$ | 1.77 | 0.95 | 1.42 | 1.84 | 1.61 |
| MRK1 ($s$) | IT | 20835.6 | 6365.0 | 1237.1 | 38683.7 | 4620.4 |
| | CPU | 0.1631 | 0.0450 | 0.0102 | 0.2094 | 0.0757 |
| | $speed - up_{1,3}$ | 6.20 | 10.65 | 7.83 | 10.11 | 5.02 |
| MRK2 | IT | 22387.0 | 6402.6 | 1294 | 41970 | 4809.8 |
| | CPU | 0.2504 | 0.0461 | 0.0114 | 0.3121 | 0.0805 |
| | $speed - up_2$ | 4.04 | 10.40 | 7.01 | 6.78 | 4.71 |
| GRK | IT | 8412.8 | 812.6 | 268.1 | 4694.8 | 939.5 |
| | CPU | 0.5259 | 0.1449 | 0.0252 | 0.2854 | 0.3228 |
| MGRK (2) | IT | 20146.7 | 1258.8 | 452.1 | 9210.6 | 1470.0 |
| | CPU | 0.4625 | 0.0777 | 0.0178 | 0.2145 | 0.1762 |
| MGRK (3) | IT | 26015.9 | 1154.4 | 437.2 | 11471.9 | 1362.4 |
| | CPU | 0.4713 | 0.0603 | 0.0140 | 0.2084 | 0.1292 |

CPU acceleration effect on RK,GRK algorithm. Finally, as the acceleration effect of MRK1 algorithm becomes more and more obvious with the increase of parameter $m$, the comprehensive acceleration effect of MRK1 algorithm is better than that of MRK2 algorithm.

The third kind of examples are matrices from practical applications. These matrices are from combinatorial problems (bibd_15_3, n2c6-b1, flower_5_1), least squares problems (ash958), linear programming problem (cari), structural problems (can_24, bcsstm01), counterexample problems (rgg010), wbg model problems (WorldCities) and so on. The properties of the matrix are shown in Table 7. The experimental results are shown in Table 8.

For the third type of examples, the same conclusion is that the acceleration effect of the MRK1 algorithm becomes more and more significant as the parameter $m$

**Table 7.** Matrices from practical applications

| Matrix Name | Order | Density/% | Rank | Condition Number |
|---|---|---|---|---|
| n2c6-b1 | $105 \times 15$ | 13.33 | 14 | 4.336746e + 15 |
| rgg010 | $10 \times 10$ | 76.00 | 4 | Inf |
| bcsstm01 | $48 \times 48$ | 1.04 | 24 | Inf |
| flower_5_1 | $211 \times 201$ | 1.42 | 179 | 2.000742e + 16 |
| WorldCities | $315 \times 100$ | 23.87 | 100 | 6.599990e + 01 |

**Table 8.** Experimental results for full-rank matrices

| $s \times t$ | | n2c6-b1 | rgg010 | bcsstm0 | flower_5_1 | WorldCities |
|---|---|---|---|---|---|---|
| RK | IT | 166.3 | 400.5 | 202.9 | 35188.1 | 44008.3 |
| | CPU | 0.0108 | 0.0221 | 0.0133 | 2.2594 | 2.6710 |
| MRK1 (2) | IT | 167.4 | 408.2 | 178.1 | 35299.2 | 66259.2 |
| | CPU | 0.0108 | 0.0218 | 0.0116 | 2.2334 | 1.9075 |
| | $speed - up_{1,2}$ | 1.00 | 1.01 | 1.15 | 1.01 | 1.40 |
| MRK1 (3) | IT | 253.6 | 610.3 | 263.5 | 52284.7 | 59062.7 |
| | CPU | 0.0087 | 0.0135 | 0.0073 | 1.7660 | 1.4273 |
| | $speed - up_{1,3}$ | 1.24 | 1.64 | 1.82 | 1.28 | 1.87 |
| MRK1 ($s$) | IT | 182 | 387 | 175 | 35042.5 | 44582.7 |
| | CPU | 0.0012 | 0.0020 | 0.0015 | 0.2727 | 0.2309 |
| | $speed - up_{1,3}$ | 9.00 | 11.05 | 8.87 | 8.29 | 11.57 |
| MRK2 | IT | 172.4 | 484.3 | 190.2 | 35482.1 | 44510.3 |
| | CPU | 0.0016 | 0.0064 | 0.0013 | 0.2847 | 0.2474 |
| | $speed - up_2$ | 6.75 | 3.45 | 10.23 | 7.94 | 10.80 |
| GRK | IT | 30.8 | 141.1 | 24 | 4941.8 | 8390.9 |
| | CPU | 0.0035 | 0.0091 | 0.0018 | 0.4676 | 0.7708 |
| MGRK (2) | IT | 61.4 | 390.0 | 53.3 | 9479.5 | 14187.3 |
| | CPU | 0.0019 | 0.0101 | 0.0016 | 0.3364 | 0.5062 |
| MGRK (3) | IT | 69.9 | 513.6 | 62 | 10834.1 | 14133.2 |
| | CPU | 0.0018 | 0.0104 | 0.0012 | 0.3106 | 0.3678 |

increases. And we found that for most matrices, MRK1(s) and MRK2 algorithms both play an accelerated role in RK, GRK and MGRK algorithms. And the maximum speedup is 11.57. The difference for the third kind matrices is the acceleration effect of MRK1(s) is better than that of MRK2 in terms of IT and CPU sometimes.

Finally, for MRK algorithm and MGRK algorithm, for the same matrix, MGRK algorithm needs fewer iterative steps than MRK algorithm, but such a conclusion cannot be given for the CPU. That is, sometimes MRK algorithm achieves better CPU effect, and sometimes MGRK algorithm achieves better CPU effect, but the new algorithm is also efficient, and its CPU performance is remarkable. And it can be observed that the difference between the two CPUs is not very significant through experimental data in Tables 1–4 and Figures 7–8. In addition, for some matrices with special structures and properties (Tables 7–8), MRK algorithm has more advantages on CPU than MGRK algorithm.

We also observed MRK algorithm is superior to MGRK algorithm in optimal parameter selection. This is because on the premise that the parameter of MRK1 algorithm is smaller than the number of rows of the coefficient matrix, the larger the parameter $m$ is, the faster the convergence rate will be. In addition, the parameters of MRK2 algorithm are generated randomly according to the function, so a better effect can be achieved without intervention. However, the parameter $m1$ of MGRK algorithm is limited by the index set $U_k$, and $m1$ cannot always increase. Therefore, the optimal parameter corresponding to each iteration is different, so it is difficult to determine the optimal parameter for MGRK algorithm.

# 6. Conclusion

In this paper, the MRK algorithm is proposed to select some rows of the coefficient matrix according to the probability of each iteration. According to different rules of selecting parameter $m$, the algorithm is divided into MRK1 and MRK2. Theoretical analysis and numerical experiments show that although MRK algorithm has no significant improvement in the number of iteration steps, it has a very significant acceleration effect on CPU, and the maximum acceleration is 12.54. Therefore, MRK algorithm has a good improvement. However, the experiment found that when the row and column ratio (the number of rows to the number of columns) of the coeffiient matrix is very large, and the parameter $m$ of the MRK1 algorithm is very small, the acceleration effect is not achieved. Therefore, the MRK1 algorithm needs to be further improved.

# Acknowledgements

The authors would like to thank the anonymous referee of this paper for very helpful comments and suggestions.

# References

[1] Z. Bai and W. Wu, *On greedy randomized Kaczmarz method for solving large sparse linear systems,* SIAM Journal on Scientific Computing, 2018, 40(1), A592–A606.

[2] J. Cai and Y. Tang, *A new randomized Kaczmarz based kernel canonical correlation analysis algorithm with applications to information retrieval,* Neural Networks, 2018, 98, 178–191.

[3] T. A. Davis and Y. Hu, *The university of Florida sparse matrix collection,* ACM Transactions on Mathematical Software (TOMS), 2011, 38(1), 1–25.

[4] C. Gu, M. Sun and P. F, *On randomized sampling Kaczmarz method with application in compressed sensing,* Mathematical Problems in Engineering, 2020, 1–11.

[5] X. Jiang, *Subspace algorithm and random Kaczmarz algorithm for web page sorting and their applications*, PhD thesis, Shanghai University (Doctoral Dissertation), 2019.

[6] S. Karczmarz, *Angenaherte auflosung von systemen linearer Glei-Chungen,* Bull. Int. Acad. Pol. Sic. Let., Cl. Sci. Math. Nat., 1937, 355–357.

[7] C. Pechstein, M. Eslitzbichler and R. Ramlau, *An H1-Kaczmarz reconstructor for atmospheric tomography,* Journal of Inverse and Ill-Posed Problems, 2013, 21(3), 431–450.

[8] C. Popa and R. Zdunek, *Kaczmarz extended algorithm for tomographic image reconstruction from limited-data,* Mathematics and Computers in Simulation, 2004, 65(6), 579–598.

[9] P. Ramanan, G. Kamath and W. Song, *Distributed randomized Kaczmarz and applications to seismic imaging in sensor network,* In 2015 International Conference on Distributed Computing in Sensor Systems, IEEE, 2015, 169–178.

[10] D. Schmidt, J. Leliaert and O. Posth, *Interpreting the magnetorelaxometry signal of suspended magnetic nanoparticles with Kaczmarz'algorithm,* Journal of Physics D: Applied Physics, 2017, 50(19), 195002.

[11] T. Strohmer and R. Vershynin, *A randomized Kaczmarz algorithm with exponential convergence,* Journal of Fourier Analysis and Applications, 2009, 15(2), 262.

[12] J. Yin, Y. Du and K. Zhang, *Greedy distance stochastic Kaczmarz method for solving large sparse linear equations,* Journal of Tongji University (NATURAL SCIENCE EDITION), 2020, 48(8), 1224–1231.