AIMS Mathematics, 9(12): 33843–33860.
DOI: 10.3934/math.20241614
Received: 25 September 2024
Revised: 21 November 2024
Accepted: 25 November 2024
Published: 28 November 2024

https://www.aimspress.com/journal/Math

Research article

On maximum residual block Kaczmarz method for solving large consistent linear systems

Wen-Ning Sun and Mei Qin*

College of Science, University of Shanghai for Science and Technology, Shanghai 200093, China

* **Correspondence:** Email: qin5670830@163.com.

Abstract: In this paper, we propose two block variants of the Kaczmarz method for solving large-scale consistent linear equations $Ax = b$. The first method, named the maximum residual block Kaczmarz (denoted as MRBK) method, first partitions the coefficient matrix, and then captures the largest block in the residual vector during each block iteration to ensure that it is eliminated first. Simultaneously, in order to avoid the pseudo-inverse calculation of the MRBK method during block iteration and improve the convergence speed, we further propose the maximum residual average block Kaczmarz method. This method completes the iterative process by projecting the current solution vector onto each row of the constrained subset of the matrix A and averaging it with different extrapolation steps. We analyze and prove the deterministic convergence of both methods. Numerical experiments validate our theory and show that our proposed methods are superior to some other block Kaczmarz methods.

Keywords: consistent linear system; maximum residual block Kaczmarz; maximum residual average block Kaczmarz; free pseudo-inverse; convergence property
Mathematics Subject Classification: 65F10, 65F20

1. Introduction

Consider the following large-scale system of linear equations:

$$Ax = b, \quad \text{with} \quad A \in \mathbb{R}^{m\times n} \quad \text{and} \quad b \in \mathbb{R}^m, \tag{1.1}$$

where researchers often employ iterative methods [1, 2] to solve them. A simple and effective method is the Kaczmarz method [3], which is extensively utilized across various large-scale computing fields, including computed tomography [4–8], image reconstruction [9–11], signal processing [11, 12], distributed computing [13, 14], etc. As a row iteration method, the classic Kaczmarz method iteratively updates the solution vector by selecting the working row of the coefficient matrix in sequence and projecting it orthogonally to the hyperplane where the row is located. To be more specific, let $A^{(i)}$

represent the ith row of A, and $b^{(i)}$ represent the ith entry of b. Given an initial approximation value x_0, the Kaczmarz method can be expressed as follows:

$$x_{k+1} = x_k + \frac{(b^{(i_k)} - A^{(i_k)}x_k)}{\| A^{(i_k)} \|_2^2}(A^{(i_k)})^T$$

where $(\cdot)^T$ and $\| \cdot \|_2$ denote the transpose and Euclidean norm of a vector or matrix, respectively, and the target row index $i_k = mod(k, m) + 1$.

The theory of the Kaczmarz method has seen significant development since its inception. Initially, Kaczmarz [3] demonstrated the convergence of this method. Subsequently, Galántai [15] provided an upper bound on its convergence rate, while Knight conducted error analysis on it under a limited precision operation in [16]. In recent years, Bai and Liu [17] established a new convergence theorem for the Kaczmarz method using the block Meany inequality. For further research literature, please refer to [18, 19].

When the scale of the coefficient matrix is very large, the efficiency of the classic Kaczmarz method significantly decreases as it selects rows in a sequential manner to approximate the solution of the system of equations. In 2009, Strohmer and Vershynin [20] introduced the randomized Kaczmarz (RK) method with expected exponential rate of convergence for solving overdetermined consistent linear systems, reigniting interest in Kaczmarz methods. They refined the row selection strategy by proposing to select the working row with probability $\Pr(\text{row} = i_k) = \|A^{(i_k)}\|_2^2/\|A\|_F^2$, resulting in a substantial acceleration of its convergence rate. Subsequently, Bai and Wu [21] proposed a greedy randomized Kaczmarz (GRK) method, which introduced a greedy probability criterion to obtain the larger component of the module of the residual vector in each iteration, so that it would be eliminated first in the iteration process, and thus accelerate the convergence rate. It is worth noting that Ansorge [22] proposed a maximal residual Kaczmarz (MRK) method, and Popa analyzed it in [23]. Similar to the working row obtained by the GRK method, this method selects the target working row index i_k so that the i_k-th component of the residual has relatively the largest absolute value compared to other components, i.e., $i_k = \arg\max_{1\le i\le m} |b^{(i)} - A^{(i)}x_k|$. Based on the GRK method, Bai and Wu [24] proposed a more efficient method, named the relaxed greedy randomized Kaczmarz (RGRK) method, by introducing a relaxation factor. Zhang [25] developed a new greedy Kaczmarz method, and Bai and Wu proved a more precise convergence upper bound for the randomized Kaczmarz method in [26]. For further study of the randomized Kaczmarz method and its variants, see [27–31].

For iterative solutions of large linear equations, in order to further accelerate the convergence rate of the Kaczmarz method, it is a natural idea to use block iteration instead of single-row iteration, so the block Kaczmarz method emerges as the times require. Bai and Liu proved the convergence of the block Kaczmarz method in [17]. Needell et al. pointed out in [32] that the matrix has good row paving, introduced a block strategy that depends on matrix eigenvalues, and proposed the first block Kaczmarz method with (expected) linear convergence to solve the overdetermined least squares problems, which projected the current iterative solution vector onto the solution space of the constrained subsets at each iteration step. To be specific, if the subset $\mathcal{J}_k$ is selected at the k-th iteration, let $A_{\mathcal{J}_k}^{\dagger}$ denote the Moore-Penrose pseudo-inverse of $A_{\mathcal{J}_k}$, and the iteration formula for x_k can be expressed as:

$$x_{k+1} = x_k + A_{\mathcal{J}_k}^{\dagger}(b_{\mathcal{J}_k} - A_{\mathcal{J}_k}x_k)$$

with $A_{\mathcal{J}_k} = A(\mathcal{J}_k, :)$ and $b_{\mathcal{J}_k} = b(\mathcal{J}_k)$. Needell et al. then proposed a randomized block Kaczmarz (RBK) method for solving the least squares problem in [33]. As a natural follow-up to the RBK

method, Liu and Gu [34] proposed the greedy randomized block Kaczmarz (GRBK) method for solving consistent linear systems. To avoid the high computational cost of pseudo-inverse calculations during block iterations, Necoara [35] proposed the randomized average block Kaczmarz (RABK) method, which projects the current iteration vector onto each row of the selected submatrix and averages them using different extrapolated step sizes. Subsequently, based on the idea of the average block method, Miao and Wu [36] extended the GRBK method to propose the greedy randomized average block Kaczmarz (GRABK) method, while Li et al. introduced the greedy average block Kaczmarz (GABK) method in [37]. Recently, Xiao et al. [38] combined the greedy strategy with the average block method to develop the fast greedy randomized block Kaczmarz method. For more research on the block Kaczmarz method, refer to [39–43].

In this paper, combining the row selection strategy in the MRK method [22] with the row partitioning strategy in the RBK method [33], we construct the maximum residual block Kaczmarz (denoted as MRBK) method, which is designed to prioritize the elimination of the largest block within the residual vector r_k during every iteration. Simultaneously, in order to avoid the pseudo-inverse calculation of the MRBK method during iteration, we further develop the maximum residual average block Kaczmarz (denoted as MRABK) method, which completes each iteration by projecting the current solution x_k onto each row of $A_{\mathcal{V}_{i_k}}$ and applying different extrapolation step sizes to average them. We give the deterministic convergence analysis of these two methods. Numerical experiments show that the MRABK method outperforms the MRBK method, while both methods are more efficient than the GRK, MRK, RBK, and GRBK methods.

The structure of this paper is outlined as follows. We introduce the maximum residual block Kaczmarz method and prove its convergence in Section 2. In Section 3, we establish the maximum residual average block Kaczmarz method and analyze its convergence. Section 4 features numerical experiments that validate the effectiveness of the proposed methods. Lastly, Section 5 ends the paper with conclusions.

Some basic assumptions: For a real matrix $A \in \mathbb{R}^{m\times n}$, $\|A\|_2$, $\|A\|_F$, and $A^\dagger$ signify the Euclidean norm, Frobenius norm, and Moore-Penrose pseudo-inverse of matrix A, respectively. We define A as a standardized matrix if each row has a Euclidean norm of 1, meaning $\|A^{(i)}\|_2 = 1$, for all $i = 1, 2, \ldots, m$. Similarly, for a given vector u, $\|u\|_2$ also represents its Euclidean norm. The notation $\sigma_{min}(A)$ and $\sigma_{max}(A)$ are employed to express the smallest nonzero and largest singular values of matrix A, respectively. Additionally, let us define the set $[m]$ as $\{1, 2, ..., m\}$, where m is an arbitrary positive integer. We consider the collection $\mathcal{V} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_t\}$ to be a partition of $[m]$ such that the index sets $\mathcal{V}_i$ (for $i = 1, 2, \ldots, t$) satisfy the conditions $\mathcal{V}_i \cap \mathcal{V}_j = \emptyset$ for $i \neq j$, and $\cup_{i=1}^{t} \mathcal{V}_i = [m]$. Furthermore, given a specified row index set $\mathcal{V}_i$, we represent the row submatrix of matrix A that corresponds to $\mathcal{V}_i$ as $A_{\mathcal{V}_i}$, and we denote the subvector of vector b as $b_{\mathcal{V}_i}$. The identity matrix of appropriate dimensions is denoted by I. Define the randomized partition of $[m]$ as $\mathcal{V} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_t\}$ with

$$\mathcal{V}_i = \{\pi(k) : k = \lfloor (i-1)m/t \rfloor + 1, \lfloor (i-1)m/t \rfloor + 2, \ldots, \lfloor im/t \rfloor\} \tag{1.2}$$

where $i = 1, 2, \ldots, t$, and we assume that the row partition anywhere else in this paper is as shown in (1.2).

2. Maximum residual block Kaczmarz method

In this section, drawing inspiration from the ideas behind the MRK [22], RBK [33], and GRBK [34] methods, we are going to construct the maximum residual block Kaczmarz (MRBK) method and analyze its convergence property.

There are typically two approaches to accelerate the Kaczmarz method: the first approach focuses on selecting working rows more efficiently to achieve faster convergence in each iteration, while the second approach aims to utilize row block iteration instead of single-row iteration for acceleration. Building upon these approaches, we propose the MRBK method as follows as Method 1. First, we partition the rows of matrix A to obtain the row block division $\mathcal{V}$ of A, i.e., $\{A_{\mathcal{V}_1}, A_{\mathcal{V}_2}, \ldots, A_{\mathcal{V}_t}\}$. Next, we denote $r_k^{(i)}$ as the ith block component corresponding to the residual vector r_k, and then $r_k^{(i)} = b_{\mathcal{V}_i} - A_{\mathcal{V}_i} x_k$, where $i = 1, 2, \ldots, t$. We select the working row block $\mathcal{V}_{i_k}$ according to $i_k = \arg\max_{1\le i\le t} \|b_{\mathcal{V}_i} - A_{\mathcal{V}_i} x_k\|_2^2$, ensuring that the largest block component of the residual vector r_k is eliminated first in each iteration. This significantly improves the convergence rate.

Next, we try to analyze the differences and improvements of the MRBK method compared to the other three methods:

(1) The MRBK method versus the MRK method: The MRBK method accelerates the MRK method naturally by utilizing row block iterations instead of single-row iterations.

(2) The MRBK method versus the RBK and GRBK methods: In comparison to the RBK method, the GRBK method improves the RBK method by introducing a new greedy probability criterion to randomly select the index of the row blocks, which ensures that row blocks with large residual values are prioritized for elimination, thus accelerating the RBK method. However, the GRBK method requires constructing the index set of row blocks and then selecting them based on probability in each iteration. In contrast, our proposed MRBK method selects the row blocks with the largest residual directly, enhancing the iteration efficiency. In fact, along with the idea of "greedy", our method can be called "extremely greedy" when it comes to selecting the index of the row blocks.

Method 1 The MRBK Method

Input: A, b, ℓ, x_0.
Output: x_ℓ
1: Let $\{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_t\}$ be a partition of $[m]$
2: **for** $k = 0, 1, \ldots, \ell - 1$ **do**
3: Select $i_k = \arg\max_{1\le i\le t} \|b_{\mathcal{V}_i} - A_{\mathcal{V}_i} x_k\|_2^2$
4: Set $x_{k+1} = x_k + A_{\mathcal{V}_{i_k}}^{\dagger} \left(b_{\mathcal{V}_{i_k}} - A_{\mathcal{V}_{i_k}} x_k\right)$
5: **end for**

Definition 2.1. *[33] (Row paving) A row paving $A(t, \alpha, \beta)$ of an $m \times n$ matrix A is defined as a partition $\mathcal{V} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_t\}$ of the rows such that for each $\mathcal{V}_i \in \mathcal{V}$, the following inequalities hold:*

$$\alpha \le \sigma_{min}^2(A_{\mathcal{V}_i}) \quad \text{and} \quad \sigma_{max}^2(A_{\mathcal{V}_i}) \le \beta,$$

where t is referred to as the size of the paving, while α and β are known as the lower and upper paving bounds, respectively.

For the MRBK method, Theorem 2.1 is given to illustrate its convergence.

Theorem 2.1. *Assuming that the linear system* (1.1) *is consistent, for a fixed partition* $\mathcal{V} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_t\}$ *of* $[m]$*, we assume that* $\sigma^2_{max}(A_{\mathcal{V}_i}) \leq \beta$ *for each* $\mathcal{V}_i \in \mathcal{V} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_t\}$*, starting from any initial vector* $x_0 \in \mathcal{R}(A^T)$*, the iteration sequence* $\{x_k\}_{k=0}^{\infty}$ *generated by the MRBK method, converges to the unique least-norm solution* $x_\star = A^\dagger b$*. Moreover, for any* $k \geq 0$*, we have*

$$\|x_1 - x_\star\|_2^2 \leq \left(1 - \frac{\sigma^2_{min}(A)}{\beta t}\right)\|x_0 - x_\star\|_2^2 \tag{2.1}$$

and

$$\|x_{k+1} - x_\star\|_2^2 \leq \left(1 - \frac{\sigma^2_{min}(A)}{\beta(t-1)}\right)^k \left(1 - \frac{\sigma^2_{min}(A)}{\beta t}\right)\|x_0 - x_\star\|_2^2. \tag{2.2}$$

Proof. From the definition of the MRBK method, for a partition $\mathcal{V} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_t\}$, $k = 0, 1, 2, \ldots$ and $i_k \in \{1, 2, \ldots, t\}$, we have

$$x_{k+1} - x_\star = x_k - x_\star + A^\dagger_{\mathcal{V}_{i_k}}\left(b_{\mathcal{V}_{i_k}} - A_{\mathcal{V}_{i_k}} x_k\right).$$

Since $A_{\mathcal{V}_{i_k}} x_\star = b_{\mathcal{V}_{i_k}}$, it holds that

$$x_{k+1} - x_\star = x_k - x_\star - A^\dagger_{\mathcal{V}_{i_k}} A_{\mathcal{V}_{i_k}}(x_k - x_\star).$$

Since $A^\dagger_{\mathcal{V}_{i_k}} A_{\mathcal{V}_{i_k}}$ is an orthogonal projector, we can derive the following relation using the Pythagorean Theorem:

$$\|x_{k+1} - x_\star\|_2^2 = \|x_k - x_\star\|_2^2 - \|A^\dagger_{\mathcal{V}_{i_k}} A_{\mathcal{V}_{i_k}}(x_k - x_\star)\|_2^2. \tag{2.3}$$

We note that

$$\begin{aligned}
\|A^\dagger_{\mathcal{V}_{i_k}} A_{\mathcal{V}_{i_k}}(x_k - x_\star)\|_2^2 &\geq \sigma^2_{min}(A^\dagger_{\mathcal{V}_{i_k}})\|A_{\mathcal{V}_{i_k}}(x_k - x_\star)\|_2^2 \\
&= \frac{1}{\sigma^2_{max}(A_{\mathcal{V}_{i_k}})}\|A_{\mathcal{V}_{i_k}}(x_k - x_\star)\|_2^2 \\
&\geq \frac{1}{\beta}\|A_{\mathcal{V}_{i_k}}(x_k - x_\star)\|_2^2 \\
&= \frac{1}{\beta}\|b_{\mathcal{V}_{i_k}} - A_{\mathcal{V}_{i_k}} x_k\|_2^2 \\
&= \frac{1}{\beta}\max_{1\leq i\leq t}\|b_{\mathcal{V}_i} - A_{\mathcal{V}_i} x_k\|_2^2.
\end{aligned} \tag{2.4}$$

Furthermore, from Method 1, we have

$$\begin{aligned} b_{\mathcal{V}_{i_k}} - A_{\mathcal{V}_{i_k}} x_{k+1} &= b_{\mathcal{V}_{i_k}} - A_{\mathcal{V}_{i_k}} \left(x_k + A_{\mathcal{V}_{i_k}}^{\dagger} (b_{\mathcal{V}_i} - A_{\mathcal{V}_i} x_k) \right) \\ &= b_{\mathcal{V}_{i_k}} - A_{\mathcal{V}_{i_k}} x_k - A_{\mathcal{V}_{i_k}} A_{\mathcal{V}_{i_k}}^{\dagger} \left(b_{\mathcal{V}_{i_k}} - A_{\mathcal{V}_{i_k}} x_k \right) \\ &= b_{\mathcal{V}_{i_k}} - A_{\mathcal{V}_{i_k}} x_k - A_{\mathcal{V}_{i_k}} A_{\mathcal{V}_{i_k}}^{\dagger} b_{\mathcal{V}_{i_k}} + A_{\mathcal{V}_{i_k}} A_{\mathcal{V}_{i_k}}^{\dagger} A_{\mathcal{V}_{i_k}} x_k \\ &= b_{\mathcal{V}_{i_k}} - A_{\mathcal{V}_{i_k}} A_{\mathcal{V}_{i_k}}^{\dagger} b_{\mathcal{V}_{i_k}} - A_{\mathcal{V}_{i_k}} x_k + A_{\mathcal{V}_{i_k}} x_k \\ &= A_{\mathcal{V}_{i_k}} x_\star - A_{\mathcal{V}_{i_k}} A_{\mathcal{V}_{i_k}}^{\dagger} A_{\mathcal{V}_{i_k}} x_\star \\ &= A_{\mathcal{V}_{i_k}} x_\star - A_{\mathcal{V}_{i_k}} x_\star \\ &= 0. \end{aligned}$$

Thus we obtain, for $k = 1, 2, \ldots$, that

$$\|b - Ax_k\|_2^2 = \sum_{\mathcal{V}_{i_k} \in \mathcal{V} \setminus \mathcal{V}_{i_{k-1}}} \|b_{\mathcal{V}_{i_k}} - A_{\mathcal{V}_{i_k}} x_k\|_2^2 \leq (t-1) \max_{1 \leq i \leq t} \|b_{\mathcal{V}_i} - A_{\mathcal{V}_i} x_k\|_2^2,$$

and hence

$$\max_{1 \leq i \leq t} \|b_{\mathcal{V}_i} - A_{\mathcal{V}_i} x_k\|_2^2 \geq \frac{1}{t-1} \|b - Ax_k\|_2^2. \tag{2.5}$$

Combining (2.3), (2.4), and (2.5), for $k = 1, 2, \ldots$, we finally have

$$\begin{aligned} \|x_{k+1} - x_\star\|_2^2 &\leq \|x_k - x_\star\|_2^2 - \frac{1}{\beta} \frac{\|b - Ax_k\|_2^2}{t-1} \\ &= \|x_k - x_\star\|_2^2 - \frac{1}{\beta} \frac{\|A(x_k - x_\star)\|_2^2}{t-1} \\ &\leq \|x_k - x_\star\|_2^2 - \frac{\sigma_{min}^2(A)}{\beta(t-1)} \|x_k - x_\star\|_2^2 \\ &= (1 - \frac{\sigma_{min}^2(A)}{\beta(t-1)}) \|x_k - x_\star\|_2^2. \end{aligned}$$

In particular, when $k = 0$, we also obtain

$$\|b - Ax_0\|_2^2 \leq t \max_{1 \leq i \leq t} \|b_{\mathcal{V}_i} - A_{\mathcal{V}_i} x_0\|_2^2.$$

Thus, we have

$$\|x_1 - x_\star\|_2^2 \leq \left(1 - \frac{\sigma_{min}^2(A)}{\beta t} \right) \|x_0 - x_\star\|_2^2.$$

□

Subsequently, we aim to conduct a more in-depth analysis of the convergence factors associated with the MRBK, RBK, and GRBK methods. The convergence factors of these three methods are denoted as ρ_{MRBK}, ρ_{RBK}, and ρ_{GRBK}, respectively. Based on Theorem 2.1, we derive

$$\rho_{MRBK} = 1 - \frac{\sigma_{min}^2(A)}{\beta(t-1)}, \tag{2.6}$$

and from Theorem 3.1 in [34], we can get

$$\rho_{GRBK} = 1 - \frac{\zeta}{2}\left(\frac{\|A\|_F^2}{\|A\|_F^2 + \zeta} + 1\right)\frac{\sigma_{min}^2(A)}{\beta\|A\|_F^2}, \tag{2.7}$$

where $\zeta = \min\limits_{\mathcal{V}_i \in \mathcal{V}} \|A_{\mathcal{V}_i}\|_F^2$.

Assuming that the matrix A is a standardized matrix, we can obtain from Theorem 2.1 in [33] that

$$\rho_{RBK} = 1 - \frac{\sigma_{min}^2(A)}{\beta m}.$$

We observe that $\rho_{MRBK} < \rho_{RBK}$ as long as $t - 1 < m$. In order to compare ρ_{MRBK} and ρ_{GRBK}, we consider rewriting ρ_{MRBK}:

$$\rho_{MRBK} = 1 - \frac{\sigma_{min}^2(A)}{\beta(t-1)} = 1 - \frac{\|A\|_F^2}{t-1}\frac{\sigma_{min}^2(A)}{\beta\|A\|_F^2}. \tag{2.8}$$

For the row paving $\{A_{\mathcal{V}_1}, A_{\mathcal{V}_2}, \ldots, A_{\mathcal{V}_t}\}$ of standardized matrix A, we assume that every cardinality of $A_{\mathcal{V}_i}$ is equal, i.e., equal to $\lfloor \frac{m}{t} \rfloor$, where $\lfloor \cdot \rfloor$ means to round down to an integer. Thus

$$\frac{\|A\|_F^2}{t-1} = \frac{m}{t-1}, \tag{2.9}$$

and

$$\frac{\zeta}{2}\left(\frac{\|A\|_F^2}{\|A\|_F^2 + \zeta} + 1\right) = \frac{\lfloor \frac{m}{t} \rfloor}{2}\left(\frac{t}{t+1} + 1\right) \leq \frac{m}{2t}\left(\frac{t}{t+1} + 1\right) = \frac{1}{2}\left(\frac{m}{t+1} + \frac{m}{t}\right). \tag{2.10}$$

Substitute (2.9) into (2.8), and (2.10) into (2.7), and it can be concluded that $\rho_{MRBK} < \rho_{GRBK}$.

Remark 2.1. *Based on the preceding analysis, it is evident that when the dimensions of matrix A are substantial, the convergence factors of both the MRBK and GRBK methods exhibit a close resemblance, indicating that their iterative steps are similarly aligned. However, owing to the lower computational cost associated with selecting working row block $A_{\mathcal{V}_{i_k}}$ in the MRBK method, it demonstrates a faster convergence rate in practical applications. The numerical experiments in Section 4 verify our inference.*

3. Maximum residual average block Kaczmarz method

In this section, we consider further improvements to the MRBK method. In step 4 of Method 1, each update of x_k requires the pseudo-inverse of $A_{\mathcal{V}_{i_k}}$ to be applied to the vector, which is computationally expensive when the size of matrix A is very large. We develop the maximum residual average block Kaczmarz (MRABK) method by projecting x_k onto each row of $A_{\mathcal{V}_{i_k}}$ and averaging them with different extrapolation steps, avoiding the computation of the pseudo-inverse and greatly saving the computational cost at each iteration, as shown in Method 2.

For the MRABK method, Theorem 3.1 is given to illustrate its convergence.

Method 2 The MRABK Method

Input: A, b, ℓ, $\omega \in (0,2)$ and x_0.

Output: x_ℓ

1: Let $\{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_t\}$ be a partition of $[m]$

2: **for** $k = 0, 1, \ldots, \ell - 1$ **do**

3: Select $i_k = \arg\max_{1\le i\le t} \|b_{\mathcal{V}_i} - A_{\mathcal{V}_i} x_k\|_2^2$

4: Compute $\alpha_k = \omega \frac{\|b_{\mathcal{V}_{i_k}} - A_{\mathcal{V}_{i_k}} x_k\|_2^2 \|A_{\mathcal{V}_{i_k}}\|_F^2}{\|A_{\mathcal{V}_{i_k}}^T (b_{\mathcal{V}_{i_k}} - A_{\mathcal{V}_{i_k}} x_k)\|_2^2}$

5: Set $x_{k+1} = x_k + \alpha_k \frac{A_{\mathcal{V}_{i_k}}^T (b_{\mathcal{V}_{i_k}} - A_{\mathcal{V}_{i_k}} x_k)}{\|A_{\mathcal{V}_{i_k}}\|_F^2}$

6: **end for**

Theorem 3.1. *Assuming that the linear system* (1.1) *is consistent, for a fixed partition* $\mathcal{V} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_t\}$ *of* $[m]$ *and* $\omega \in (0,2)$*, we assume that* $\sigma_{max}^2(A_{\mathcal{V}_i}) \le \beta$ *for each* $\mathcal{V}_i \in \mathcal{V} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_t\}$*, starting from any initial vector* $x_0 \in \mathcal{R}(A^T)$*, the iteration sequence* $\{x_k\}_{k=0}^{\infty}$ *generated by the MRABK method, converges to the unique least-norm solution* $x_\star = A^\dagger b$*. Moreover, for any* $k \ge 0$*, we have*

$$\|x_1 - x_\star\|_2^2 \le \left(1 - (2\omega - \omega^2)\frac{\sigma_{min}^2(A)}{\beta t}\right)\|x_0 - x_\star\|_2^2 \tag{3.1}$$

and

$$\|x_{k+1} - x_\star\|_2^2 \le \left(1 - (2\omega - \omega^2)\frac{\sigma_{min}^2(A)}{\beta(t-1)}\right)^k \left(1 - (2\omega - \omega^2)\frac{\sigma_{min}^2(A)}{\beta t}\right)\|x_0 - x_\star\|_2^2. \tag{3.2}$$

Proof. From step 5 of Method 2 and $A_{\mathcal{V}_{i_k}} x_\star = b_{\mathcal{V}_{i_k}}$, we can get

$$\begin{aligned} x_{k+1} - x_\star &= x_k - x_\star + \alpha_k \frac{A_{\mathcal{V}_{i_k}}^T (b_{\mathcal{V}_{i_k}} - A_{\mathcal{V}_{i_k}} x_k)}{\|A_{\mathcal{V}_{i_k}}\|_F^2} \\ &= x_k - x_\star - \alpha_k \frac{A_{\mathcal{V}_{i_k}}^T A_{\mathcal{V}_{i_k}} (x_k - x_\star)}{\|A_{\mathcal{V}_{i_k}}\|_F^2} \\ &= \left(I - \alpha_k \frac{A_{\mathcal{V}_{i_k}}^T A_{\mathcal{V}_{i_k}}}{\|A_{\mathcal{V}_{i_k}}\|_F^2}\right)(x_k - x_\star). \end{aligned}$$

By squaring the Euclidean norm on both sides of the above equation, it holds that

$$\begin{aligned} \|x_{k+1} - x_\star\|_2^2 &= \left\| \left(I - \alpha_k \frac{A_{\mathcal{V}_{i_k}}^T A_{\mathcal{V}_{i_k}}}{\|A_{\mathcal{V}_{i_k}}\|_F^2}\right)(x_k - x_\star)\right\|_2^2 \\ &= \|x_k - x_\star\|_2^2 - 2\alpha_k \frac{\|A_{\mathcal{V}_{i_k}}(x_k - x_\star)\|_2^2}{\|A_{\mathcal{V}_{i_k}}\|_F^2} + \alpha_k^2 \frac{\|A_{\mathcal{V}_{i_k}}^T A_{\mathcal{V}_{i_k}}(x_k - x_\star)\|_2^2}{\|A_{\mathcal{V}_{i_k}}\|_F^4}. \end{aligned}$$

Substituting α_k into this equality, we have

$$\begin{aligned}
\|x_{k+1} - x_\star\|_2^2 &= \|x_k - x_\star\|_2^2 - (2\omega - \omega^2)\frac{\|b_{\mathcal{V}_{i_k}} - A_{\mathcal{V}_{i_k}} x_k\|_2^4}{\|A_{\mathcal{V}_{i_k}}^T (b_{\mathcal{V}_{i_k}} - A_{\mathcal{V}_{i_k}} x_k)\|_2^2} \\
&\leq \|x_k - x_\star\|_2^2 - (2\omega - \omega^2)\frac{\|b_{\mathcal{V}_{i_k}} - A_{\mathcal{V}_{i_k}} x_k\|_2^4}{\sigma_{max}^2(A_{\mathcal{V}_{i_k}})\|(b_{\mathcal{V}_{i_k}} - A_{\mathcal{V}_{i_k}} x_k)\|_2^2} \\
&\leq \|x_k - x_\star\|_2^2 - (2\omega - \omega^2)\frac{\|b_{\mathcal{V}_{i_k}} - A_{\mathcal{V}_{i_k}} x_k\|_2^2}{\beta} \\
&= \|x_k - x_\star\|_2^2 - (2\omega - \omega^2)\frac{\max\limits_{1\leq i\leq t} \|b_{\mathcal{V}_i} - A_{\mathcal{V}_i} x_k\|_2^2}{\beta}.
\end{aligned}$$

We see that the first of these inequalities is true since $2\omega - \omega^2 > 0$ for $\omega \in (0, 2)$, and

$$\|A_{\mathcal{V}_{i_k}}^T (b_{\mathcal{V}_{i_k}} - A_{\mathcal{V}_{i_k}} x_k)\|_2^2 \leq \sigma_{max}^2(A_{\mathcal{V}_{i_k}})\|(b_{\mathcal{V}_{i_k}} - A_{\mathcal{V}_{i_k}} x_k)\|_2^2.$$

From (2.5) in the proof of Theorem 2.1, we can obtain, for $k = 1, 2, \ldots$, that

$$\begin{aligned}
\|x_{k+1} - x_\star\|_2^2 &\leq \|x_k - x_\star\|_2^2 - (2\omega - \omega^2)\frac{\|b - Ax_k\|_2^2}{\beta(t-1)} \\
&\leq \|x_k - x_\star\|_2^2 - (2\omega - \omega^2)\frac{\sigma_{min}^2(A)\|(x_k - x_\star)\|_2^2}{\beta(t-1)} \\
&= \left(1 - (2\omega - \omega^2)\frac{\sigma_{min}^2(A)}{\beta(t-1)}\right)\|x_k - x_\star\|_2^2.
\end{aligned}$$

In particular, when $k = 0$, we have

$$\|x_1 - x_\star\|_2^2 \leq \left(1 - (2\omega - \omega^2)\frac{\sigma_{min}^2(A)}{\beta t}\right)\|x_0 - x_\star\|_2^2.$$

□

From (3.2), we can obtain

$$\rho_{MRABK} = 1 - (2\omega - \omega^2)\frac{\sigma_{min}^2(A)}{\beta(t-1)}, \tag{3.3}$$

and it is not difficult to find that when $\omega = 1$, ρ_{MRABK} reaches its minimum value, which is $1 - \frac{\sigma_{min}^2(A)}{\beta(t-1)}$, and at the same time, $\rho_{MRABK} = \rho_{MRBK}$. However, as can be seen from steps 4 and 5 of Method 2, the average block method is adopted in the MRABK method, which avoids the pseudo-inverse calculation in the x_k update process in step 4 of the MRBK method. Therefore, the convergence speed of the MRABK method may be faster than that of the MRBK method. The numerical experiment results in Section 4 verify our conclusions well in the convergence rate.

4. Experimental results

In this section, the efficiency of the MRBK and MRABK methods is verified through numerical experiments. We test and compare our proposed two methods with the GRK, MRK, RBK, RABK [35], and GRBK methods. In each iteration of the RBK, GRBK, and MRBK methods, we utilize CGLS [44] to solve linear problems. Additionally, we use the same randomized row partition $\{A_{\mathcal{V}_1}, A_{\mathcal{V}_2}, \ldots, A_{\mathcal{V}_t}\}$ defined as (1.2) for the RBK, RABK, GRBK, MRBK, and MRABK methods, and for the selection of the number of blocks, [36] proves that $\lceil \|A\|_2^2 \rceil$ is a good choice, so we set the number of blocks $t = \lceil \|A\|_2^2 \rceil$ uniformly, where $\lceil \cdot \rceil$ means to round up to an integer. Specifically, in the case of the MRABK method, ω is set to 1 to ensure that the convergence factor of MRABK is minimized. Similarly, in the RABK method, ω is also set to 1. The effectiveness of the aforementioned method is assessed based on the quantity of iterative steps (referred to as "IT") and the computing time taken in seconds (referred to as "CPU"). Both IT and CPU indicate the arithmetic mean of the iterative steps and CPU duration needed to perform the process 20 times for every method. To illustrate the effectiveness of our proposed methods, we also compute the speed-up values of the MRBK method in comparison to both the MRK and GRBK methods, as well as the speed-up value of the MRABK method relative to the MRBK method. The definitions for these speed-up values are provided below:

$$\mathrm{SU}_1 = \frac{\text{CPU of MRK}}{\text{CPU of MRBK}},$$

$$\mathrm{SU}_2 = \frac{\text{CPU of GRBK}}{\text{CPU of MRBK}},$$

$$\mathrm{SU}_3 = \frac{\text{CPU of MRBK}}{\text{CPU of MRABK}}.$$

All experiments are conducted on a personal computer equipped with MATLAB (version R2022a), featuring an AMD Ryzen 7 8845H w/ Radeon(TM) 780M Graphics CPU clocked at 3.80 GHz, alongside 32.00 GB of RAM and running on the Windows 11 operating system.

For consistent linear systems (1.1), the coefficient matrix is either given by the MATLAB function sprandn or taken from the SuiteSparse Matrix Collection [45]. For the coefficient matrix A being tested, any zero row vectors are removed and A is normalized to the standard matrix. The right vector is set to $b = Ax_*$, where vector x_* is the solution vector generated using the MATLAB function randn. All calculations start with the initial zero vector $x_0 = 0$ and stop once the relative solution error (RSE) at the current iteration meets the criterion of RSE $< 10^{-6}$ or when IT surpasses the set limit of 200,000, defined as

$$\mathrm{RSE} = \frac{\|x_k - x_\star\|_2^2}{\|x_\star\|_2^2},$$

where the minimum norm solution $x_\star$ is obtained using the MATLAB function lsqminnorm.

Define the density of a matrix as

$$\text{density} = \frac{\text{number of nonzeros of an } m\text{-by-}n \text{ matrix}}{mn}.$$

For the first kind of sparse matrix A, we set its sparse parameter is 0.01, i.e., A = sprandn (m, n, 0.01), in Tables 1 and 2, the IT and CPU for the GRK, MRK, RBK, RABK, GRBK, MRBK, and MRABK methods are listed, and we also list the speed-up values for several methods.

Table 1. Numerical results for m-by-n random matrices A with $m = 6000$ and different n.

$m \times n$		6000×1000	6000×1500	6000×2000	6000×2500	6000×3000
$\|\|A\|\|_2^2$		12.29	9.13	7.64	6.63	5.86
GRK	IT	2325.8	5217.6	10418.6	18855.0	34941.8
	CPU	0.5867	1.8596	4.9190	12.7430	28.2539
MRK	IT	2230.0	5115.0	10297.0	18647.0	34598.0
	CPU	0.1538	0.5342	1.4035	5.2019	11.9263
RBK	IT	31.4	41.4	55.6	81.6	113.8
	CPU	0.0339	0.1514	0.1013	0.1851	0.2908
RABK	IT	55.6	69.2	87.8	123.4	177.4
	CPU	0.0118	0.0191	0.0332	0.0540	0.0912
GRBK	IT	25.2	29.8	37.0	52.4	72.8
	CPU	0.0486	0.0773	0.1340	0.2639	0.3927
MRBK	IT	**22.0**	**29.0**	**36.0**	**51.0**	**69.0**
	CPU	0.0239	0.0319	0.0590	0.1035	0.1391
MRABK	IT	40.0	50.0	62.0	79.0	104.0
	CPU	**0.0085**	**0.0160**	**0.0260**	**0.0410**	**0.0599**
SU_1		6.44	16.75	23.79	50.28	85.74
SU_2		2.04	2.42	2.27	2.55	2.82
SU_3		2.82	1.99	2.27	2.52	2.32

From Table 1, we can find that when $m = 6000$ and $n = 1000$, 1500, 2000, 2500, or 3000, both the MRBK method and MRABK method proposed by us outperform other methods in terms of CPU time. Now we focus on the MRK, GRBK, MRBK, and MRABK methods. We observe that the MRBK method, as a block-improved version of the MRK method, is significantly more efficient in terms of IT and CPU time. The SU_1 value of these two methods is at least 6.44 and the maximal is 85.74. We analyzed the convergence factors of the GRBK and MRBK methods in Section 2, and concluded that their IT should be very close when the size of A is large. From Table 1, we can observe that the IT of these two methods is very close, and the IT of the MRBK method is slightly smaller than that of the GRBK method. The SU_2 value is at least 2.04 and at most 2.82. The MRABK method has the shortest CPU time among all the above methods, and its SU_3 value relative to the MRBK method is at least 1.99 and up to 2.82.

From Table 2, we can find that when $n = 6000$ and $m = 1000$, 1500, 2000, 2500, or 3000, the MRBK and MRABK methods are still superior to other methods. Among all the above methods, the MRBK method has the fewest IT, while the MRABK method has the shortest CPU time. We note that under these conditions, the SU_1 value is at least 75.35 and the maximum is 155.46. Even though the IT of the MRBK and GRBK methods are almost the same, the CPU time of the MRBK method is better than that of the GRBK method. Compared with the MRBK method, the SU_3 value of the MRABK method is at least 1.99 and at most 2.89.

Table 2. Numerical results for m-by-n random matrices A with $n = 6000$ and different m.

$m \times n$		1000×6000	1500×6000	2000×6000	2500×6000	3000×6000
$\|\|A\|\|_2^2$		2.01	2.24	2.50	2.69	2.93
GRK	IT	4057.8	8544.0	14833.6	27124.0	43610.2
	CPU	2.4823	6.4250	13.0586	27.2911	48.0314
MRK	IT	4051.0	8324.0	14761.0	27112.0	42903.0
	CPU	1.5622	3.6929	7.3926	14.9410	24.4223
RBK	IT	19.2	43.4	58.2	81.2	113.8
	CPU	0.0396	0.0954	0.1552	0.2670	0.2885
RABK	IT	26.2	59.4	58.6	102.2	140.0
	CPU	0.0120	0.0322	0.0452	0.0713	0.1010
GRBK	IT	10.0	22.0	29.4	40.6	55.6
	CPU	0.0303	0.0779	0.1245	0.2487	0.3010
MRBK	IT	**10.0**	**21.0**	**28.0**	**40.0**	**55.0**
	CPU	0.0207	0.0482	0.0748	0.1277	0.1571
MRABK	IT	19.0	32.0	40.0	55.0	73.0
	CPU	**0.0104**	**0.0198**	**0.0281**	**0.0441**	**0.0638**
SU_1		75.35	76.58	98.88	116.96	155.46
SU_2		1.46	1.62	1.67	1.95	1.92
SU_3		1.99	2.44	2.66	2.89	2.46

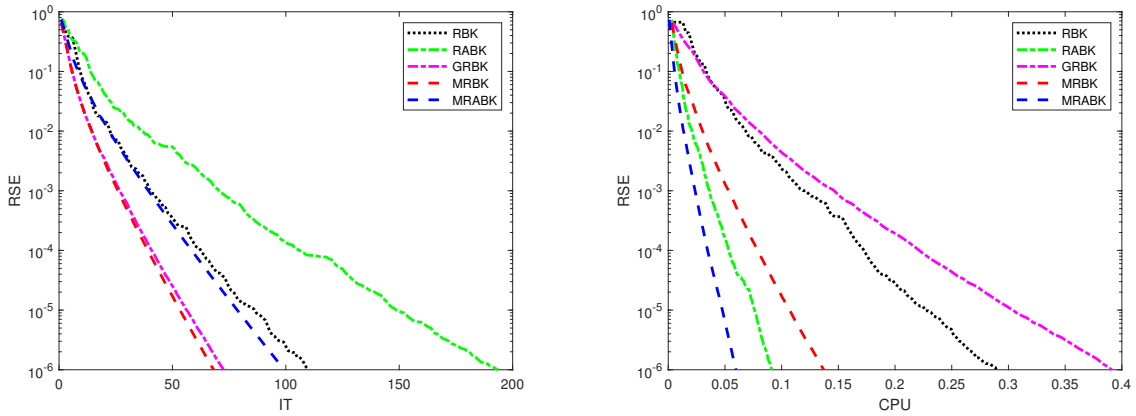
The second type of coefficient matrices selected from the SuiteSparse Matrix Collection [45] are derived from different applications, such as the combinatorial problem and linear programming problem. These matrices possess some unique structures and characteristics, such as being thin ($m > n$) (e.g., Franz9, GL7d12), fat ($m < n$) (e.g., p6000, lp_80bau3b), or square ($m = n$) (e.g., Trefethen_700), and we list details about them in Table 3, including their size, density, and cond(A). For these matrices, we implement the GRK, MRK, RBK, GRBK, RABK, MRBK, and MRABK methods, and list the IT and CPU time for each method in Table 3.

In Table 3, we observe that the MRABK method still maintains the shortest CPU time. When the coefficient matrix is GL7d12 and lp_80bau3b, the MRBK method has the smallest IT. When the coefficient matrix is Trefethen_700 or p6000, the IT is the same as that of the GRBK method. Compared to the MRK method, the MRBK method exhibits a minimal SU_1 value of 1.60 and a maximal SU_1 value of 60.55. In comparison to the GRBK method, the MRBK method demonstrates a minimal SU_2 value of 1.19 and a maximal SU_2 value of 2.29. Furthermore, compared to the MRBK method, the MRABK method showcases a minimal SU_3 value of 1.09, reaching a maximal SU_3 value of 3.33.

Table 3. Numerical results for matrices A from the SuiteSparse Matrix Collection.

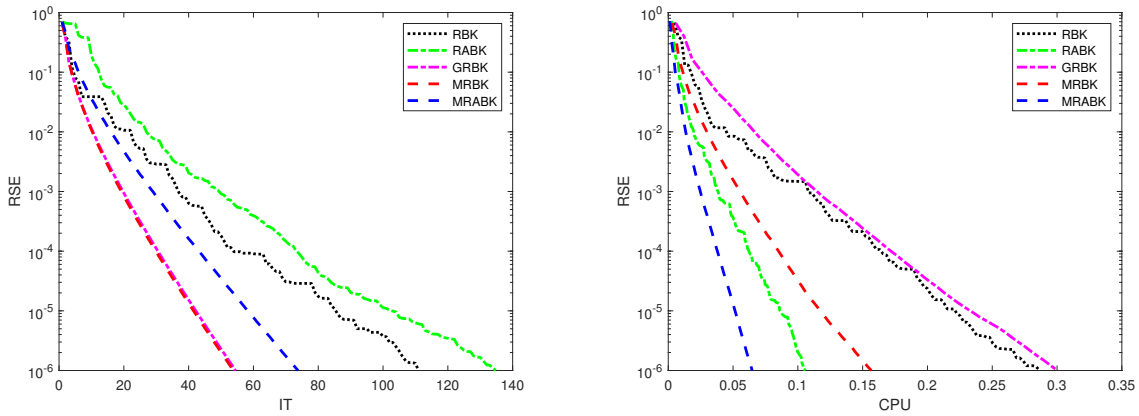
name		Franz9	GL7d12	Trefethen_700	p6000	lp_80bau3b
$m \times n$		19588×4164	8899×1019	700×700	2095×7967	2262×12061
density		0.12%	0.41%	2.58%	0.12%	0.09%
cond(A)		1.46e+16	Inf	4.71e+03	6.50e+05	567.23
$\|\|A\|\|_2^2$		60.80	55.75	2.54	2.35	2.82
GRK	IT	9056.8	2394.0	1555.2	4676.8	15874.6
	CPU	5.5755	0.6003	0.0969	1.2815	6.3763
MRK	IT	2307.0	2492.0	1536.0	4387.0	15924.0
	CPU	1.6704	0.1403	0.0326	0.8933	4.2627
RBK	IT	382.6	722.6	37.2	34.6	77.8
	CPU	0.6476	0.5484	0.0180	0.0292	0.2153
RABK	IT	1351.6	877.8	221.6	35.6	202.0
	CPU	0.1695	0.1576	0.0132	0.0071	0.0467
GRBK	IT	**124.8**	145.3	10.0	18.0	35.0
	CPU	0.5942	0.2011	0.0055	0.0193	0.1097
MRBK	IT	129.0	**121.0**	**10.0**	**18.0**	**34.0**
	CPU	0.3076	0.0879	0.0044	0.0148	0.0921
MRABK	IT	545.0	176.0	75.0	26.0	126.0
	CPU	**0.1662**	**0.0264**	**0.0040**	**0.0052**	**0.0301**
SU_1		5.43	1.60	7.38	60.55	46.29
SU_2		1.93	2.29	1.25	1.31	1.19
SU_3		1.85	3.33	1.09	2.83	3.06

To further assess the performance of various block Kaczmarz methods, Figures 1 and 2 display the relationship between the RSE and the IT, as well as the relationship between the RSE and CPU time for different block Kaczmarz methods. As we can see from Figures 1 and 2, with the increase of the IT, the RSE of the MRBK and MRABK methods both decrease faster than the RBK and RABK methods, and the MRBK method has the fastest decline in RSE among all block Kaczmarz methods. With the increase of the CPU time, the RSE of the MRABK method decreases the fastest, followed by the RABK and MRBK methods. Both of the MRBK and MRABK methods are ahead of the RBK and GRBK methods.



(a) (b)

Figure 1. (a) RSE versus IT and (b) RSE versus CPU for different block Kaczmrarz methods when the coefficient matrices are the 6000×3000 matrix in Table 1.



(a) (b)

Figure 2. (a) RSE versus IT and (b) RSE versus CPU for different block Kaczmrarz methods when the coefficient matrices are the 3000×6000 matrix in Table 2.

5. Conclusions

This paper presents two new Kaczmarz (MRBK and MRABK) methods for solving consistent linear systems. Both methods utilize uniform randomized partition of the rows of matrix A to construct the row subsets $\{A_{\mathcal{V}_1}, A_{\mathcal{V}_2}, \ldots, A_{\mathcal{V}_t}\}$. In each iteration, the MRBK method updates the solution vector by projecting x onto the hyperplane defined by $A_{\mathcal{V}_{i_k}} x = b_{\mathcal{V}_{i_k}}$, where $\mathcal{V}_{i_k}$ is selected according to $i_k = \arg\max_{1 \le i \le t} \|b_{\mathcal{V}_i} - A_{\mathcal{V}_i} x_k\|_2^2$, ensuring that the block with the largest residual is eliminated first, leading to rapid convergence. Building upon the MRBK method, the MRABK method introduces an adaptive step size, eliminating the need for calculating the pseudo-inverse of the row subset $A_{\mathcal{V}_{i_k}}$ of A during the x update process, thereby enhancing the convergence speed. We provide a comprehensive analysis

of the convergence theory for these two methods and conduct numerical experiments to validate their effectiveness. Both the theoretical analysis and numerical results demonstrate the superiority of our proposed methods over other Kaczmarz methods, including the GRK, MRK, RBK, and GRBK methods. In addition, we have realized that some valuable topics deserve further study, such as fully considering the structure and properties of A, making more efficient row partition of A, and finding a better step size for the MRABK method.

Author contributions

Wen-Ning Sun: Conceptualization, methodology, method design and implementation, numerical experiments and visualization, writing original drafts, reviewing and editing drafts; Mei Qin: Methodology, reviewing, editing drafts. All authors have reviewed and approved the final version of the manuscript prior to its publication.

Conflict of interest

The authors declare no competing interests.

References

1. L.-P. Sun, Y.-M. Wei, J.-Y. Zhou, On an iterative method for solving the least squares problem of rank-deficient systems, *Int. J. Comput. Math.*, **92** (2014), 532–541. https://doi.org/10.1080/00207160.2014.900173
2. Z.-Z. Bai, C.-H. Jin, Column-decomposed relaxation methods for the overdetermined systems of linear equations, *Int. J. Appl. Math.*, **13** (2003) , 71–82.
3. S. Kaczmarz, Angenäherte Auflösung von systemen linearer gleichungen, *Bull. Int. Acad. Polon. Sci. Lett.*, **35** (1937), 355–357.
4. M. A. Brooks, *A survey of algebraic algorithms in computerized tomography*, Oshawa: University of Ontario Institute of Technology, 2010.
5. G. N. Hounsfield, Computerized transverse axial scanning (tomography): Part 1. Description of system, *Brit. J. Radiol.*, **46** (1973), 1016–1022. https://doi.org/10.1259/0007-1285-46-552-1016
6. G. T. Herman, *Fundamentals of computerized tomography: Image reconstruction from projections*, 2 Eds., London: Springer, 2009. https://doi.org/10.1007/978-1-84628-723-7
7. F. Natterer, *The mathematics of computerized tomography*, Philadelphia: SIAM Publications, 2001. https://doi.org/10.1137/1.9780898719284
8. G. T. Herman, R. Davidi, Image reconstruction from a small number of projections, *Inverse Probl.*, **24** (2008), 045011. https://doi.org/10.1088/0266-5611/24/4/045011
9. R. Gordon, R. Bender, G. T. Herman, Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography, *J. Theor. Biol.*, **29** (1970), 471–481. https://doi.org/10.1016/0022-5193(70)90109-8

10. G. T. Herman, L. B. Meyer, Algebraic reconstruction techniques can be made computationally efficient (positron emission tomography application), *IEEE. Trans. Med. Imaging*, **12** (1993), 600–609. https://doi.org/10.1109/42.241889

11. C. Byrne, A unified treatment of some iterative algorithms in signal processing and image reconstruction, *Inverse Probl.*, **20** (2004), 103. https://doi.org/10.1088/0266-5611/20/1/006

12. D. A. Lorenz, S. Wenger, F. Schöpfer, M. Magnor, A sparse Kaczmarz solver and a linearized Bregman method for online compressed sensing, In: *2014 IEEE International conference on image processing*, 2014, 1347–1351. https://doi.org/10.1109/ICIP.2014.7025269

13. F. Pasqualetti, R. Carli, F. Bullo, Distributed estimation via iterative projections with application to power network monitoring, *Automatica*, **48** (2012), 747–758. https://doi.org/10.1016/j.automatica.2012.02.025

14. J. M. Elble, N. V. Sahinidis, P. Vouzis, GPU computing with Kaczmarz's and other iterative algorithms for linear systems, *Parallel Comput.*, **36** (2010), 215–231. https://doi.org/10.1016/j.parco.2009.12.003

15. A. Galántai, *Projectors and projection methods*, New York: Springer, 2004. https://doi.org/10.1007/978-1-4419-9180-5

16. P. A. Knight, *Error analysis of stationary iteration and associated problems*, Manchester: University of Manchester, 1993.

17. Z.-Z. Bai, X.-G. Liu, On the Meany inequality with applications to convergence analysis of several row-action iteration methods, *Numer. Math.*, **124** (2013), 215–236. https://doi.org/10.1007/s00211-012-0512-6

18. A. Ma, D. Needell, A. Ramdas, Convergence properties of the randomized extended Gauss-Seidel and Kaczmarz methods, *SIAM J. Matrix Anal. Appl.*, **36** (2015), 1590–1604. https://doi.org/10.1137/15M1014425

19. L. Dai, T. B. Schön, On the exponential convergence of the Kaczmarz algorithm, *IEEE Signal Process. Lett.*, **22** (2015), 1571–1574. https://doi.org/10.1109/LSP.2015.2412253

20. T. Strohmer, R. Vershynin, A randomized Kaczmarz algorithm with exponential convergence, *J. Fourier Anal. Appl.*, **15** (2009), 262–278. https://doi.org/10.1007/s00041-008-9030-4

21. Z.-Z. Bai, W.-T. Wu, On greedy randomized Kaczmarz method for solving large sparse linear systems, *SIAM J. Sci. Comput.*, **40** (2018), A592–A606. https://doi.org/10.1137/17M1137747

22. R. Ansorge, Connections between the Cimmino-method and the Kaczmarz-method for the solution of singular and regular systems of equations, *Computing*, **33** (1984), 367–375. https://doi.org/10.1007/bf02242280

23. C. Popa, Convergence rates for Kaczmarz-type algorithms, *Numer. Algor.*, **79** (2018), 1–17. https://doi.org/10.1007/s11075-017-0425-7

24. Z.-Z. Bai, W.-T. Wu, On relaxed greedy randomized Kaczmarz methods for solving large sparse linear systems, *Appl. Math. Lett.*, **83** (2018), 21–26. https://doi.org/10.1016/j.aml.2018.03.008

25. J.-J. Zhang, A new greedy Kaczmarz algorithm for the solution of very large linear systems, *Appl. Math. Lett.*, **91** (2019), 207–212. https://doi.org/10.1016/j.aml.2018.12.022

26. Z.-Z. Bai, W.-T. Wu, On convergence rate of the randomized Kaczmarz method, *Linear Algebra Appl.*, **553** (2018), 252–269. https://doi.org/10.1016/j.laa.2018.05.009

27. Y. Jiang, G. Wu, L. Jiang, A semi-randomized Kaczmarz method with simple random sampling for large-scale linear systems, *Adv. Comput. Math.*, **49** (2023), 20. https://doi.org/10.1007/s10444-023-10018-2

28. Y. Zeng, D. Han, Y. Su, J. Xie, Randomized Kaczmarz method with adaptive stepsizes for inconsistent linear systems, *Numer. Algor.*, **94** (2023), 1403–1420. https://doi.org/10.1007/s11075-023-01540-x

29. Z.-Z. Bai, W.-T. Wu, Randomized Kaczmarz iteration methods: Algorithmic extensions and convergence theory, *Japan J. Indust. Appl. Math.*, **40** (2023), 1421–1443. https://doi.org/10.1007/s13160-023-00586-7

30. Z.-Z. Bai, L. Wang, On convergence rates of Kaczmarz-type methods with di erent selection rules of working rows, *Appl. Numer. Math.*, **186** (2023), 289–319. https://doi.org/10.1016/j.apnum.2023.01.013

31. X.-Z. Wang, M.-L. Che, Y.-M. Wei, Randomized Kaczmarz methods for tensor complementarity problems, *Comput. Optim. Appl.*, **82** (2022) , 595–615. https://doi.org/10.1007/s10589-022-00382-y

32. D. Needell, J. A. Tropp, Paved with good intentions: Analysis of a randomized block Kaczmarz method, *Linear Algebra Appl.*, **441** (2014), 199–221. https://doi.org/10.1016/j.laa.2012.12.022

33. D. Needell, R. Zhao, A. Zouzias, Randomized block Kaczmarz method with projection for solving least squares, *Linear Algebra Appl.*, **484** (2015), 322–343. https://doi.org/10.1016/j.laa.2015.06.027

34. Y. Liu, C.-Q. Gu, On greedy randomized block Kaczmarz method for consistent linear systems, *Linear Algebra Appl.*, **616** (2021), 178–200. https://doi.org/10.1016/j.laa.2021.01.024

35. I. Necoara, Faster randomized block Kaczmarz algorithms, *SIAM J. Matrix Anal. Appl.*, **40** (2019), 1425–1452. https://doi.org/10.1137/19M1251643

36. C.-Q. Miao, W.-T. Wu, On greedy randomized average block Kaczmarz method for solving large linear systems, *J. Comput. Appl. Math.*, **413** (2022), 114372. https://doi.org/10.1016/j.cam.2022.114372

37. W. Li, F. Yin, Y.-M. Liao, G.-X. Huang, A greedy average block Kaczmarz method for the large scaled consistent system of linear equations, *AIMS Mathematics*, **7** (2022), 6792–6806. https://doi.org/10.3934/math.2022378

38. A.-Q. Xiao, J.-F. Yin, N. Zheng, On fast greedy block Kaczmarz methods for solving large consistent linear systems, *Comput. Appl. Math.*, **42** (2023), 119. https://doi.org/10.1007/s40314-023-02232-x

39. J. Briskman, D. Needell, Block Kaczmarz method with inequalities, *J. Math. Imaging Vis.*, **52** (2015), 385–396. https://doi.org/10.1007/s10851-014-0539-7

40. Y. Zhang, H. Li, Block sampling Kaczmarz-Motzkin methods for consistent linear systems, *Calcolo*, **58** (2021), 39. https://doi.org/10.1007/s10092-021-00429-2

41. Y. Zhang, H. Li, Randomized block subsampling Kaczmarz-Motzkin method, *Linear Algebra Appl.*, **667** (2023), 133–150. https://doi.org/10.1016/j.laa.2023.03.003

42. R.-R. Li, H. Liu, On randomized partial block Kaczmarz method for solving huge linear algebraic systems, *Comput. Appl. Math.*, **41** (2022), 278. https://doi.org/10.1007/s40314-022-01978-0

43. J.-Q. Chen, Z.-D. Huang, On a fast deterministic block Kaczmarz method for solving large-scale linear systems, *Numer. Algor.*, **89** (2022), 1007–1029. https://doi.org/10.1007/s11075-021-01143-4

44. Å. Björck, *Numerical methods for least squares problems*, Philadelphia: SIAM Publications, 1996. https://doi.org/10.1137/1.9781611971484

45. S. P. Kolodziej, M. Aznaveh, M. Bullock, J. David, T. A. Davis, M. Henderson, et al., The SuiteSparse matrix collection website interface, *J. Open Source Softw.*, **4** (2019), 1244. https://doi.org/10.21105/joss.01244