AIMS Mathematics, 10(3): 6270–6290.
DOI: 10.3934/math.2025286
Received: 03 January 2025
Revised: 26 February 2025
Accepted: 07 March 2025
Published: 20 March 2025

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

Research article

The maximum residual block Kaczmarz algorithm based on feature selection

Ran-Ran Li[1] **and Hao Liu**[1,2,*]

[1] School of Mathematics, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China

[2] Shenzhen Research Institute, Nanjing University of Aeronautics and Astronautics, Shenzhen 518063, China

* **Correspondence:** Email: hliu@nuaa.edu.cn.

Abstract: Based on the K-means method, an effective block-row partitions algorithm was proposed in [1], which involves partitioning the rows of the coefficient matrix $A \in \mathbb{R}^{m\times n}$. However, with the increase of the size of the coefficient matrix, the time required for the partitioning process will increase significantly. To address this problem, we considered selecting features from the columns of the matrix A to obtain a low-rank matrix $\tilde{A} \in \mathbb{R}^{m\times d}$ $(d \ll n)$. Lasso is a regression analysis method for feature selection, which is simple and has excellent processing ability for high-dimensional data. In view of this, we first introduced a new criterion for selecting the projection block, and proposed the maximum residual block Kaczmarz algorithm. Then, we put forward the feature selection algorithm based on Lasso, and further presented a maximum residual block Kaczmarz algorithm based on feature selection. We analyzed the convergence of these algorithms and demonstrated their effectiveness through numerical results, while also verifying the performance of the proposed algorithms in image reconstruction.

Keywords: block Kaczmarz method; Lasso method; feature selection; convergence property
Mathematics Subject Classification: 15A06, 65F10, 65F20

1. Introduction

For decades, research on solving large linear systems $Ax = b$ with $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$ has maintained an abundant and enduring history. Among the various solving approaches, iterative methods stand out due to their outstanding performance in efficiency and stability [2–4], particularly in dealing with large linear systems. The Kaczmarz method [5] is a typical representative, which has been widely used in many fields, such as computerized tomography, signal processing, and image

reconstruction [6–9].

A randomized version of the Kaczmarz method was proposed in [10], known as the randomized Kaczmarz (RK) method, which randomly selects rows of the coefficient matrix instead of selecting them in a cyclic manner. It has been proven that this method possesses an expected linear convergence rate. In [11], a better upper bound on the convergence rate of the RK method was provided. To enhance the convergence speed, Bai and Wu presented an effective probability criterion for selecting the larger entries in the residual vector as much as possible at each iteration, based on which the greedy randomized Kaczmarz (GRK) method [12] was proposed. Additionally, they demonstrated that its convergence rate is significantly faster than that of the RK method. Based on the GRK method, the relaxed GRK method was introduced in [13], and a non-randomized greedy Kaczmarz method was presented in [14]. For more discussions on the RK method, see also [15–18] and the references therein. References [19] and [20] provided a recent comprehensive investigation and review of Kaczmarz-type methods. In addition, the RK method is applied to solve the tensor equations, as detailed in references [21–23].

In many calculation processes, the block Kaczmarz method [24] can be implemented more efficiently than the Kaczmarz method, which uses some rows of the coefficient matrix at each iteration step. In order to improve the convergence speed, Needell and Tropp presented the randomized block Kaczmarz method [25], which has an expected linear rate of convergence. Afterwards, Necoara developed the randomized average block Kaczmarz method [26], which can be deployed on distributed computing units. Miao and Wu proposed the greedy randomized average block Kaczmarz [27] method, which can be implemented in a distributed environment and also analyzed two kinds of extrapolated step sizes. For the block version of the RK method, see also [28–31] and the references therein. A partition strategy based on the K-means method was proposed in [1, 32], which used cosine distance to measure the similarity between row vectors of the coefficient matrix. Due to the fact that cosine distance measures both the distance and directional similarity between two vectors, this strategy often yields better partitioning results. However, when the size of the coefficient matrix is too large or even huge, the time required for partitioning will significantly increase. Therefore, we consider utilizing a feature selection method to obtain a low-rank matrix, thereby improving solving efficiency.

Lasso (least absolute shrinkage and selection operator) [33] is a regression analysis method that performs both variable selection and parameter estimation by shrinking coefficients with smaller absolute values to zero through the penalty term. Based on the idea of Lasso, we consider selecting features from the columns of coefficient matrix $A \in \mathbb{R}^{m\times n}$ and obtain the low-rank matrix $\tilde{A} \in \mathbb{R}^{m\times d}$ ($d \ll n$). Therefore, our objective is to solve the following problem:

$$\min_{\beta}\left\{\frac{1}{2}\|b - A\beta\|_2^2 + \lambda\|\beta\|_1\right\}, \tag{1.1}$$

where $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $\beta \in \mathbb{R}^n$, and $\lambda > 0$. By adjusting the value of λ, the sparsity of the vector β can be controlled, where non-zero entries correspond to the positions of selected features from coefficient matrix A. Let $\beta^{(j)}$ represent the j-th entry of β, and construct the index set of non-zero entries in β, denoted by

$$\mathcal{D} = \left\{\tilde{j}\middle|\beta^{(j)} \neq 0, \quad j = 1, 2, \cdots, n\right\}, \tag{1.2}$$

and the number of elements in $\mathcal{D}$ is d. The low-rank matrix $\tilde{A} \in \mathbb{R}^{m\times d}$ can be obtained, with its column indices given by $\mathcal{D}$. Then, we partition the row vectors of matrix $\tilde{A}$ instead of matrix A, which can

improve the efficiency of the partition process. This process of feature selection and partitioning is shown in Figure 1.

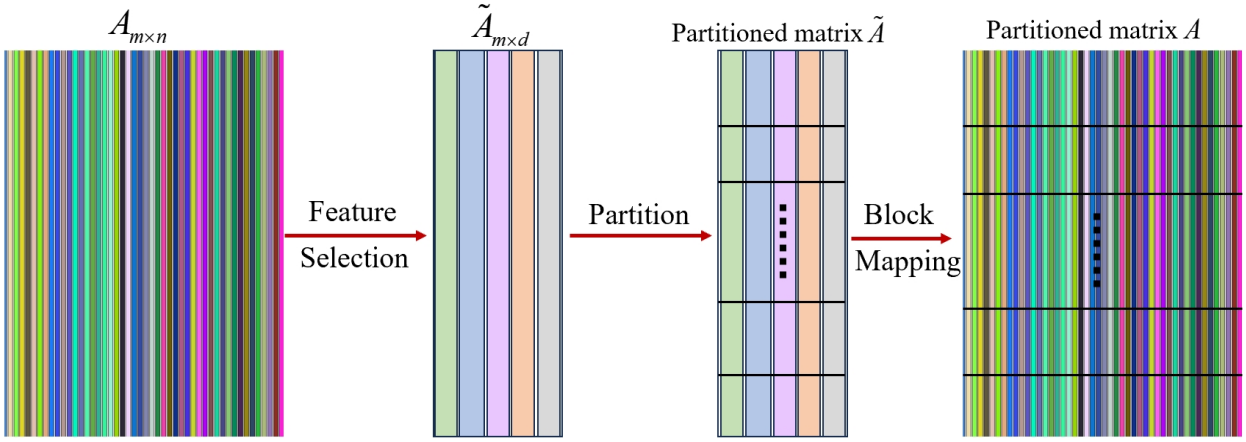


Figure 1. The process of feature selection and partitioning.

By partitioning the row vectors of matrix $\tilde{A}$, we obtain the block index set $\tilde{J} = [\tilde{J}_1, \cdots, \tilde{J}_q]$. This index set applies to the row vectors of the coefficient matrix A, which is divided into q blocks $A_{\tilde{J}_1}, \cdots, A_{\tilde{J}_q}$. Compared to directly partitioning the row vectors of A, partitioning those of $\tilde{A}$ can significantly reduce the time required for the partitioning process.

In this paper, we first introduce a new criterion for selecting the projection block, which involves this block containing the row with the maximum modulus of the residual vector, and propose the maximum residual block Kaczmarz algorithm. Then, we present the feature selection algorithm based on Lasso, and further put forward the maximum residual block Kaczmarz algorithm based on feature selection. The convergence of these algorithms is analyzed respectively, and numerical results demonstrate their effectiveness. In addition, we also verify the performance of both algorithms in image reconstruction.

2. Preliminaries and notation

For a matrix $A = \left(a_{ij}\right) \in \mathbb{R}^{m\times n}$, we use $A^\top$, $A^\dagger$, $\|A\|_2$, $A^{(i)}$, $\sigma_{\min}(A)$, and $\sigma_{\max}(A)$ to denote its transpose, Moore-Penrose pseudoinverse, spectral norm, i-th row, smallest nonzero value, and largest singular value, respectively. Let $[m] := \{1, 2, \cdots, m\}$, $J = \left\{J_1, J_2, \cdots, J_q\right\}$, and $J_v \cap J_s = \emptyset$, $v, s \in \{1, 2, \cdots q\}$, $\bigcup_{s=1}^{q} J_s = [m]$. A_{J_s} and $|J_s|$ represent the row submatrix indexed by J_s and the number of rows contained in block J_s. When generating test images, N, θ, and p denote the number of pixels, the scanning angle, and the number of X-ray beams, respectively. The cosine distance [1] between the n-dimensional vectors x_1 and x_2 is expressed as $d_c(x_1, x_2)$.

2.1. Soft-thresholding operator

The soft-thresholding operator [34] is a commonly used feature selection method that allows for screening and compression of features based on their importance, particularly in solving the Lasso estimator. By setting feature values below a certain threshold to zero and filtering out features that have minimal or redundant impact on the target variable, the dimensionality of the data is

effectively reduced. This approach reduces computational complexity and enhances the simplicity and effectiveness of the model. The soft-thresholding operator is defined as

$$\mathcal{S}_\lambda(x) = \operatorname{sign}(x)(|x| - \lambda)_+,$$

where $x \in \mathbb{R}^n$, t_+ denotes the positive part of $t \in \mathbb{R}$. It has a simple explicit expression, given by

$$\mathcal{S}_\lambda\left(x_j\right) = \begin{cases} x_j - \lambda, & x_j > \lambda, \\ 0, & \left|x_j\right| \le \lambda, \\ x_j + \lambda, & x_j < -\lambda. \end{cases}$$

2.2. The block-row partitions algorithm

K-means clustering is a powerful tool for exploratory data analysis and data compression, and it is particularly valuable for identifying natural groupings within the data. Based on the clustering idea of the K-means method, an efficient partitioning strategy was presented in [1]. The block-row partitions algorithm is as follows.

Algorithm 2.1. The block-row partitions algorithm.

Input: $A \in \mathbb{R}^{m\times n}$ and the number of the blocks $q\,(q < m)$

Output: $A_{J_1}, A_{J_2}, \cdots, A_{J_q}$

(1) Randomly select q rows of the matrix A as the initial block centers $\bar{x}_1, \bar{x}_2, \cdots \bar{x}_q$, where $\bar{x}_s\,(s = 1, \cdots, q) \in \mathbb{R}^{1\times n}$;

(2) **Repeat**

(3) Assign each row $A^{(i)}\,(i = 1, 2, \cdots, m)$ to the block which has the nearest block A_{J_l}, i.e.,

$$d_c^{(i)}\left(A^{(i)}, \bar{x}_l\right) = \min_s \left(d_c^{(i)}\left(A^{(i)}, \bar{x}_s\right)\right)(s = 1, 2, \cdots, q);$$

(4) Update the average value $\bar{x}_s = \sum_{A^{(i)}\in A_{J_s}} \frac{A^{(i)}}{|A_{J_s}|}$, where $|A_{J_s}|$ is the number of row vectors in the block J_s;

(5) Compute the criterion function: $C = \sum_{s=1}^{q} \sum_{x\in A_{J_s}} d_c(x, \bar{x}_s)^2$;

(6) **Until** C no longer changes, A_{J_s} is generated.

3. The maximum residual block Kaczmarz algorithm based on feature selection

3.1. The maximum residual block Kaczmarz algorithm

Based on the block-row partitions algorithm, we can obtain q sub-matrices (referred to as blocks) of the coefficient matrix A, i.e., $A_{J_1}, A_{J_2}, \cdots, A_{J_q}$. Thereafter, the selection of the projection block becomes our main focus. Inspired by the GRK method, when solving large linear systems, those terms with large modulus of the residual vector are given priority for the iteration to achieve faster convergence. However, in the block iteration method, the components of the residual vector in each block can be uneven, so determining the appropriate total residual metric to represent the entire block is

a complex task. Therefore, we introduce a criterion for selecting the projection block, which involves selecting the block containing the row with the maximum modulus of the residual vector.

At the k-th iterate, the row index of the maximum modulus of the residual vector is denoted as h_k, given by

$$h_k = \arg\max_{1\le i\le m} \left|b^{(i)} - A^{(i)}x_k\right|, \tag{3.1}$$

which satisfy $h_k \in J_{s_k}$. Afterward, the current iterate x_k is orthogonally projected onto the hyperplane defined by $A_{J_{s_k}}x_k = b_{J_{s_k}}$. For any initial vector x_0, the maximum residual block Kaczmarz (MBK) algorithm can be formulated as

$$x_{k+1} = x_k + A^{\dagger}_{J_{s_k}}\left(b_{J_{s_k}} - A_{J_{s_k}}x_k\right), \quad k = 0, 1, 2, \cdots, \tag{3.2}$$

where $h_k \in J_{s_k}$ is defined in (3.1). The maximum residual block Kaczmarz algorithm is given as follows.

Algorithm 3.1. The maximum residual block Kaczmarz algorithm: MBK.

Input: $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, l, $q\,(q < m)$, and x_0

Output: x_l

(1) The blocks A_{J_s}, b_{J_s} $(s = 1, 2, \cdots, q)$ are obtained by Algorithm 2.1, and satisfy $J_v \cap J_s = \emptyset$, $v, s \in \{1, 2, \cdots q\}$ and $\bigcup_{s=1}^{q} J_s = [m]$;

(2) **for** $k = 0, 1, \cdots, l-1$ **do**

(3) Compute $r_k = b - Ax_k$;

(4) Select the projection block $A_{J_{s_k}}$ satisfying

$$h_k = \arg\max_{1\le i\le m} \left|r_k^{(i)}\right|, \quad h_k \in J_{s_k};$$

(5) Set $x_{k+1} = x_k + A^{\dagger}_{J_{s_k}}\left(b_{J_{s_k}} - A_{J_{s_k}}x_k\right)$;

(6) **end for**

Theorem 3.1. *Let the large linear system $Ax = b$, with the coefficient matrix $A \in \mathbb{R}^{m\times n}$ and the right-hand side $b \in \mathbb{R}^m$, be consistent. The iteration sequence $\{x_k\}_{k=0}^{\infty}$ generated by the MBK algorithm starting from any initial vector $x_0 \in \mathcal{R}(A^{\top})$ converges to the unique least-norm solution $x_\star = A^{\dagger}b$. Moreover, the solution error for the iteration sequence $\{x_k\}_{k=0}^{\infty}$ obeys*

$$\|x_k - x_\star\|_2^2 \le \left(1 - \frac{\sigma_{\min}^2(A)}{\hat{\sigma}^2\left(m - \hat{N}_{\min}\right)}\right)^k \|x_0 - x_\star\|_2^2, \tag{3.3}$$

where $\hat{\sigma} = \max\limits_{J_{s_k}}\left\{\sigma_{\max}\left(A_{J_{s_k}}\right)\right\}$ and $\hat{N}_{\min} = \min\limits_{J_{s_{k-1}}}\left\{\left|J_{s_{k-1}}\right|\right\}$ (defining $\left|J_{s_{-1}}\right| = 0$).

Proof. From the definition of the maximum residual block Kaczmarz algorithm, for $k = 0, 1, 2, \cdots$, we have

$$x_{k+1} - x_k = A^{\dagger}_{J_{s_k}}A_{J_{s_k}}(x_\star - x_k).$$

Since $A_{J_{s_k}} x_\star = b_{J_{s_k}}$, we immediately get

$$x_{k+1} - x_\star = \left(I - A_{J_{s_k}}^\dagger A_{J_{s_k}}\right)(x_k - x_\star).$$

Based on the properties of the Moore-Penrose pseudoinverse, we can obtain

$$\begin{aligned}
&(x_\star - x_k)^\top \left(A_{J_{s_k}}^\dagger A_{J_{s_k}}\right)^\top \left(I - A_{J_{s_k}}^\dagger A_{J_{s_k}}\right)(x_k - x_\star) \\
&= (x_\star - x_k)^\top \left(A_{J_{s_k}}^\dagger A_{J_{s_k}} - A_{J_{s_k}}^\dagger A_{J_{s_k}} A_{J_{s_k}}^\dagger A_{J_{s_k}}\right)(x_k - x_\star) \\
&= 0,
\end{aligned}$$

and it follows that

$$\|x_{k+1} - x_\star\|_2^2 = \|x_k - x_\star\|_2^2 - \left\|A_{J_{s_k}}^\dagger A_{J_{s_k}} (x_\star - x_k)\right\|_2^2. \tag{3.4}$$

From the definition of the MBK algorithm, again, we know that

$$\left|b^{(h_k)} - A^{(h_k)} x_k\right|^2 = \max_{1 \le i \le m} \left|b^{(i)} - A^{(i)} x_k\right|^2, \quad h_k \in J_{s_k}, \tag{3.5}$$

and

$$\begin{aligned}
\|b - Ax_k\|_2^2 &= \sum_{u \in J_{s_{k-1}}} \left|b^{(u)} - A^{(u)} x_k\right|^2 + \sum_{v \in [m] \setminus J_{s_{k-1}}} \left|b^{(v)} - A^{(v)} x_k\right|^2 \\
&= \left\|b_{J_{s_{k-1}}} - A_{J_{s_{k-1}}} \left(x_{k-1} + A_{J_{s_{k-1}}}^\dagger \left(b_{J_{s_{k-1}}} - A_{J_{s_{k-1}}} x_{k-1}\right)\right)\right\|_2^2 + \sum_{v \in [m] \setminus J_{s_{k-1}}} \left|b^{(v)} - A^{(v)} x_k\right|^2 \\
&= \left\|A_{J_{s_{k-1}}} x_\star - A_{J_{s_{k-1}}} x_{k-1} - A_{J_{s_{k-1}}} A_{J_{s_{k-1}}}^\dagger A_{J_{s_{k-1}}} x_\star + A_{J_{s_{k-1}}} A_{J_{s_{k-1}}}^\dagger A_{J_{s_{k-1}}} x_{k-1}\right\|_2^2 \\
&+ \sum_{v \in [m] \setminus J_{s_{k-1}}} \left|b^{(v)} - A^{(v)} x_k\right|^2 \\
&= \sum_{v \in [m] \setminus J_{s_{k-1}}} \left|b^{(v)} - A^{(v)} x_k\right|^2 \\
&\le \left(m - \left|J_{s_{k-1}}\right|\right) \max_{v \in [m] \setminus J_{s_{k-1}}} \left\{\left|b^{(v)} - A^{(v)} x_k\right|^2\right\} \\
&\le \left(m - \left|J_{s_{k-1}}\right|\right) \left|b^{(h_k)} - A^{(h_k)} x_k\right|^2,
\end{aligned} \tag{3.6}$$

where $\left|J_{s_{k-1}}\right|$ represents the number of rows contained in block $J_{s_{k-1}}$ and $[m] \setminus J_{s_{k-1}}$ denotes the set difference between the set $[m]$ and $J_{s_{k-1}}$. Combining (3.4), (3.5), and (3.6), it leads to the inequality

$$
\begin{aligned}
\|x_{k+1}-x_\star\|_2^2 &= \|x_k-x_\star\|_2^2-\left\|A_{J_{s_k}}^{\dagger}A_{J_{s_k}}(x_\star-x_k)\right\|_2^2\\
&\le \|x_k-x_\star\|_2^2-\sigma_{\min}^2\left(A_{J_{s_k}}^{\dagger}\right)\sum_{i_k\in J_{s_k}}\left|A^{(i_k)}(x_\star-x_k)\right|^2\\
&\le \|x_k-x_\star\|_2^2-\sigma_{\min}^2\left(A_{J_{s_k}}^{\dagger}\right)\max_{i_k\in J_{s_k}}\left\{\left|b^{(i_k)}-A^{(i_k)}x_k\right|^2\right\}\\
&\le \|x_k-x_\star\|_2^2-\sigma_{\min}^2\left(A_{J_{s_k}}^{\dagger}\right)\left|b^{(h_k)}-A^{(h_k)}x_k\right|^2\\
&\le \|x_k-x_\star\|_2^2-\frac{\sigma_{\min}^2\left(A_{J_{s_k}}^{\dagger}\right)}{m-\left|J_{s_{k-1}}\right|}\|b-Ax_k\|_2^2\\
&\le \|x_k-x_\star\|_2^2-\frac{\sigma_{\min}^2(A)}{\sigma_{\max}^2\left(A_{J_{s_k}}\right)\left(m-\left|J_{s_{k-1}}\right|\right)}\|x_k-x_\star\|_2^2,
\end{aligned}
$$

where we have used the following estimate:

$$
\|Az\|_2^2\ge\sigma_{\min}^2(A)\|z\|_2^2,\quad z\in\mathcal{R}(A^{\top}).
$$

Therefore, for $k=0,1,2,\cdots$, we have

$$
\|x_{k+1}-x_\star\|_2^2\le\left(1-\frac{\sigma_{\min}^2(A)}{\hat{\sigma}^2\left(m-\hat{N}_{\min}\right)}\right)\|x_k-x_\star\|_2^2, \tag{3.7}
$$

where $\hat{\sigma}=\max\limits_{J_{s_k}}\left\{\sigma_{\max}\left(A_{J_{s_k}}\right)\right\}$ and $\hat{N}_{\min}=\min\limits_{J_{s_{k-1}}}\left\{\left|J_{s_{k-1}}\right|\right\}$. By induction on the iteration index k, the estimate (3.3) holds.

3.2. The maximum residual block Kaczmarz algorithm based on feature selection

The maximum residual block Kaczmarz algorithm introduces an effective criterion for determining the projection block, which contains the row with the maximum modulus of the residual vector. However, as the size of the coefficient matrix increases, the time required for the partitioning process will increase significantly. Therefore, by solving the problem (1.1), we can select features from the columns of the coefficient matrix $A\in\mathbb{R}^{m\times n}$, and obtain the low-rank matrix $\tilde{A}\in\mathbb{R}^{m\times d}$ $(d\ll n)$. Then, partitioning the row vectors of $\tilde{A}$ using Algorithm 2.1 can significantly reduce the required time.

For solving the problem (1.1), the proximal gradient descent method [35] is a straightforward and efficient computational approach, which is a variant of the gradient descent method and is often referred to as the iterative soft-thresholding method [36]. It iteratively updates the solution by combining gradient descent with the proximal operator, effectively handling non-smooth and regularized optimization problems. The proximal operator is defined as follows:

$$
\begin{aligned}
\mathrm{prox}_t(\beta) &= \arg\min_z\left\{\frac{1}{2t}\|\beta-z\|_2^2+\lambda\|z\|_1\right\}\\
&= \arg\min_z\left\{\|\beta-z\|+2\lambda t\|z\|_1\right\}\\
&= \mathcal{S}_{\lambda t}(\beta),
\end{aligned}
$$

where $\mathcal{S}_{\lambda t}(\beta)$ is the soft-thresholding operator. Choosing any initial approximation β_0 and t_0, the proximal gradient update of problem (1.1) is

$$\beta_{k+1} = \mathcal{S}_{\lambda t_k}\left(\beta_k - t_k \nabla g(\beta_k)\right), \quad k = 0, 1, 2, \cdots,$$

where $\nabla g(\beta_k) = -A^\top (b - A\beta_k)$, that is,

$$\beta_{k+1} = \mathcal{S}_{\lambda t_k}\left(\beta_k + t_k A^\top (b - A\beta_k)\right), \quad k = 0, 1, 2, \cdots, \tag{3.8}$$

where t_k is the step size [36].

For the value of the parameter λ, cross-validation can be used to select the optimal regularization parameter, but this requires an amount of computation. Our objective is not to obtain the optimal solution but to perform feature selection from the columns of the coefficient matrix A. An effective and practical approach is adopted where the penalty parameter λ is determined by rough estimation to ensure that the number of selected columns falls within a given range. This approach can help us quickly obtain feature selection results while maintaining a certain level of accuracy and reducing computational complexity. On this basis, the feature selection algorithm based on Lasso is as follows.

Algorithm 3.2. The feature selection algorithm based on Lasso.

Input: $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^n$, λ_{step}, $[\lambda_l, \lambda_r]$, $[d_l, d_r]$, t_0, β_0, max_Iter

Output: $\tilde{A} \in \mathbb{R}^{m\times d}$

(1) **for** $\lambda = \lambda_l : \lambda_{\text{step}} : \lambda_r$ **do**

(2) **for** $k = 0, 1, 2, \cdots,$ max_Iter **do**

(3) Update the step size $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$;

(4) Compute $\beta_{k+1} = \mathcal{S}_{\lambda\tilde{t}}\left(\beta_k + \frac{t_k - 1}{t_{k+1}} A^\top (b - A\beta_k)\right)$;

(5) **if** $\|\beta_{k+1} - \beta_k\|_2 < 10^{-6}$ **end if**

(6) **end for**

(7) Determine the index set $\mathcal{D} = \left\{ \tilde{j} \middle| \beta_k^{(j)} \neq 0,\ j = 1, 2, \cdots, n \right\}$, where the number of elements in $\mathcal{D}$ is d;

(8) **if** $d_l < d < d_r$ **end if**

(9) **end for**

(10) Construct the low-rank matrix $\tilde{A} \in \mathbb{R}^{m\times d}$, where the column index is denoted by $\mathcal{D}$.

Remark 3.1. At the k-th iterate, the constant step size is $\tilde{t} = \frac{t_k - 1}{t_{k+1}}$. The initial value is set to $t_0 = \frac{1}{\sigma_{\max}^2(A)}$, ensuring that $0 < \tilde{t} < \frac{1}{\sigma_{\max}^2(A)}$ always holds true.

Many studies [35–38] have analyzed the convergence of the proximal gradient descent method. Below is the convergence analysis of Algorithm 3.2 for solving the problem (1.1).

Theorem 3.2. *We are given the coefficient matrix $A \in \mathbb{R}^{m\times n}$ and the right-hand side $b \in \mathbb{R}^m$. The iteration sequence $\{\beta_k\}_{k=0}^{\infty}$ generated by the feature selection algorithm based on Lasso starting from*

any initial approximation β_0 converges to the optimal solution $\beta_\star$. Moreover, the solution error for the iteration sequence $\{\beta_k\}_{k=0}^{\infty}$ obeys

$$\|\beta_{k+1} - \beta_\star\|_2 \leq \left(1 - \tilde{t}\sigma_{\max}^2(A)\right) \|(\beta_k - \beta_\star)\|_2, \tag{3.9}$$

where $0 < \tilde{t} < \frac{1}{\sigma_{\max}^2(A)}$.

Proof. From the definition of Algorithm 3.2, we know that

$$\beta_{k+1} = \mathcal{S}_{\lambda\tilde{t}}\left(\beta_k + \tilde{t}A^\top (b - A\beta_k)\right), \quad k = 0, 1, 2, \cdots, \tag{3.10}$$

where $\tilde{t} = \frac{t_k - 1}{t_{k+1}}$ and $S_{\lambda\tilde{t}}(\cdot)$ is the soft-thresholding operator. For n-dimensional vectors β_1 and β_2, where $\beta_1^{(j)}$ and $\beta_2^{(j)}$ represent their j-th entry ($j = 1, 2, \cdots, n$), if they have the same sign, we have

$$\begin{aligned} \|S_{\lambda\tilde{t}}(\beta_1) - S_{\lambda\tilde{t}}(\beta_2)\|_2 &= \begin{cases} \left\|(\beta_1 - \lambda\tilde{t}) - (\beta_2 - \lambda\tilde{t})\right\|_2, & \beta_1^{(j)} > 0, \beta_2^{(j)} > 0, \\ \left\|(\beta_1 + \lambda\tilde{t}) - (\beta_2 + \lambda\tilde{t})\right\|_2, & \beta_1^{(j)} < 0, \beta_2^{(j)} < 0, \end{cases} \\ &= \|\beta_1 - \beta_2\|_2. \end{aligned}$$

On the contrary, if $\beta_1^{(j)}$ and $\beta_2^{(j)}$ have opposite signs, we have

$$\begin{aligned} \|S_{\lambda\tilde{t}}(\beta_1) - S_{\lambda\tilde{t}}(\beta_2)\|_2 &= \begin{cases} \left\|(\beta_1 - \lambda\tilde{t}) - (\beta_2 + \lambda\tilde{t})\right\|_2, & \beta_1^{(j)} > 0, \beta_2^{(j)} < 0, \\ \left\|(\beta_1 + \lambda\tilde{t}) - (\beta_2 - \lambda\tilde{t})\right\|_2, & \beta_1^{(j)} < 0, \beta_2^{(j)} > 0, \end{cases} \\ &= \begin{cases} \left\|\beta_1 - \beta_2 - 2\lambda\tilde{t}\right\|_2, & \beta_1^{(j)} > 0, \beta_2^{(j)} < 0, \\ \left\|\beta_1 - \beta_2 + 2\lambda\tilde{t}\right\|_2, & \beta_1^{(j)} < 0, \beta_2^{(j)} > 0, \end{cases} \\ &< \|\beta_1 - \beta_2\|_2. \end{aligned}$$

Thus, we can get the estimation

$$\|S_{\lambda\tilde{t}}(\beta_1) - S_{\lambda\tilde{t}}(\beta_2)\|_2 \leq \|\beta_1 - \beta_2\|_2. \tag{3.11}$$

Using (3.10) and (3.11), we obtain

$$\begin{aligned} \|\beta_{k+1} - \beta_\star\|_2 &= \left\|\mathcal{S}_{\lambda\tilde{t}}\left(\beta_k + \tilde{t}A^\top (b - A\beta_k)\right) - \mathcal{S}_{\lambda\tilde{t}}\left(\beta_\star + \tilde{t}A^\top (b - A\beta_\star)\right)\right\|_2 \\ &\leq \left\|\beta_k + \tilde{t}A^\top (b - A\beta_k) - \left(\beta_\star + \tilde{t}A^\top (b - A\beta_\star)\right)\right\|_2 \\ &\leq \left\|\left(I_n - \tilde{t}A^\top A\right)(\beta_k - \beta_\star)\right\|_2 \\ &\leq \left(1 - \tilde{t}\sigma_{\max}^2(A)\right) \|(\beta_k - \beta_\star)\|_2, \end{aligned}$$

where $0 < \tilde{t} < \frac{1}{\sigma_{\max}^2(A)}$.

Based on the idea of Lasso, we accomplish feature selection from the columns of the coefficient matrix A by employing Algorithm 3.2, which involves obtaining a low-rank matrix $\tilde{A}$. Subsequently, guided by the criterion outlined in (3.1) for selecting the projection block, we present the maximum residual block Kaczmarz (LMBK) algorithm based on feature selection. The specific implementation steps of LMBK are as follows.

Algorithm 3.3. The maximum residual block Kaczmarz algorithm based on feature selection: LMBK.

Input: $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^n$, $[\lambda_l, \lambda_r]$, $[d_l, d_r]$, t_0, β_0, max_Iter, l, q $(q < m)$, and x_0

Output: x_l

(1) The low-rank matrix $\tilde{A} \in \mathbb{R}^{m\times d}$ is obtained by Algorithm 3.2;

(2) Partition the rows of the matrix $\tilde{A}$ by Algorithm 2.1 to obtain the block set $\tilde{J} = \left[\tilde{J}_1, \cdots, \tilde{J}_q\right]$ satisfying $\tilde{J}_v \cap \tilde{J}_s = \emptyset$ and $\bigcup_{s=1}^{q} \tilde{J}_s = [m]$;

(3) **for** $k = 0, 1, \cdots, l-1$ **do**

(4) Compute $r_k = b - Ax_k$;

(5) Select the projection block $A_{\tilde{J}_{\tilde{s}_k}}$ satisfying

$$\tilde{h}_k = \arg\max_{1\le i\le m} \left|r_k^{(i)}\right|,\ \tilde{h}_k \in \tilde{J}_{\tilde{s}_k};$$

(6) Set $x_{k+1} = x_k + A^{\dagger}_{\tilde{J}_{\tilde{s}_k}} \left(b_{\tilde{J}_{\tilde{s}_k}} - A_{\tilde{J}_{\tilde{s}_k}} x_k\right)$;

(7) **end for**

Remark 3.2. In this algorithm, we first select features from the columns of the coefficient matrix A to obtain a low-rank matrix $\tilde{A}$, and then the rows of matrix $\tilde{A}$ are partitioned using Algorithm 2.1, which is notably faster than Step 1 in Algorithm 3.1.

Theorem 3.3. *Let the large linear system $Ax = b$, with the coefficient matrix $A \in \mathbb{C}^{m\times n}$ and the right-hand side $b \in \mathbb{C}^m$, be consistent. The iteration sequence $\{x_k\}_{k=0}^{\infty}$ generated by the LMBK algorithm starting from any initial approximation $x_0 \in \mathcal{R}(A^{\top})$ converges to the unique least-norm solution $x_\star = A^{\dagger}b$. Moreover, the solution error for the iteration sequence $\{x_k\}_{k=0}^{\infty}$ obeys*

$$\|x_k - x_\star\|_2^2 \le \left(1 - \frac{\sigma_{\min}^2(A)}{\tilde{\sigma}^2\left(m - \tilde{N}_{\min}\right)}\right)^k \|x_0 - x_\star\|_2^2, \tag{3.12}$$

where $\tilde{\sigma} = \max_{\tilde{J}_{\tilde{s}_k}} \left\{\sigma_{\max}\left(A_{\tilde{J}_{\tilde{s}_k}}\right)\right\}$ and $\tilde{N}_{\min} = \min_{\tilde{J}_{\tilde{s}_{k-1}}} \left\{\left|\tilde{J}_{\tilde{s}_{k-1}}\right|\right\}$ (defining $\left|\tilde{J}_{\tilde{s}_{-1}}\right| = 0$).

Proof. The difference between the MBK algorithm and the LMBK algorithm lies in the feature selection preprocessing of the coefficient matrix, which significantly reduces the time required for the partitioning process. For the convergence analysis of the LMBK algorithm, refer to the proof process of Theorem 3.1.

4. Numerical examples

In this section, we implement the greedy randomized Kaczmarz (GRK) method [12], the maximum residual block Kaczmarz (MBK) algorithm, and the maximum residual block Kaczmarz algorithm based on feature selection (LMBK), and demonstrate that the latter is numerically advantageous over the former in terms of the number of iteration steps (IT) and the computing time in seconds (CPU).

The coefficient matrix $A \in \mathbb{R}^{m\times n}$ is obtained from three different sources. The first class of matrices is generated randomly by the MATLAB function `randn`, the second class is selected from

the University of Florida Sparse Matrix Collection [39], and the third class is taken from problems related to image reconstruction. The right-hand side is $b = Ax_\star$, where $x_\star \in \mathbb{R}^n$ is generated by the function `randn`. Note that b is noise-free in our experiments. Let max_Iter =100, the initial vector $x_0 = \mathbf{0}, \beta_0 = \mathbf{0}$, and the relative residual (rres) is defined by

$$\text{rres} = \frac{\|b - Ax_k\|_2^2}{\|b\|_2^2}.$$

The iterations are stopped when x_k satisfies rres $\leq 10^{-4}$ or CPU exceeds 10,000 seconds.

All experiments are performed using MATLAB (version R2021a) on a personal computer with 3.10 GHz central processing unit (Intel(R) Core (TM) i5-11300H), 16 GB memory, and a Windows 11 operating system.

Example 4.1. The coefficient matrix $A \in \mathbb{R}^{m\times n}$ is randomly generated by using the MATLAB function `randn`. Let $\left[\lambda_l : \lambda_{step} : \lambda_r\right] = [1 : 0.5 : 10]$, $[d_l, d_r] = [2, 100]$, and the numerical results of GRK, MBK, and LMBK are as follows. Note that q is used for MBK and LMBK, whereas d is used for LMBK only.

From Tables 1 and 2, it is observed that the LMBK algorithm outperforms both the GRK and MBK algorithms in terms of IT and CPU. Furthermore, as the size of the coefficient matrix A increases, the GRK method fails to converge within the specified time, whereas the two proposed algorithms in this paper perform quite well. To visualize the performance of these three algorithms, we plot the convergence curves based on the aforementioned numerical results as follows.

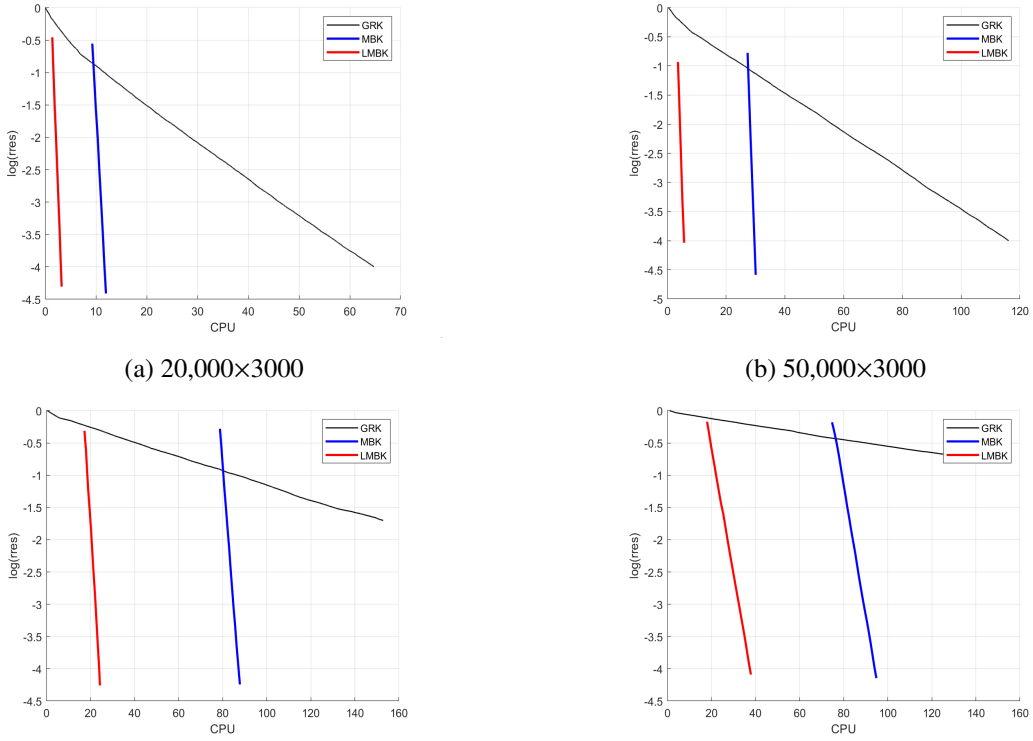
Table 1. IT and CPU of GRK, MBK, and LMBK for matrix A with $n = 3000$ and different m.

$m \times n$	q	d	GRK		MBK		LMBK	
			IT	CPU	IT	CPU	IT	CPU
20,000×3000	10	7	4154	64.1042	9	11.9749	9	**3.2472**
50,000×3000	20	9	3033	117.6046	6	30.0806	5	**5.7335**
80,000×3000	40	18	–	–	9	60.7425	9	**12.7522**
100,000×3000	60	17	–	–	12	77.6643	12	**18.8722**
200,000×3000	80	11	–	–	6	304.5365	6	**54.6815**

Table 2. IT and CPU of GRK, MBK, and LMBK for matrix A with $m = 80,000$ and different n.

$m \times n$	q	d	GRK		MBK		LMBK	
			IT	CPU	IT	CPU	IT	CPU
$m \times n$	q	d	GRK		MBK		LMBK	
			IT	CPU	IT	CPU	IT	CPU
80,000×2000	40	12	1710	123.3912	1	35.2884	1	**6.4711**
80,000×4000	40	22	–	–	14	87.8377	14	**24.3464**
80,000×6000	40	28	–	–	23	94.8711	23	**37.9381**
80,000×8000	40	3	–	–	32	150.6195	31	**39.0150**
80,000×10,000	40	42	–	–	42	238.9067	41	**73.1683**

Figure 2 intuitively demonstrates the sharp drop in convergence curves for both the MBK and LMBK algorithms, underscoring their high effectiveness. The key disparity between these two algorithms lies in the fact that the LMBK algorithm first selects features from the columns of the coefficient matrix, which greatly reduces the time for the partition process. The following figure illustrates the comparison of the time for the partition process (Pcpu) between these two algorithms, corresponding to the matrices outlined in Tables 1 and 2, respectively.

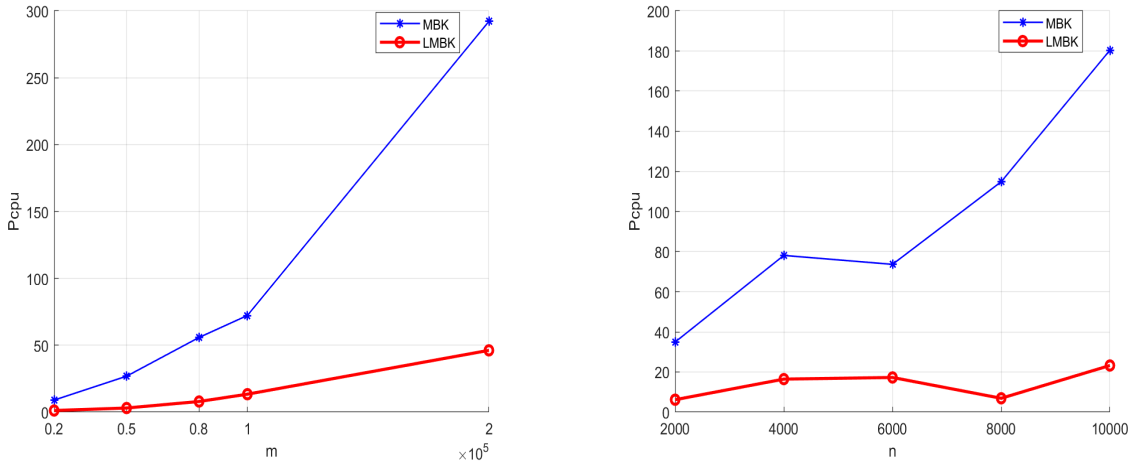


(a) 20,000×3000 (b) 50,000×3000

(c) 80,000×4000 (b) 80,000×6000

Figure 2. Convergence curves of GRK, MBK, and LMBK.

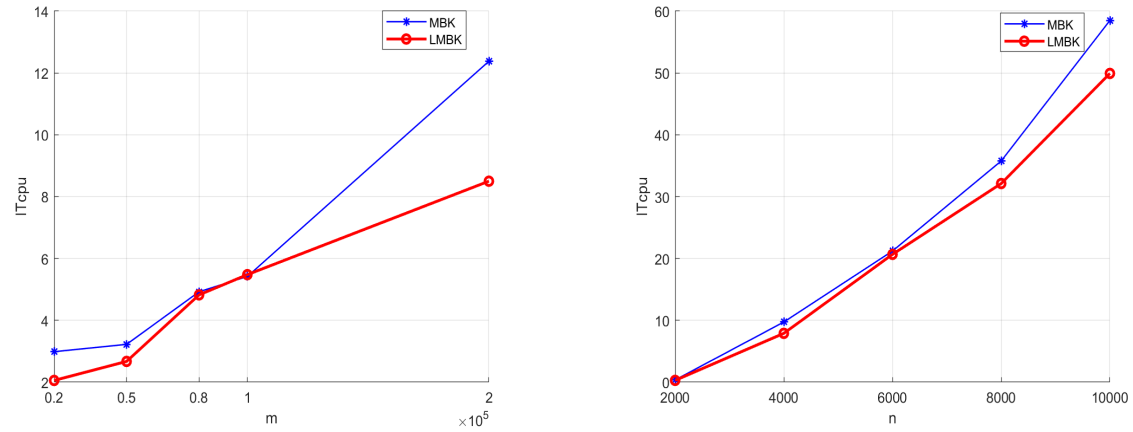
From Figure 3, it is evident that after the feature selection of the coefficient matrix using Algorithm 3.2, the time for the partition process is significantly reduced. Since the quality of the partitioning also directly affects the convergence rate, we provide the following figure to compare the time for the iteration process (ITcpu) for these two algorithms, corresponding to the matrices listed in Tables 1 and 2, respectively.



(a) Table 1 (b) Table 2

Figure 3. CPU for the partition process.

From Figure 4, it can be observed that the time for the iteration process of the MBK and LMBK algorithms are nearly the same, with LMBK being relatively smaller. Therefore, by employing Algorithm 3.2 for the feature selection of the coefficient matrix and subsequently partitioning its rows, the obtained partitioning result is reliable.



(a) Table 1 (b) Table 2

Figure 4. CPU for the iteration process.

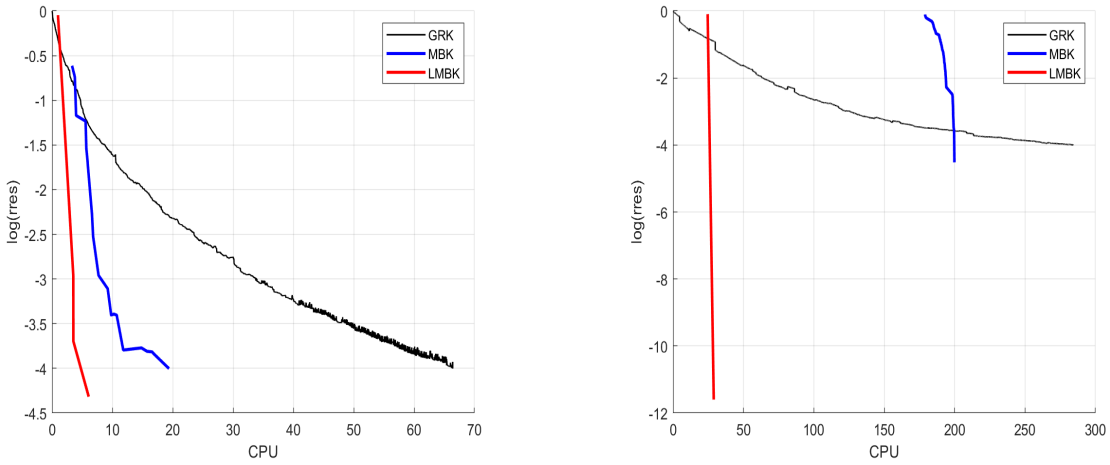
Example 4.2. The coefficient matrix $A \in \mathbb{R}^{m\times n}$ is selected from the University of Florida Sparse

Matrix Collection. Let $\left[\lambda_l : \lambda_{step} : \lambda_r\right] = \left[10^{-2} : 10^{-2} : 1\right]$, $[d_l, d_r] = [2, 100]$, and the numerical results of GRK, MBK, and LMBK are as in Table 3.

Table 3. IT and CPU of GRK, MBK, and LMBK.

name	$m \times n$	q	d	GRK		MBK		LMBK	
				IT	CPU	IT	CPU	IT	CPU
Franz10	19,588×4164	5	48	5574	3.3594	7	1.3927	1	**0.4788**
scsd8-2c$^\top$	35,910×5130	15	38	20,824	27.5643	10	5.0037	1	**1.1135**
scsd8-2r$^\top$	60,550×8650	25	20	33,384	66.8752	20	19.3880	10	**6.0809**
lp_nug20$^\top$	72,600× 15,240	30	38	17,779	68.4472	60	48.2827	5	**20.8793**
mesh_deform	234,023×9393	85	10	24,131	284.8606	60	199.8508	4	**28.9125**
fome20$^\top$	108,175× 33,874	30	41	–	–	148	1465.9682	10	**420.4182**

In Table 3, the GRK method fails for the matrix fome20$^\top$, and the IT and CPU of LMBK are considerably smaller than those of the GRK and MBK algorithms for all convergent cases. We also describe the convergence curves in Figure 5.



(a) scsd8-2r (b) mesh-deform

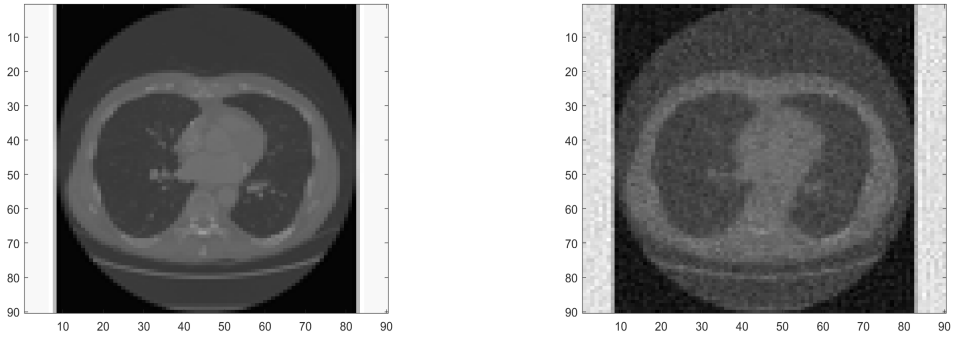
Figure 5. Convergence curves of GRK, MBK, and LMBK.

According to the display in Figure 5, the LMBK algorithm exhibits the fastest convergence speed among the three algorithms. In comparison, the convergence speed of the GRK method is significantly slower.

Example 4.3. In this example, the GRK, MBK, and LMBK algorithms are used to reconstruct two test images. The structural similarity index (SSIM) [40] is used as the metric to evaluate the similarity between the test images and the reconstructed images, which comprehensively considers information

regarding brightness, contrast, and structure, providing a score within the range of [-1, 1]. When SSIM = 1, it indicates that the two images are identical. Let $\left[\lambda_l : \lambda_{step} : \lambda_r\right] = \left[10^{-2} : 10^{-2} : 1\right]$ and $[d_l, d_r] = [2, 1000]$.

The first test image is obtained by loading human 3-D chest CT scan data into the MATLAB workspace, then extracting the central slice in the X-Y plane, which performs lung segmentation, as shown in subplot (a) of Figure 6. Let $N = 90$, $\theta = 0 : 1 : 179$, $p = \left\lfloor \sqrt{2}N \right\rfloor$, $q = 5$, and we can obtain the coefficient matrix A with a size of 22,860 × 8100 [30]. Then, the GRK, MBK, and LMBK algorithms are used to solve this linear system, and the corresponding reconstructed images are generated and displayed in the following figure.



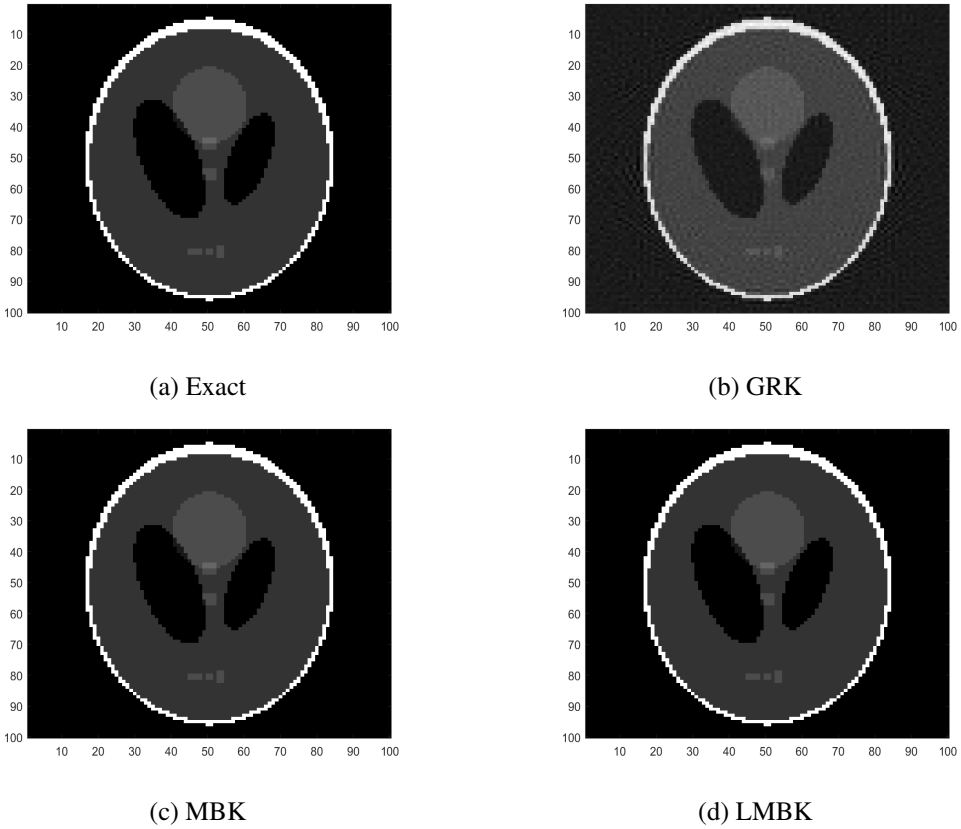
(a) Exact (b) GRK

(c) MBK (d) LMBK

Figure 6. Image reconstruction.

In Figure 6, it is observed that the corresponding subplot (d) for the LMBK algorithm is the clearest, followed by the MBK algorithm (see subplot (c)). The image for the GRK method (see subplot (b)) is relatively blurry.

The second test image is obtained by using the function `phantom` in the Image Processing Toolbox. Let $N = 100$, $\theta = 0 : 1 : 179$, $p = \left\lfloor \sqrt{2}N \right\rfloor$, $q = 15$, and the coefficient matrix A with a size of 25,380 × 10,000 can be obtained. The corresponding reconstructed images are generated and displayed in Figure 7.



(a) Exact (b) GRK

(c) MBK (d) LMBK

Figure 7. Image reconstruction.

Figure 7 clearly demonstrates that the reconstructed images obtained by the LMBK algorithm (see subplot (d)) are the clearest. To provide a more intuitive evaluation of image reconstruction achieved by the three algorithms, Tables 4 and 5 list the corresponding IT, CPU, and SSIM values for each algorithm.

Table 4. Test image 1 (d = 196).

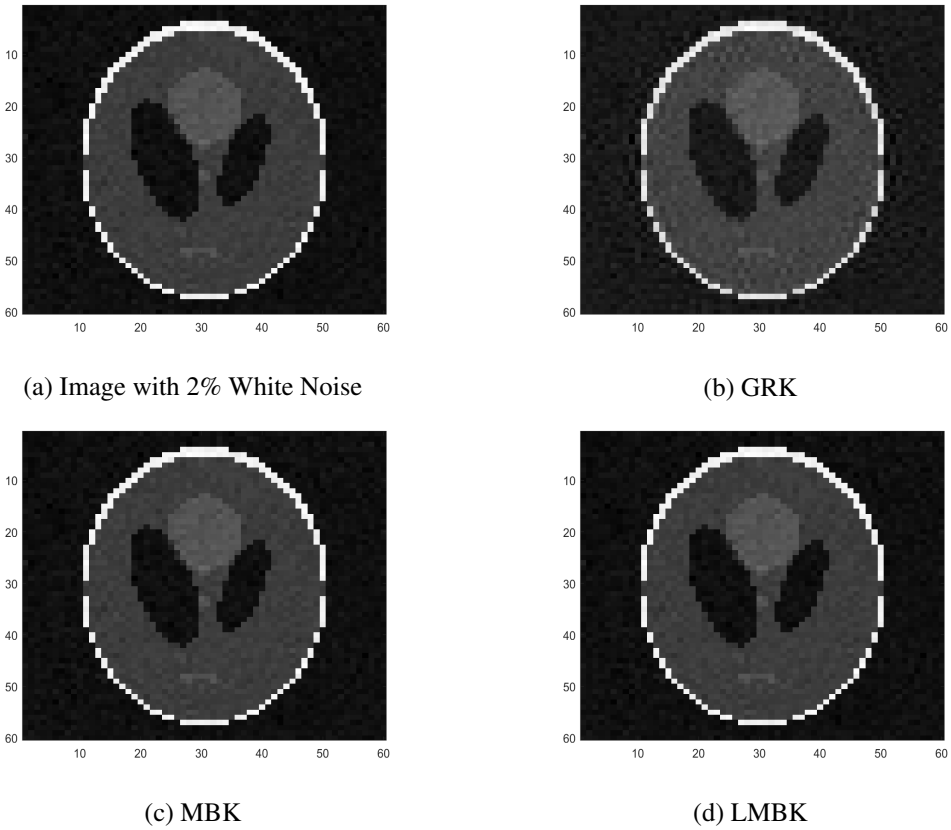
	GRK	MBK	LMBK
IT	196	1	1
CPU	12.5840	10.4741	**3.9521**
SSIM	0.6171	**1.0000**	**1.0000**

Table 5. Test image 2 (d = 554).

	GRK	MBK	LMBK
IT	8248	2	2
CPU	71.6283	58.2951	**11.7294**
SSIM	0.6198	**1.0000**	**1.0000**

According to Tables 4 and 5, it is observed that the SSIM values for MBK and LMBK are both 1, indicating that the images reconstructed by these two algorithms are highly similar to the test images. In contrast, the SSIM values for GRK are 0.6171 in Table 4 and 0.6198 in Table 5, which are relatively lower, correlating with the clarity of the subplot (a) in Figures 6 and 7. Additionally, the reconstruction CPU for the LMBK algorithm is the shortest.

In addition, we added 2% white noise to the second test image and performed image reconstruction. Let the image pixel size be $N = 60$, $p = \left\lfloor \sqrt{2}N \right\rfloor$, $q = 5$, and we get the coefficient matrix A with a size of $15{,}120 \times 3600$. The reconstructed images generated by the GRK, MBK, LMBK, and LMBK algorithms are shown in Figure 8.



(a) Image with 2% White Noise

(b) GRK

(c) MBK

(d) LMBK

Figure 8. Image reconstruction.

As shown in Figure 8, the GRK, MBK, LMBK, and LMBK algorithms have successfully reconstructed the test image with 2% white noise. The corresponding IT, CPU, and SSIM values of these algorithms are compared below to more intuitively evaluate their performance in image reconstruction.

From Table 6, the numerical results indicate that the LMBK algorithm performs the best in terms of IT and CPU. Both the MBK and LMBK algorithms achieve an SSIM value of 1, while the SSIM value for the GRK method is 0.7578, demonstrating the effectiveness of the MBK and LMBK algorithms proposed in this paper.

Table 6. Image with 2% White Noise ($d = 227$).

	GRK	MBK	LMBK
IT	4821	1	1
CPU	17.9093	3.4363	**1.3232**
SSIM	0.7578	**1.0000**	**1.0000**

5. Conclusions

In this paper, to efficiently solve large linear systems, we consider selecting features from the columns of the coefficient matrix, and propose the feature selection algorithm based on Lasso. Subsequently, we introduce a new criterion for selecting the projection block and propose two solving algorithms. One is the maximum residual block Kaczmarz algorithm, and the other is the maximum residual block Kaczmarz algorithm based on feature selection. The convergence theory of these algorithms is analyzed, and their effectiveness is demonstrated through numerical results. Additionally, we also verify the performance of the proposed algorithms in image reconstruction.

Author contributions

Ran-Ran Li: Methodology, data curation, writing–original draft; Hao Liu: Conceptualization, methodology, writing–review and editing. Both of the authors have read and approved the final version of the manuscript for publication.

Use of Generative-AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This research was supported in part by the National Natural Science Foundation of China (Grant Nos. 11401305 and 11571171) and Shenzhen Science and Technology Program (Grant No. JCYJ 20230807142002006).

Conflict of interest

The authors declare no conflicts of interest.

References

1. 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
2. Y. Censor, Row-action methods for huge and sparse systems and their applications, *SIAM Rev.*, **23** (1981), 444–466. https://doi.org/10.1137/1023097

3. C. Brezinski, *Projection methods for systems of equations*, Amsterdam: Elsevier, 1997.
4. 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.
5. S. Kaczmarz, Angenäherte auflösung von systemen linearer gleichungen, *Bull. Int. Acad. Polon. Sci. Lett. A*, **35** (1937), 355–357.
6. F. Natterer, *The mathematics of computerized tomography*, Philadelphia: SIAM, 2001.
7. G. T. Herman, *Fundamentals of computerized tomography: Image reconstruction from projections*, Springer, 2009.
8. C. Byrne, A unified treatment of some iterative algorithms in signal processing and image reconstruction, *Inverse Probl.*, **20** (2004), 103–120. https://doi.org/10.1088/0266-5611/20/1/006
9. C. Popa, R. Zdunek, Kaczmarz extended algorithm for tomographic image reconstruction from limited-data, *Math. Comput. Simulat.*, **65** (2004), 579–598. https://doi.org/10.1016/j.matcom.2004.01.021
10. 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
11. 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
12. Z.-Z. Bai, W.-T. Wu, On greedy randomized Kaczmarz method for solving large sparse linear systems, *SIAM J. Sci. Comput.*, **40** (2018), 592–606. https://doi.org/10.1137/17m1137747
13. 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
14. 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
15. Y. Liu, C.-Q. Gu, Variant of greedy randomized Kaczmarz for ridge regression, *Appl. Numer. Math.*, **143** (2019), 223–246. https://doi.org/10.1016/j.apnum.2019.04.008
16. Z.-Z. Bai, W.-T. Wu, On greedy randomized augmented Kaczmarz method for solving large sparse inconsistent linear systems, *SIAM J. Sci. Comput.*, **43** (2021), A3892–A3911. https://doi.org/10.1137/20m1352235
17. S. Steinerberger, A weighted randomized Kaczmarz method for solving linear systems, *Math. Comput.*, **90** (2021), 2815–2826. https://doi.org/10.1090/mcom/3644
18. X. Yang, A geometric probability randomized Kaczmarz method for large scale linear systems, *Appl. Numer. Math.*, **164** (2021), 139–160. https://doi.org/10.1016/j.apnum.2020.10.016
19. Z.-Z. Bai, L. Wang, On convergence rates of Kaczmarz-type methods with different selection rules of working rows, *Appl. Numer. Math.*, **186** (2023), 289–319. https://doi.org/10.1016/j.apnum.2023.01.013
20. 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

21. A. Ma, D. Molitor, Randomized Kaczmarz for tensor linear systems, *BIT Numer. Math.*, **62** (2022), 171–194. https://doi.org/10.1007/s10543-021-00877-w
22. X.-Z Wang, M.-L Che, C.-X. Mo, Y.-M. Wei, Solving the system of nonsingular tensor equations via randomized Kaczmarz-like method, *J. Comput. Appl. Math.*, **421** (2023), 114856. https://doi.org/10.1016/j.cam.2022.114856
23. S.-Y. Zhong, G.-X. Huang, An almost-maximal residual tensor block Kaczmarz method for large tensor linear systems, *J. Comput. Appl. Math.*, **437** (2024), 115439. https://doi.org/10.1016/j.cam.2023.115439
24. T. Elfving, Block-iterative methods for consistent and inconsistent linear equations, *Numer. Math.*, **35** (1980), 1–12. https://doi.org/10.1007/bf01396365
25. 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
26. I. Necoara, Faster randomized block Kaczmarz algorithms, *SIAM J. Matrix Anal. Appl.*, **40** (2019), 1425–1452. https://doi.org/10.1137/19m1251643
27. 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
28. 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
29. 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
30. R.-R. Li, H. Liu, On global randomized block Kaczmarz method for image reconstruction, *Electron. Res. Arch.*, **30** (2022), 1442–1453. https://doi.org/10.3934/era.2022075
31. 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
32. X.-L. Jiang, K. Zhang, J.-F. Yin, Randomized block Kaczmarz methods with k-means clustering for solving large linear systems, *J. Comput. Appl. Math.*, **403** (2022), 113828. https://doi.org/10.1016/j.cam.2021.113828
33. T. Hastie, R. Tibshirani, M. Wainwright, *Statistical learning with sparsity: The Lasso and generalizations*, London: Chapman and Hall/CRC, 2015.
34. D. L. Donoho, De-noising by soft-thresholding, *IEEE Trans. Inf. Theory*, **41** (1995), 613–627. https://doi.org/10.1109/18.382009
35. L. Xiao, T. Zhang, A proximal stochastic gradient method with progressive variance reduction, *SIAM J. Optim.*, **24** (2014), 2057–2075. https://doi.org/10.1137/140961791
36. A. Beck, M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, *SIAM J. Imag. Sci.*, **2** (2009), 183–202. https://doi.org/10.1137/080716542
37. K. Bredies, D. A. Lorenz, Linear convergence of iterative soft-thresholding, *J. Fourier Anal. Appl.*, **14** (2008), 813–837. https://doi.org/10.1007/s00041-008-9041-1

38. H. Karimi, J. Nutini, M. Schmidt, Linear convergence of gradient and proximal-gradient methods under the polyak-Lojasiewicz condition, In: *Machine learning and knowledge discovery in databases*, Cham: Springer, 2016, 795–811. https://doi.org/10.1007/978-3-319-46128-1_50

39. T. A. Davis, Y. Hu, The university of Florida sparse matrix collection, *ACM Trans. Math. Software*, **38** (2011), 1–25. https://doi.org/10.1145/2049662.2049663

40. U. Sara, M, Akter, M. S. Uddin, Image quality assessment through FSIM, SSIM, MSE and PSNR-a comparative study, *J. Comput. Commun.*, **7** (2019), 8–18. https://doi.org/10.4236/jcc.2019.73002

AIMS Press