On weighted average fast block Kaczmarz methods for solving large consistent linear systems*

Hong-Yu Li[a] and Xin-Hui Shao[†,a]

[a]Department of Mathematics, College of Sciences, Northeastern University, Shenyang 110819, PR China

Abstract

To solve large consistent linear systems, a weighted average fast block Kaczmarz(WAFBK) method is proposed, which is based on the ideas of greedy distance and weighted average. This method does not require the calculation of pseudoinverses of submatrices. Furthermore, we provide a detailed analysis of the convergence properties of various methods, corresponding to four different weights used in the selection probability criterion. And it has been proven that WAFBK-type methods converge to the unique least-norm solutions of linear systems. Numerical results confirm the effectiveness of the WAFBK-type methods and demonstrate their ability to accelerate the convergence rate of the fast deterministic block Kaczmarz method.

Keywords: Consistent systems, Greedy distance, Weighted average, Block Kaczmarz method

1 Introduction

In the fields of scientific and engineering computing such as artificial intelligence, big data analysis, and data mining, the efficient solution of large-scale linear systems remains one of the core concerns of researchers. In this paper, we consider solving large consistent linear systems in the following form

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

where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. Stefan Kaczmarz [12] proposed the Kaczmarz method for solving linear systems (1.1) in 1937, which is a classic and representative row projection iteration method. Due to its advantages of simple operation, low computational cost, and easy implementation, the theory of this method has rapidly developed and been widely applied in image processing [11, 20], distributed computing [8, 19] and computed tomography [13, 14]. In 2009, Strohmer and Vershynin [21] combined the idea of randomly selecting the target row of the coefficient matrix with the Kaczmarz method to present the Randomized Kaczmarz (RK) method, and demonstrated that it had linear convergence speed. In order to speed up the convergence of RK method, Bai and Wu [3] established greedy randomized Kaczmarz (GRK) method by utilizing a novel and beneficial probability criterion. In recent years, there have been a large number of extensions and variations of RK and GRK methods, and we can refer to works [2, 4, 25, 26]. It is not difficult to see that the above-mentioned methods only use one row to project in each iteration. Considering that using multiple work rows in each iteration can accelerate the convergence speed of single row projection methods, block-type methods have been proposed successively.

The idea of block Kaczmarz method can date back to Elfving's research [9], which utilizes a great many of equations simultaneously at each iteration. The iterative format of the block method is defined as follows

$$x_{k+1} = x_k + A_{\mathcal{I}_k}^{\dagger}(b_{\mathcal{I}_k} - A_{\mathcal{I}_k}x_k), \quad k = 0, 1, 2, \ldots,$$

*The project is supported by the Fundamental Research Funds for the Central Universities (N2224005-1)

†Corresponding author. Email address: xinhui1002@126.com

where $A_{\mathcal{I}_k}^{\dagger}$ represents the Moore-Penrose pseudoinverse of the submatrix $A_{\mathcal{I}_k}$ and $\mathcal{I}_k \in [m]$. By observing the above iterative formats, it is evident that the structure and properties of the blocks $A_{\mathcal{I}_k}$ have played a significant role in the convergence speed of this method. Needell et al. [16, 17] presented a randomized block Kaczmarz (RBK) method and demonstrated that if the submatrix $A_{\mathcal{I}_k}$ is well conditioned, in other words, given a great row paving, then this method can be extremely efficient. And they also extended it to solve inconsistent systems. Further, a greedy block Kaczmarz method [18] was presented, which combined GRK and RBK methods. Obviously, this type of block method requires calculating the Moore-Penrose pseudoinverse of submatrix and a great row paving, which is quite time-consuming and generally difficult to satisfy. Nicoara [15] came up with a unified framework for the random average block Kaczmarz (RaBK) method by using a convex combination of several updatings as the search direction to avoid these issues. Moreover, Necoara explored various types of step size and conducted the expected linear convergence rate. Regarding the improvement and expansion of this method, we can read [7, 22]. However, RaBK method requires pre-partitioning the row indices of the coefficient matrix A , and it will only be highly effective when it has a good sampling of the rows into well-conditioned blocks. Therefore, with the inspiration of Gaussian Kaczmarz [10] method, Chen and Huang[5] constructed a fast deterministic block Kaczmarz (FDBK) method so as to further accelerate the convergence of the block method. The iterative format is defined as

$$x_{k+1} = x_k + \frac{d_k^T(b - Ax_k)}{\|A^T d_k\|_2^2} A^T d_k,\ d_k = \sum_{i \in \mathcal{I}_k} \left(b^{(i)} - A^{(i)} x_k\right) e_i.$$

The FDBK method only needs to compute a linear combination of all columns of submatrices $A_{\mathcal{I}_k}$ as search direction. On the one hand, this method eliminates the demand for the calculation of the Moore-Penrose pseudoinverse of submatrix, and on the other hand, it updates the index set through a greedy criterion during each iteration, which means it does not involve pre-determining the row indices. It is obvious that this method greatly reduces the complexity of operations and saves a lot of computation time. Numerical experiments also showed that the convergence performance of this method is quite significant. Afterwards, Xiao et al. [24] constructed a fast greedy block Kaczmarz (FGBK) method by adopting different probability criterion for selecting work rows.

In this paper, we build a weighted average fast block Kaczmarz (WAFBK) method with the goal to further improve the convergence speed of FDBK method, which constructs a new probability criterion for selecting rows by combining the advantages of greedy distance and weighted average. Added to that, we list four types of weights in the probability criterion for this method. According to the different weights, we name them as WAFBK_U, WAFBK_NU, WAFBK_R and WAFBK_D methods respectively. The WAFBK-type methods not only do not need to spend a lot of time to calculate the Moore-Penrose pseudoinverse of submatrix in each step, but the probability criterion for selecting rows is also different from those in FDBK and FGBK methods. And theoretical analyses of WAFBK-type methods corresponding to different weights are provided. Numerical experiments also demonstrate that the WAFBK-type methods outperform both FDBK and FGBK methods in terms of iteration steps and computing time.

Next, we will explore the organizational structure of this article. In the remaining part of this section, we elucidate the symbols and notation employed throughout this article. In Section 2, we first provide a succinct overview of the FDBK and FGBK methods, followed by the construction of the WAFBK-type methods and an in-depth discussion of their convergences on different weights in probability criteria. Following that, the numerical results of six methods are provided in Section 3. Finally, Section 4 serves as an epilogue, encapsulating the conclusion of the entire paper.

For any vector $q \in \mathbb{R}^n$, $q^{(j)}$ is used to denote its jth entry. Let e_j be the column vector of n dimensions whose jth item is 1 and the rest items are 0. For a matrix $Q = (a_{ij}) \in \mathbb{R}^{m \times n}$, $Q^{(i)}$, $\|Q\|_2 := \max\limits_{x \neq 0} \frac{\|Qx\|_2}{\|x\|_2}$, $\|Q\|_F := \sqrt{\sum\limits_{i=1}^{m} \sum\limits_{j=1}^{n} |q_{ij}|^2}$ and $\mathcal{R}(Q)$ represent its ith row, Euclidean norm, Frobenius norm and the range space of the matrix Q, respectively. Then, let $\sigma_{\min}(Q)$ and $\sigma_{\max}(Q)$ denote the smallest nonzero and the largest singular value of the matrix Q, respectively. We use $[t]$ to denote the set of positive integers $[1, 2, \ldots, t]$. For any index set η, we make use of Q_η and $|\eta|$ to represent the row submatrix of Q indexed by η and the cardinality of the set η, respectively.

2 The weighted average fast block Kaczmarz method

Initially, we conduct a brief review of FDBK and FGBK methods. Subsequently, based on the ideas of greedy distance and weighted average, we construct a new probability criterion to derive the WAFBK-type methods.

In FDBK [5] method, the indicator set is defined as follows

$$\mathcal{I}_k = \{i \Big| \left|b^{(i)} - A^{(i)}x_k\right|^2 \geq \varepsilon_k \|b - Ax_k\|_2^2 \left\|A^{(i)}\right\|_2^2\}$$

and

$$\varepsilon_k = \frac{1}{2}\left(\frac{1}{\|b - Ax_k\|_2^2} \max_{1\leq i\leq m}\left\{\frac{\left|b^{(i)} - A^{(i)}x_k\right|^2}{\left\|A^{(i)}\right\|_2^2}\right\} + \frac{1}{\|A\|_F^2}\right).$$

On the basis of FDBK method, Xiao et al. [24] further innovated by constructing a different indicator set and taking into account the utilization of different norm standards to determine the working rows. They proposed FGBK method, whose indicator set is

$$\mathcal{I}_k = \{i \Big| \left|b^{(i)} - A^{(i)}x_k\right|^p \geq \varepsilon_k \left\|A^{(i)}\right\|_p^p\}$$

and

$$\varepsilon_k = \theta \cdot \max_{1\leq i\leq m}\left\{\frac{\left|b^{(i)} - A^{(i)}x_k\right|^p}{\left\|A^{(i)}\right\|_p^p}\right\}, \theta \in (0,1] \quad \text{and} \quad p \in [1, +\infty).$$

One drawback of this method lies in the necessity of calculating $\max\limits_{1\leq i\leq m}\left\{\frac{\left|b^{(i)} - A^{(i)}x_k\right|^p}{\left\|A^{(i)}\right\|_p^p}\right\}$ in each iteration, which will consume a lot of time. Based on this, we combine the advantages of greedy distance and weighted average to construct a new indicator set as shown in Method 1. This approach to determine the indicator set can more accurately describe the distribution of distance between the current iteration solution and the corresponding hyperplane for each row. Additionally, we introduce a relaxation parameter θ, which can control the size of set $\mathcal{I}_k$ by adjusting its value. The proposed method not only avoids calculating the value of $\max\limits_{1\leq i\leq m}\left\{\frac{\left|b^{(i)} - A^{(i)}x_k\right|^2}{\left\|A^{(i)}\right\|_2^2}\right\}$ in each iteration but also has no direct correlation with the Frobenius norm of A. Next, we will discuss the possible selection of weights in ε_k and ensure that the weights satisfy $\omega_i^k \in [0,1]$ and $\sum\limits_{1\leq i\leq m}\omega_i^k = 1$. In this paper, we provide the following four selection rules with regard to the weights in ε_k.

- The uniform selection: $\omega_i^k = \frac{1}{m}$, which can be referred to as WAFBK_U method.
- The non-uniform selection $\omega_i^k = \frac{\left\|A^{(i)}\right\|_2^2}{\|A\|_F^2}$, which can be referred to as WAFBK_NU method.
- The residual selection: $\omega_i^k = \frac{\left|r_k^{(i)}\right|^2}{\|r_k\|_2^2} = \frac{\left|b^{(i)} - A^{(i)}x_k\right|^2}{\|b - Ax_k\|_2^2}$, which can be referred to as WAFBK_R method.
- The distance selection: $\omega_i^k = \frac{\left(d_k^{(i)}\right)^2}{\sum\limits_{1\leq i\leq m}\left(d_k^{(i)}\right)^2}, d_k^{(i)} = \frac{|r_k^{(i)}|}{\left\|A^{(i)}\right\|_2}$ which can be referred to as WAFBK_D method.

We provide some remarks and lemmas for the sake of analyzing the convergence of the WAFBK method more conveniently and clearly.

Remark 2.1. We know that

$$\max_{1\leq i\leq m}\left\{\frac{|b^{(i)} - A^{(i)}x_k|^2}{\left\|A^{(i)}\right\|_2^2}\right\} \geq \sum_{1\leq i\leq m}\omega_i^k\frac{\left|b^{(i)} - A^{(i)}x_k\right|^2}{\left\|A^{(i)}\right\|_2^2} \geq \theta\sum_{1\leq i\leq m}\omega_i^k\frac{\left|b^{(i)} - A^{(i)}x_k\right|^2}{\left\|A^{(i)}\right\|_2^2}, \theta \in [0,1],$$

where $\frac{\left|b^{(i)} - A^{(i)}x_k\right|^2}{\left\|A^{(i)}\right\|_2^2}$ indicates the distance between the current solution x_k and the ith hyperplane, which implies the index set $\mathcal{I}_k$ cannot be empty. That is to say, the maximal distance between

Method 1 The WAFBK method

Input: $A, b, x_0 = 0, k = 0, \theta \in [0, 1]$;
Output: x_{K+1};
1: while $k \leq K$
2: Compute

$$\varepsilon_k = \theta \sum_{1 \leq i \leq m} \omega_i^k \frac{\left|b^{(i)} - A^{(i)} x_k\right|^2}{\left\|A^{(i)}\right\|_2^2};$$

3: Determine the index set

$$\mathcal{I}_k = \left\{ i_k \left| \left|b^{(i)} - A^{(i)} x_k\right|^2 \geq \varepsilon_k \left\|A^{(i)}\right\|_2^2 \right. \right\};$$

4: Compute

$$\xi_k = \sum_{i \in \mathcal{I}_k} r_k^{(i)} e_i; \tag{2.1}$$

5: Update

$$x_{k+1} = x_k + \frac{\xi_k^T r_k}{\|A^T \xi_k\|_2^2} A^T \xi_k;$$

6: $k = k + 1$;
7: end while

the current solution x_k and the ith hyperplane, that is $\max\limits_{1 \leq i \leq m} \left\{ \frac{|b^{(i)} - A^{(i)} x_k|^2}{\|A^{(i)}\|_2^2} \right\}$, must be included in the set $\mathcal{I}_k$.

Lemma 2.1. [23] If both $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ and $\beta = \{\beta_1, \beta_2, \ldots, \beta_n\}$ are two arrays with real components and satisfy $\alpha_j \geq 0, \beta_j \geq 0, j \in \{1, 2, \ldots, n\}$, then it holds that

$$\sum_{j=1}^{n} \frac{\alpha_j}{\beta_j} \geq \frac{\sum\limits_{j=1}^{n} \alpha_j}{\sum\limits_{j=1}^{n} \beta_j}.$$

Lemma 2.2. [3] For any $\nu \in \mathbb{R}^n$ belonging to the column space of A^T, the following inequality holds that

$$\|A\nu\|_2^2 \geq \sigma_{\min}^2(A) \|\nu\|_2^2.$$

Lemma 2.3. [1] Define m be a prescribed positive integer, $\{\alpha_i\}_{i=1}^m$ be nonnegative reals. Then, it holds that:

$$\sum_{i=1}^{m} \alpha_i^2 \geq \frac{1}{m} \left(\sum_{i=1}^{m} \alpha_i \right)^2.$$

Theorem 1. For $A \in \mathbb{R}^{m \times n}$ without any zero row and any initial vector $x_0 \in \mathcal{R}(A^T)$, the iteration sequence $\{x_k\}_{k=0}^{\infty}$ generated by the WAFBK method converges to the least norm solution $x_* = A^{\dagger} b$ of the consistent linear system (1.1). Furthermore, the error estimate obeys

(1) For the WAFBK_U method,

$$\|x_{k+1} - x_*\|_2^2 \leq (1 - \theta \frac{\sigma_{\min}^2(A)}{m \|A\|_F^2} \frac{\|A_{\mathcal{I}_k}\|_F^2}{\sigma_{\max}^2(A_{\mathcal{I}_k})}) \|x_k - x_*\|_2^2, k = 0, 1, 2, \ldots.$$

(2) For the WAFBK_NU method,

$$\|x_{k+1} - x_*\|_2^2 \leq (1 - \theta \frac{\sigma_{\min}^2(A)}{\|A\|_F^2} \frac{\|A_{\mathcal{I}_k}\|_F^2}{\sigma_{\max}^2(A_{\mathcal{I}_k})}) \|x_k - x_*\|_2^2, k = 0, 1, 2, \ldots.$$

(3) For the WAFBK_R method,

$$\|x_{k+1}-x_*\|_2^2 \le (1-\theta\frac{\sigma_{\min}^2(A)}{\|A\|_F^2}\frac{\|A_{\mathcal{I}_k}\|_F^2}{\sigma_{\max}^2(A_{\mathcal{I}_k})})\|x_k-x_*\|_2^2, k=0,1,2,\ldots.$$

(4) For the WAFBK_D method,

$$\|x_{k+1}-x_*\|_2^2 \le (1-\theta\frac{\sigma_{\min}^2(A)}{m\|A\|_F^2}\frac{\|A_{\mathcal{I}_k}\|_F^2}{\sigma_{\max}^2(A_{\mathcal{I}_k})})\|x_k-x_*\|_2^2, k=0,1,2,\ldots.$$

Proof. From step 5 of WAFBK method, we get

$$\begin{aligned} x_{k+1}-x_* &= x_k-x_*+\frac{\xi_k^T(b-Ax_k)}{\|A^T\xi_k\|_2^2}A^T\xi_k \\ &= x_k-x_*-\frac{\xi_k^TA(x_k-x_*)}{\|A^T\xi_k\|_2^2}A^T\xi_k \\ &= x_k-x_*-\frac{A^T\xi_k\xi_k^TA}{\|A^T\xi_k\|_2^2}(x_k-x_*). \end{aligned}$$

According to the Pythagorean theorem, it can be concluded that

$$\begin{aligned} \|x_{k+1}-x_*\|_2^2 &= \|x_k-x_*\|_2^2-\left\|\frac{\xi_k^TA(x_k-x_*)}{\|A^T\xi_k\|_2^2}A^T\xi_k\right\|_2^2 \\ &= \|x_k-x_*\|_2^2-\frac{|\xi_k^T(b-Ax_k)|^2}{\|A^T\xi_k\|_2^2}. \end{aligned} \tag{2.2}$$

Let $E_k\in\mathbb{R}^{m\times|\mathcal{I}_k|}$ be a matrix whose columns are composed of all the unit vector e_i with $i\in\mathcal{I}_k$, $A_{\mathcal{I}_k}=E_k^TA, \hat{\xi}_k=E_k^T\xi_k$, then

$$\left\|\hat{\xi}_k\right\|_2^2=\xi_k^TE_kE_k^T\xi_k=\|\xi_k\|_2^2=\sum_{i\in\mathcal{I}_k}\left|b^{(i)}-A^{(i)}x_k\right|^2 \tag{2.3}$$

and

$$\left\|A^T\xi_k\right\|_2^2=\xi_k^TAA^T\xi_k=\hat{\xi}_k^TE_k^TAA^TE_k\hat{\xi}_k=\hat{\xi}_k^TA_{\mathcal{I}_k}A_{\mathcal{I}_k}^T\hat{\xi}_k=\left\|A_{\mathcal{I}_k}^T\hat{\xi}_k\right\|_2^2.$$

Further, we see that

$$\left\|A_{\mathcal{I}_k}^T\hat{\xi}_k\right\|_2^2=\hat{\xi}_k^TA_{\mathcal{I}_k}A_{\mathcal{I}_k}^T\hat{\xi}_k\le\sigma_{\max}^2(A_{\mathcal{I}_k})\left\|\hat{\xi}_k\right\|_2^2.$$

By combining (2.1) and (2.3), it can be obtained that

$$\begin{aligned} \xi_k^T(b-Ax_k) &= \left(\sum_{i\in\mathcal{I}_k}\left(b^{(i)}-A^{(i)}x_k\right)e_i^T\right)(b-Ax_k) \\ &= \sum_{i\in\mathcal{I}_k}\left(\left(b^{(i)}-A^{(i)}x_k\right)e_i^T(b-Ax_k)\right) \\ &= \sum_{i\in\mathcal{I}_k}\left|b^{(i)}-A^{(i)}x_k\right|^2 \\ &= \left\|\hat{\xi}_k\right\|_2^2. \end{aligned}$$

From this, the second part of equation (2.2) can be further written as

$$\begin{aligned}\frac{\left|\xi_k^T\left(b-Ax_k\right)\right|^2}{\left\|A^T\xi_k\right\|_2^2} &= \frac{\left(\sum\limits_{i\in\mathcal{I}_k}\left|b^{(i)}-A^{(i)}x_k\right|^2\right)\left\|\hat{\xi}_k\right\|_2^2}{\left\|A_{\mathcal{I}_k}^T\hat{\xi}_k\right\|_2^2}\\ &\geq \frac{\sum\limits_{i\in\mathcal{I}_k}\left|b^{(i)}-A^{(i)}x_k\right|^2}{\sigma_{\max}^2\left(A_{\mathcal{I}_k}\right)}\\ &\geq \frac{\sum\limits_{i\in\mathcal{I}_k}\left(\varepsilon_k\left\|A^{(i)}\right\|_2^2\right)}{\sigma_{\max}^2\left(A_{\mathcal{I}_k}\right)}\\ &= \frac{\varepsilon_k\sum\limits_{i\in\mathcal{I}_k}\left\|A^{(i)}\right\|_2^2}{\sigma_{\max}^2\left(A_{\mathcal{I}_k}\right)}\\ &= \frac{\varepsilon_k\left\|A_{\mathcal{I}_k}\right\|_F^2}{\sigma_{\max}^2\left(A_{\mathcal{I}_k}\right)}\end{aligned} \tag{2.4}$$

Based on that, we will discuss the value of ε_k with various weights.

(1) For the WAFBK_U method, $\omega_i = \frac{1}{m}, k = 0, 1, 2, \ldots$, and from **Lemma 2.1** and **2.2**, we have

$$\begin{aligned}\varepsilon_k &= \theta\sum_{1\leq i\leq m}\frac{1}{m}\frac{\left|b^{(i)}-A^{(i)}x_k\right|^2}{\left\|A^{(i)}\right\|_2^2}\\ &\geq \frac{\theta}{m}\frac{\sum\limits_{1\leq i\leq m}\left|b^{(i)}-A^{(i)}x_k\right|^2}{\sum\limits_{1\leq i\leq m}\left\|A^{(i)}\right\|_2^2}\\ &= \frac{\theta}{m}\frac{\left\|b-Ax_k\right\|_2^2}{\left\|A\right\|_F^2}\\ &\geq \theta\frac{\sigma_{\min}^2(A)}{m\left\|A\right\|_F^2}\left\|x_*-x_k\right\|_2^2.\end{aligned} \tag{2.5}$$

(2) For the WAFBK_NU method, $\omega_i = \frac{\left\|A^{(i)}\right\|_2^2}{\left\|A\right\|_F^2}, k = 0, 1, 2, \ldots$, and from **Lemma 2.2**, we have

$$\begin{aligned}\varepsilon_k &= \theta\sum_{1\leq i\leq m}\frac{\left\|A^{(i)}\right\|_2^2}{\left\|A\right\|_F^2}\frac{\left|b^{(i)}-A^{(i)}x_k\right|^2}{\left\|A^{(i)}\right\|_2^2}\\ &= \theta\frac{\left\|b-Ax_k\right\|_2^2}{\left\|A\right\|_F^2}\\ &\geq \theta\frac{\sigma_{\min}^2(A)}{\left\|A\right\|_F^2}\left\|x_*-x_k\right\|_2^2.\end{aligned} \tag{2.6}$$

(3) For the WAFBK_R method, $\omega_i^k = \frac{\left|r_k^{(i)}\right|^2}{\left\|r_k\right\|_2^2} = \frac{\left|b^{(i)}-A^{(i)}x_k\right|^2}{\left\|b-Ax_k\right\|_2^2}, k = 0, 1, 2, \ldots$, we have

$$\begin{aligned}\varepsilon_k &= \theta\sum_{1\leq i\leq m}\frac{\left|b^{(i)}-A^{(i)}x_k\right|^2}{\left\|b-Ax_k\right\|_2^2}\frac{\left|b^{(i)}-A^{(i)}x_k\right|^2}{\left\|A^{(i)}\right\|_2^2}\\ &= \frac{\theta}{\left\|b-Ax_k\right\|_2^2}\sum_{1\leq i\leq m}\frac{\left|b^{(i)}-A^{(i)}x_k\right|^4}{\left\|A^{(i)}\right\|_2^2}.\end{aligned}$$

Through the application of the Cauchy-Schwarz inequality, it is straightforward to deduce that

$$\sum_{1\le i\le m}\left\|A^{(i)}\right\|_2^2\cdot\sum_{1\le i\le m}\frac{\left|b^{(i)}-A^{(i)}x_k\right|^4}{\left\|A^{(i)}\right\|_2^2}\ge\left(\sum_{1\le i\le m}\left\|A^{(i)}\right\|_2\cdot\frac{\left|b^{(i)}-A^{(i)}x_k\right|^2}{\left\|A^{(i)}\right\|_2}\right)^2$$
$$=\left(\sum_{1\le i\le m}\left|b^{(i)}-A^{(i)}x_k\right|^2\right)^2=\|b-Ax_k\|_2^4.$$

Hence, from **Lemma 2.2**, we have

$$\begin{aligned}\varepsilon_k&\ge\frac{\theta}{\|b-Ax_k\|_2^2}\frac{\|b-Ax_k\|_2^4}{\sum\limits_{1\le i\le m}\left\|A^{(i)}\right\|_2^2}\\&\ge\frac{\theta\|b-Ax_k\|_2^2}{\|A\|_F^2}\\&\ge\frac{\theta\sigma_{\min}^2(A)}{\|A\|_F^2}\|x_*-x_k\|_2^2.\end{aligned}\tag{2.7}$$

(4) For the WAFBK_D method, $\omega_i^k=\frac{\left(d_k^{(i)}\right)^2}{\sum\limits_{1\le i\le m}\left(d_k^{(i)}\right)^2}$, $k=0,1,2,\ldots$, and from **Lemma 2.1 - 2.3**, we have

$$\begin{aligned}\varepsilon_k&=\theta\sum_{1\le i\le m}\frac{\left(d_k^{(i)}\right)^2}{\sum\limits_{1\le i\le m}\left(d_k^{(i)}\right)^2}\frac{\left|b^{(i)}-A^{(i)}x_k\right|^2}{\left\|A^{(i)}\right\|_2^2}\\&=\theta\frac{\sum\limits_{1\le i\le m}\left(d_k^{(i)}\right)^4}{\sum\limits_{1\le i\le m}\left(d_k^{(i)}\right)^2}\\&\ge\frac{\theta}{m}\frac{\left(\sum\limits_{1\le i\le m}\left(d_k^{(i)}\right)^2\right)^2}{\sum\limits_{1\le i\le m}\left(d_k^{(i)}\right)^2}\\&=\frac{\theta}{m}\sum_{1\le i\le m}\frac{\left|b^{(i)}-A^{(i)}x_k\right|^2}{\left\|A^{(i)}\right\|_2^2}\\&\ge\frac{\theta}{m}\frac{\|b-Ax_k\|_2^2}{\|A\|_F^2}\\&\ge\theta\frac{\sigma_{\min}^2(A)}{m\|A\|_F^2}\|x_*-x_k\|_2^2.\end{aligned}\tag{2.8}$$

So far, by substituting the estimates in (2.4)-(2.8) into (2.2) and performing recursive operations on the iteration index k, it can be obtained the upper bound in (1)-(4) of Theorem 1. □

Based on the above discussion, the convergence factors of these four methods can be written as

$$\rho_{WAFBK\_NU}=\rho_{WAFBK\_R}=1-\theta\frac{\sigma_{\min}^2(A)}{\|A\|_F^2}\frac{\|A_{\mathcal{I}_k}\|_F^2}{\sigma_{\max}^2(A_{\mathcal{I}_k})}$$

and

$$\rho_{WAFBK\_U} = \rho_{WAFBK\_D} = 1 - \theta \frac{\sigma_{\min}^2(A)}{m \|A\|_F^2} \frac{\|A_{\mathcal{I}_k}\|_F^2}{\sigma_{\max}^2(A_{\mathcal{I}_k})}.$$

Obviously, it holds that

$$\rho_{WAFBK\_NU} = \rho_{WAFBK\_R} \leq \rho_{WAFBK\_D} = \rho_{WAFBK\_U}.$$

Therefore, from a theoretical perspective, WAFBK_NU and WAFBK_R methods will exhibit significantly better convergence performance compared to WAFBK_U and WAFBK_D methods. However, in practical applications, all four methods demonstrate comparably outstanding convergence properties, surpassing the upper bounds prescribed by current theoretical frameworks.

3 Numerical experiments

We will perform extensive numerical experiments and compare them with FDBK [5] and FGBK [24] methods in order to visually demonstrate the excellent convergence of the proposed methods. And numerical experiments in [24] show that the FGBK method with $p = 2$ converges in the least computing time. Therefore, we only consider the FGBK method with $p = 2$ in this paper. As for the values of θ in FGBK method and our proposed methods, we will provide them in each table title and discuss them in detail in 3.3.

The performances of above methods are expressed in terms of the number of iteration steps (labeled as "IT") and computing time in seconds (labeled as "CPU"). For each randomized method, IT and CPU represent the average numerical value of the results that are obtained by repeating the corresponding method 50 times.

We will test two types of matrices of the consistent linear systems, one is the dense matrix generated by the function **randn** in MATLAB, and the other is the matrix from Florida sparse matrix set [6]. The detailed information of the matrices is explained in Tables 3 and 6, where the definition of density is provided here, as shown below

$$\text{density} = \frac{\text{number of nonzero elements in } m \times n}{m \times n}.$$

In addition, to ensure that the coefficient matrix A is consistent, we first generate a vector $x \in \mathbb{R}^n$ and $b = Ax$ through the function **randn**. For all methods, the iterations are commenced from $x_0 = 0$ and terminated when the relative solution error (RSE) falls below a threshold of

$$\text{RSE} = \frac{\|x_k - x_*\|_2^2}{\|x_*\|_2^2} \leq 10^{-6},$$

or the value of IT is beyond 200,000. Additionally, the symbol "--" indicates that a certain method did not meet our prescribed accuracy within 200,000 steps.

Furthermore, the CPU acceleration ratios of the proposed methods relative to FDBK methods are given for the sake of demonstrating intuitively the improvement of the methods we presented, which are respectively set as follows

$$\text{speed-up\_U} = \frac{\text{CPU}_{\text{FDBK}}}{\text{CPU}_{\text{WAFBK\_U}}},$$

$$\text{speed-up\_NU} = \frac{\text{CPU}_{\text{FDBK}}}{\text{CPU}_{\text{WAFBK\_NU}}},$$

$$\text{speed-up\_R} = \frac{\text{CPU}_{\text{FDBK}}}{\text{CPU}_{\text{WAFBK\_R}}}$$

and

$$\text{speed-up\_U} = \frac{\text{CPU}_{\text{FDBK}}}{\text{CPU}_{\text{WAFBK\_D}}}.$$

It can be seen from the definition of acceleration ratio that when these methods reach the same accuracy and the values of speed-up are greater than 1, which mean that the calculation efficiencies of these new methods are even better than that of FDBK method.

The numerical experiments are completed by making use of a computer with 2.30GHz central processor (Intel (R) Core (TM) i5-8300H CPU 2.30GHz), 64 bit Windows 10 operating system and 8.00GB memory by using MATLAB (R2018a version).

3.1 Underdetermined case

We test two types matrix with $m < n$, the numerical results of the six iterative methods are shown in Tables 1 and 2 for random matrices and in Table 3 for sparse matrices. To more intuitively demonstrate the outstanding convergence performance of our proposed methods, we have plotted the RSE in base-10 logarithm verses CPU, as shown in Figures 1 and 2 below.

By comparing the values of CPU and IT in these tables, we can definitely conclude that the proposed methods have an evident promotion at the aspect of CPU time and iteration, which are faster than the other two methods regardless of the size of the coefficient matrix. And it is intuitively observed that these four methods have excellent convergence performance from Figures 1 and 2. Besides, it should be noted that the convergence rates of the four methods we propose are very close and difficult to distinguish between each other especially for solving dense matrices, which is also reflected in Figures 1 below. From this picture, we can easily observe that the curves of four methods are very close to each other. For solving sparse matrices, it can be seen from the results in Figure 2 and Table 3 that WAFBK_U and WAFBK_NU methods are the two fastest among these six methods in all cases. In a word, the difference in convergence performance between WAFBK-type methods is not significant.

In particular, by comparing the proposed methods and FDBK methods, we conclude that when we solve the random matrices, the smallest value of CPU speed-up is 3.2551, the largest is 7.4397; When we solve the sparse matrices, the smallest value of CPU speed-up is 1.4113, the largest is 4.6022. The above data results demonstrate the effectiveness and advantages of our methods.

Table 1. Numerical results of six methods with random matrices and $\theta = 0.5$

$m \times n$		500×1000	500×2000	500×3000	500×4000	500×5000
FDBK	IT	378	106	73	56	51
	CPU	0.0872	0.0873	0.1038	0.1091	0.1258
FGBK	IT	254	77	53	44	38
	CPU	0.0607	0.0644	0.0764	0.0863	0.0946
WAFBK_U	IT	80	22	15	12	10
	CPU	0.0243	0.0222	0.0245	0.0263	0.0272
WAFBK_NU	IT	81	22	15	12	10
	CPU	0.0245	0.0222	0.0242	0.0266	0.0269
WAFBK_R	IT	89	26	19	16	15
	CPU	0.0241	0.0249	0.0288	0.0326	0.0387
WAFBK_D	IT	88	26	19	16	15
	CPU	0.0241	0.0252	0.0292	0.0326	0.0384
Speed-up_U		3.5859	3.9329	4.2330	4.1431	4.6259
Speed-up_NU		3.5529	3.9329	4.2836	4.1007	4.6757
Speed-up_R		3.6150	3.5057	3.6026	3.3444	3.2551
Speed-up_D		3.6195	3.4609	3.5518	3.3446	3.2725

3.2 Overdetermined case

We test two types matrix with $m > n$, the numerical results of the six iterative methods are shown in Tables 4 and 5 for random matrices and in Tables 6 for sparse matrices. In order to more intuitively demonstrate the outstanding convergence performance of our proposed methods, we have plotted the RSE in base-10 logarithm verses CPU, as shown in Figures 3 and 4 below.

By observing the results in Tables 4 - 6, we can draw conclusions similar to those of testing underdetermined matrices. The WAFBK-type methods use fewer IT and CPU time than the other two methods regardless of the size of the matrix or condition number. Furthermore, considering the similarity in convergence results among the four methods we proposed, especially when dealing with random matrices, it becomes quite difficult to compare the convergence performance of these four methods. For solving sparse matrices, we may select the optimal method based on the properties and structural characteristics of matrix A. For example, for solving the well1850 matrix, WAFBK_U method is the fastest method among the newly proposed methods,

Table 2. Numerical results of six methods with random matrices and $\theta = 0.3$

$m \times n$		1000×10000	2000×10000	3000×10000	4000×10000	5000×10000
FDBK	IT	54	101	174	310	547
	CPU	0.6050	2.0386	5.4412	12.9731	27.8898
FGBK	IT	22	36	58	90	148
	CPU	0.2426	0.7327	1.8148	3.8071	7.6540
WAFBK_U	IT	10	17	27	43	72
	CPU	0.1117	0.3454	0.8693	1.8238	3.8442
WAFBK_NU	IT	10	17	27	43	71
	CPU	0.1111	0.3473	0.8724	1.8250	3.7488
WAFBK_R	IT	12	18	27	44	73
	CPU	0.1355	0.3661	0.8754	1.8416	3.7821
WAFBK_D	IT	11	18	28	44	73
	CPU	0.1229	0.3677	0.9099	1.8526	3.7644
Speed-up_U		5.4160	5.9025	6.2593	7.1131	7.2550
Speed-up_NU		5.4472	5.8693	6.2371	7.1087	7.4397
Speed-up_R		4.4657	5.5692	6.2154	7.0444	7.3742
Speed-up_D		4.9231	5.5445	5.9800	7.0025	7.4088

Table 3. Numerical results of six methods for sparse matrices with $\theta = 0.5$

Name		bibd_16_8	bibd_17_8	lp_grow15	df2177	lp_qap12
$m \times n$		120×12870	136×24310	300×645	630×10358	3192×8856
Density		23.33%	20.59%	2.90%	0.34%	0.14%
Cond(A)		9.54	9.04	5.66	2.01	1.26×10^{16}
FDBK	IT	287	258	358	43	112
	CPU	0.2990	0.4857	0.0059	0.0064	0.0251
FGBK	IT	280	256	242	34	81
	CPU	0.2822	0.4784	0.0040	0.0049	0.0164
WAFBK_U	IT	168	151	66	11	22
	CPU	0.1862	0.2981	0.0013	0.0019	0.0063
WAFBK_NU	IT	168	151	67.	11	22
	CPU	0.1862	0.2940	0.0013	0.0019	0.0066
WAFBK_R	IT	209	181	85	14	30
	CPU	0.2119	0.3408	0.0017	0.0024	0.0080
WAFBK_D	IT	209	181	85	14	30
	CPU	0.2111	0.3402	0.0016	0.0022	0.0085
Speed-up_U		1.6058	1.6291	4.5598	3.3403	3.9816
Speed-up_NU		1.6062	1.6521	4.6022	3.2885	3.7740
Speed-up_R		1.4113	1.4251	3.6036	2.7041	3.1245
Speed-up_D		1.4166	1.4276	3.7396	2.8623	2.9431

and for the abb313 matrix, WAFBK_NU or WAFBK_R methods may be more suitable for solving. In general, the four methods we propose perform nearly identically in terms of accelerating convergence.

Compared to FDBK method, when we solve the random matrices, the range of CPU acceleration ratio of WAFBK-type methods is 3.0093 to 6.7306; When we solve sparse matrices, the range of CPU acceleration ratio of WAFBK-type methods is 1.6394 to 5.0802. These numerical results directly reflect the availability and efficiency of WAFBK-type methods in solving these overdetermined matrices.

3.3 Choice of parameter θ

Next, we will discuss the optimal range of θ in WAFBK-type methods and FGBK method, as well as the impact of the value of θ on these methods. Through the analysis in the first two sections, we find that the convergence speed of the four methods we proposed is almost the same. Therefore, this subsection only discusses the impact of relaxation parameter θ on FGBK and

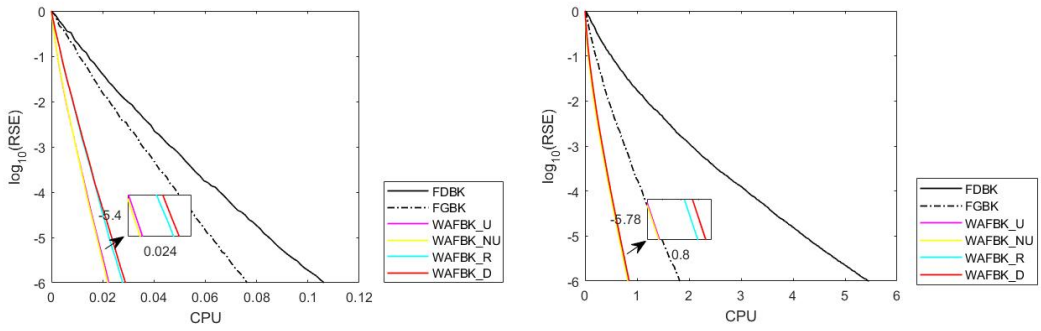

Figure 1. Pictures of six methods for random matrices with $m = 500, n = 3000$(left) and $m = 3000, n = 10000$(right): $\log_{10}$(RSE)versus CPU.

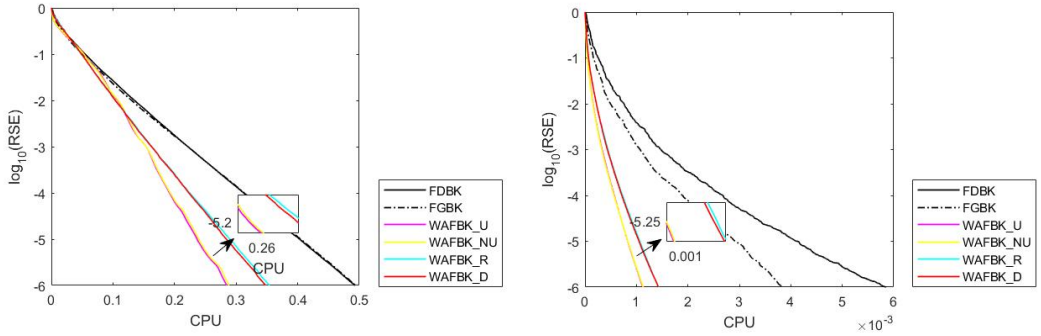

Figure 2. Pictures of six methods for sparse matrices with bibd_17_8(left) and lp_grow15(right): $\log_{10}$(RSE) versus CPU.

Table 4. Numerical results of six methods with random matrices and $\theta = 0.5$

$m \times n$		1000×500	2000×500	3000×500	4000×500	5000×500
FDBK	IT	299	76	50	38	37
	CPU	0.0922	0.0747	0.0763	0.0827	0.1024
FGBK	IT	219	57	43	30	27
	CPU	0.0698	0.0554	0.0660	0.0662	0.0752
WAFBK_U	IT	74	21	15	11	9
	CPU	0.0299	0.0228	0.0249	0.0251	0.0260
WAFBK_NU	IT	75	21	15	11	9
	CPU	0.0306	0.0227	0.0243	0.0257	0.0260
WAFBK_R	IT	78	21	15	11	10
	CPU	0.0279	0.0227	0.0245	0.0258	0.0296
WAFBK_D	IT	77	22	15	11	10
	CPU	0.0280	0.0233	0.0250	0.0263	0.0298
Speed-up_U		3.0824	3.2811	3.1447	3.2990	3.9361
Speed-up_NU		3.0093	3.2906	3.1139	3.2204	3.9440
Speed-up_R		3.3042	3.2906	3.0217	3.2014	3.4635
Speed-up_D		3.2900	3.2103	3.0509	3.1461	3.4407

WAFBK_NU methods.

From the results of Tables 7 and 8, on the one hand, it can be seen that except for $\theta = 0.1$, the WAFBK_NU method converges faster than FGBK method. On the other hand, we find that for solving the well1850 matrix, when the value of θ continuously increases, FGBK method cannot converge within 200,000 steps, indicating that FGBK method heavily relies on the value of θ. To

Table 5. Numerical results of six methods with random matrices and $\theta = 0.3$

$m \times n$		10000×1000	10000×2000	10000×3000	10000×4000	10000×5000
FDBK	IT	42	90	150	272	489
	CPU	0.4748	1.8681	5.1027	11.0190	23.7867
FGBK	IT	15	29	49	77	136
	CPU	0.1718	0.6081	1.6532	3.1420	6.6982
WAFBK_U	IT	10	17	27	43	72
	CPU	0.1109	0.3473	0.8767	1.7469	3.5782
WAFBK_NU	IT	10	17	27	43	72
	CPU	0.1115	0.3467	0.8730	1.7500	3.5691
WAFBK_R	IT	10	17	27	43	71
	CPU	0.1155	0.3578	0.9477	1.7763	3.5341
WAFBK_D	IT	10	17	27	43	72
	CPU	0.1154	0.3584	0.9510	1.7887	3.5516
Speed-up_U		4.2812	5.3792	5.8202	6.3077	6.6476
Speed-up_NU		4.2567	5.3880	5.8446	6.2967	6.6647
Speed-up_R		4.1104	5.2205	5.3842	6.2032	6.7306
Speed-up_D		4.1129	5.2118	5.3655	6.1602	6.6975

Table 6. Numerical results of six methods for sparse matrices $\theta = 0.5$

Name		flower_5_1	abb313	n4c5-b2	well1033	well1850
$m \times n$		211×201	313×176	455×105	1033×320	1850×712
Density		1.42%	2.83%	2.86%	1.43%	0.66%
Cond(A)		3.48×10^{16}	5.97×10^{17}	1.04×10^{16}	166.13	111.31
FDBK	IT	973	20418	22	87634	94786
	CPU	0.0082	0.1774	0.0003	1.7419	3.1349
FGBK	IT	719	16226	19	75219	69566
	CPU	0.0063	0.1364	0.0003	1.4275	2.1792
WAFBK_U	IT	318	9272	4	41024	15341
	CPU	0.0032	0.0962	0.0001	0.9280	0.6171
WAFBK_NU	IT	314	8068	4	36432	15031
	CPU	0.0031	0.0833	0.0001	0.8433	0.6178
WAFBK_R	IT	348	8392	6	40788	19246
	CPU	0.0037	0.0840	0.0001	0.8856	0.7105
WAFBK_D	IT	354	8463	6	49050	20310
	CPU	0.0039	0.0846	0.0001	1.0626	0.7497
Speed-up_U		2.5883	1.8431	5.0492	1.8771	5.0802
Speed-up_NU		2.6810	2.1296	4.9404	2.0657	5.0747
Speed-up_R		2.2140	2.1114	2.6299	1.9669	4.4124
Speed-up_D		2.1172	2.0969	2.5904	1.6394	4.1813

illustrate more intuitively that WAFBK_NU method has better stability than FGBK method, we plot the IT and CPU values of both methods verses θ when solving random and sparse matrices, as shown in Figures 5 and 6. We see that both the IT and CPU values of WAFBK_NU method do not fluctuate significantly with the increase of θ, while the IT and CPU values of FGBK method continue to increase. Especially when the value of θ is greater than 0.7, the convergence rate of FGBK method becomes extremely slow. This indicates that WAFBK_NU method is not highly dependent on theta values and has stronger stability than FGBK method.

4 Conclusions

For solving large linear consistent systems, combining the greedy distance criterion and the advantages of weighted average to construct a new set of indicators, we propose the WAFBK-type methods and provide their convergence analysis. Numerical experiments have shown that our proposed methods are superior to the FDBK and FGBK methods in terms of convergence

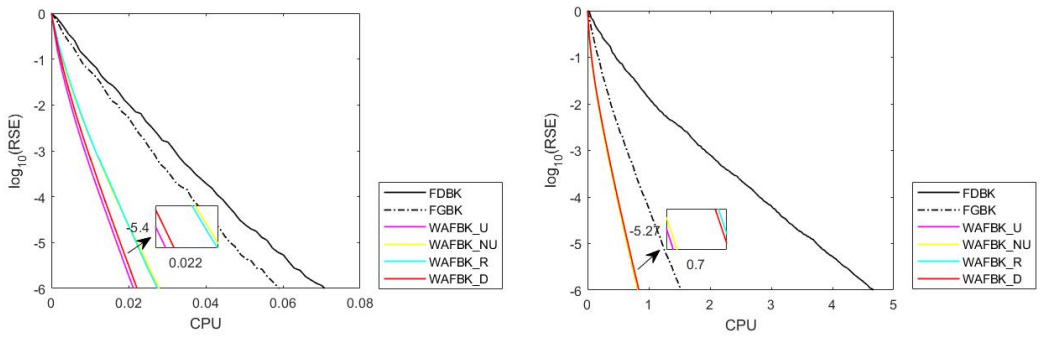

Figure 3. Pictures of six methods for random matrices with $m = 3000, n = 500$(left) and $m = 10000, n = 3000$(right): $\log_{10}$(RSE)versus CPU.

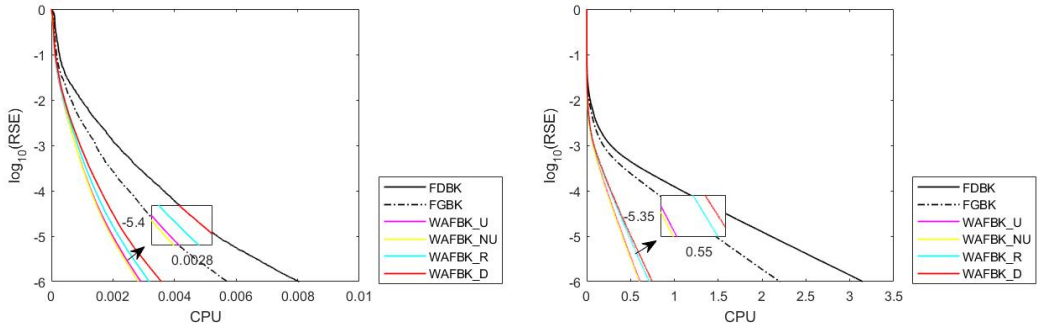

Figure 4. Pictures of six methods for sparse matrices with flower_5_1 (left) and well1850 (right): $\log_{10}$(RSE) versus CPU.

Table 7. Numerical results of FGBK, WAFBK_NU methods for random matrices with various θ

θ	1000×10000				10000×1000			
	FGBK		WAFBK_NU		FGBK		WAFBK_NU	
	IT	CPU	IT	CPU	IT	CPU	IT	CPU
0.1	12	0.1293	10	0.1094	10	0.1053	9	0.0991
0.2	16	0.1684	10	0.1088	12	0.1337	9	0.1009
0.3	22	0.2320	10	0.1105	15	0.1568	10	0.1092
0.4	30	0.3161	10	0.1101	22	0.2290	10	0.1094
0.5	41	0.4267	10	0.1105	35	0.3813	10	0.1099
0.6	58	0.6055	10	0.1087	50	0.5578	10	0.0979
0.7	88	0.9178	11	0.1196	97	1.0546	10	0.1098
0.8	150	1.6432	11	0.1204	180	1.9479	10	0.1093
0.9	359	3.7271	12	0.1319	396	4.1421	10	0.1067
1.0	3157	32.8879	12	0.1328	1704	17.4264	10	0.1115

performance, whether dealing with overdetermined or underdetermined, dense or sparse linear systems. In addition, the relaxation parameter θ in FGBK and WAFBK-type methods is discussed, and numerical experimental results showed that FGBK method has a strong dependence on parameter θ, while WAFBK-type methods are more robust and efficient except for $\theta = 0.1$.

Table 8. Numerical results of FGBK, WAFBK_NU methods for sparse matrices with various θ

θ	bibd_17_8				well1850			
	FGBK		WAFBK_NU		FGBK		WAFBK_NU	
	IT	CPU	IT	CPU	IT	CPU	IT	CPU
0.1	155	0.3213	245	0.4911	21807	0.7488	13612	0.6280
0.2	208	0.4136	151	0.2998	27358	0.9044	14054	0.6263
0.3	250	0.4829	190	0.3792	37776	1.2123	14458	0.6347
0.4	256	0.4980	166	0.3337	48546	1.5406	14735	0.6286
0.5	256	0.5287	151	0.3018	69566	2.1934	15031	0.6390
0.6	261	0.5222	135	0.2746	104046	3.3133	15255	0.6334
0.7	276	0.5495	118	0.2412	156766	4.8684	15550	0.6391
0.8	502	1.0014	111	0.2224	--	--	15835	0.6457
0.9	708	1.4028	107	0.2174	--	--	15699	0.6329
1.0	1131	2.1942	117	0.2300	--	--	15929	0.6445

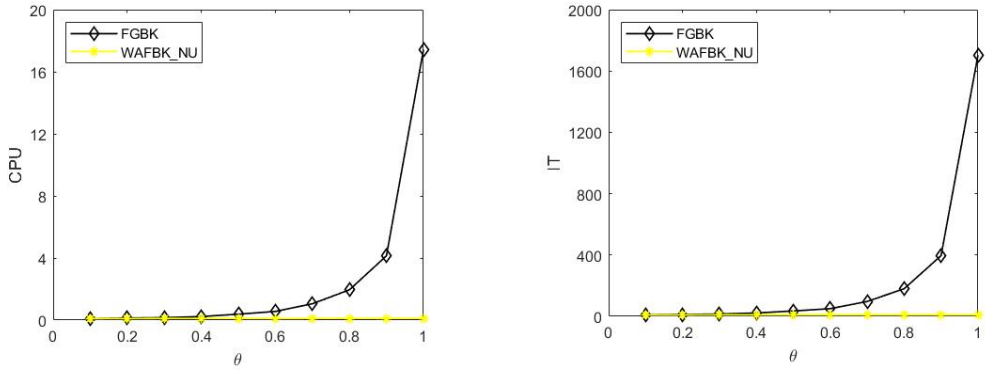

Figure 5. CPU(left) and IT(right) versus θ of FGBK and WAFBK_NU methods for 10000×1000 matrix

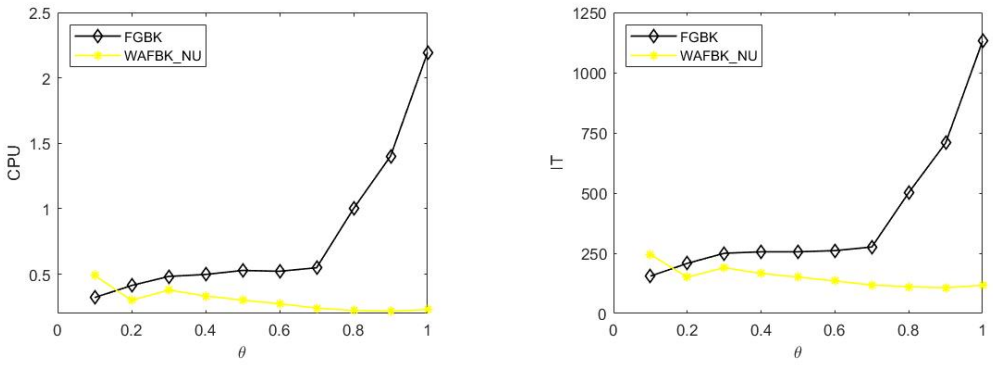

Figure 6. CPU(left) and IT(right) versus θ of FGBK and WAFBK_NU methods for bibd_17_8 matrix.

Acknowledgements

The authors are sincerely grateful to the referees for their invaluable opinions and suggestions, which significantly enhance the quality of our research. This paper is supported by the Fundamental Research Funds for the Central Universities (N2224005-1).

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

[1] Bai, Z.-Z. and Wang, L. On convergence rates of Kaczmarz-type methods with different selection rules of working rows. Appl. Numer. Math. 186 (2023), pp. 289–319.

[2] Bai, Z.-Z. and Wu, W.-T. On greedy randomized augmented Kaczmarz method for solving large sparse inconsistent linear systems. SIAM J. Sci. Comput. 43(6) (2021), A3892–A3911.

[3] Bai, Z.-Z. and Wu, W.-T. On greedy randomized Kaczmarz method for solving large sparse linear systems. SIAM J. Sci. Comput. 40(1) (2018), A592–A606.

[4] Bai, Z.-Z. and Wu, W.-T. On relaxed greedy randomized Kaczmarz methods for solving large sparse linear systems. Appl. Math. Lett. 83 (2018), pp. 21–26.

[5] Chen, J.-Q. and Huang, Z.-D. On a fast deterministic block Kaczmarz method for solving large-scale linear systems. Numer. Algorithms 89(3) (2022), pp. 1007–1029.

[6] Davis, T.A. and Hu, Y. The University of Florida sparse matrix collection. ACM Trans. Math. Softw. 38(1) (2011), pp. 1–25.

[7] Du, K., Si, W.-T., and Sun, X.-H. Randomized extended average block Kaczmarz for solving least squares. SIAM J. Sci. Comput. 42(6) (2020), A3541–A3559.

[8] Elble, J.M., Sahinidis, N.V., and Vouzis, P. GPU computing with Kaczmarz's and other iterative algorithms for linear systems. Parallel Comput. 36(5) (2010), pp. 215–231.

[9] Elfving, T. Block-iterative methods for consistent and inconsistent linear equations. Numer. Math. 35(1) (1980), pp. 1–12.

[10] Gower, R.M. and Richtárik, P. Randomized iterative methods for linear systems. SIAM J. Matrix Anal. Appl. 36(4) (2015), pp. 1660–1690.

[11] Herman, G.T. and Davidi, R. Image reconstruction from a small number of projections. Inverse probl. 24(4) (2008), p. 045011.

[12] Karczmarz, S. Angenäherte Auflösung von systemen linearer glei-chungen. Bull. Int. Acad. Polon. Sci. Lett. A 35 (1937), pp. 355–357.

[13] Lee, S and Kim, H.J. Noise properties of reconstructed images in a kilo-voltage on-board imaging system with iterative reconstruction techniques: A phantom study. Phys. Med. 30(3) (2014), pp. 365–373.

[14] Natterer, F. The mathematics of computerized tomography. SIAM, Philadelphia, PA, 2001.

[15] Necoara, I. Faster randomized block Kaczmarz algorithms. SIAM J. Matrix Anal. Appl. 40(4) (2019), pp. 1425–1452.

[16] Needell, D. and Tropp, J.A. Paved with good intentions: analysis of a randomized block Kaczmarz method. Linear Algebra Appl. 441 (2014), pp. 199–221.

[17] Needell, D., Zhao, R., and Zouzias, A. Randomized block Kaczmarz method with projection for solving least squares. Linear Algebra Appl. 484 (2015), pp. 322–343.

[18] Niu, Y.-Q. and Zheng, B. A greedy block Kaczmarz algorithm for solving large-scale linear systems. Appl. Math. Lett. 104 (2020), p. 106294.

[19] Pasqualetti, F., Carli, R., and Bullo, F. Distributed estimation via iterative projections with application to power network monitoring. Automatica J. IFAC 48(5) (2012), pp. 747–758.

[20] Popa, C. and Zdunek, R. Kaczmarz extended algorithm for tomographic image reconstruction from limited-data. Math. Comput. Simulation 65(6) (2004), pp. 579–598.

[21] Strohmer, T. and Vershynin, R. A randomized Kaczmarz algorithm with exponential convergence. J. Fourier Anal. Appl. 15(2) (2009), pp. 262–278.

[22] Tondji, L. and Lorenz, D.A. Faster randomized block sparse Kaczmarz by averaging. Numer. Algorithms 93(4) (2023), pp. 1417–1451.

[23] Wang, Q.-F., Li, W.-G., and Bao, W.-D. Nonlinear Kaczmarz algorithms and their convergence. J. Comput. Appl. Math. 399 (2022), p. 113720.

[24] Xiao, A.-Q., Yin, J.-F., and Zheng, N. On fast greedy block Kaczmarz methods for solving large consistent linear systems. Comput. Appl. Math. 42(3) (2023), p. 119.

[25] Zhang, J.-J. A new greedy Kaczmarz algorithm for the solution of very large linear systems. Appl. Math. Lett. 91 (2019), pp. 207–212.

[26] Zouzias, A. and Freris, N.M. Randomized extended Kaczmarz for solving least squares. SIAM J. Matrix Anal. Appl. 34(2) (2013), pp. 773–793.