Numerical Algorithms (2022) 89:1007–1029
https://doi.org/10.1007/s11075-021-01143-4

ORIGINAL PAPER


# On a fast deterministic block Kaczmarz method for solving large-scale linear systems

**Jia-Qi Chen**[1] · **Zheng-Da Huang**[1]

Received: 10 December 2020 / Accepted: 23 May 2021 / Published online: 18 June 2021


**Abstract**
For solving large-scale consistent systems of linear equations by iterative methods, a fast block Kaczmarz method based on a greedy criterion of the row selections is proposed. The method is deterministic and needs not compute the pseudoinverses of submatrices or solve subsystems. It is proved that the method will converge linearly to the unique least-norm solutions of the linear systems. Numerical experiments are given to illustrate that the method is more efficient and yields a significant acceleration in convergence for the tested data.

**Keywords** Consistent linear systems · Acceleration · Greedy blocks · Kaczmarz method · Convergence

**Mathematics Subject Classification (2010)** 15A06 · 65F10 · 65F20 · 90C06 · 90C25

## 1 Introduction

We consider using block Kaczmarz-type methods to solve the large-scale consistent linear system in the following form

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

with $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$, where each row of $A$ is nonzero, and $x$ is the $n$-dimensional unknown vector. Whenever the coefficient matrix $A$ is overdetermined

✉ Zheng-Da Huang
zdhuang@zju.edu.cn

Jia-Qi Chen
11835035@zju.edu.cn

[1] School of Mathematical Sciences, Zhejiang University, Hangzhou 310027, Zhejiang, People's Republic of China

(i.e., $m > n$) or underdetermined (i.e., $m < n$), we are often interested in seeking the least-norm solution $x_\star$.

Earlier researches on block Kaczmarz methods may date back at least to the work by Elfving [10] and the work by Eggermont et al. [9]. Differing from the Kaczmarz single-projection methods [2–4, 6, 12–14, 17, 24, 25], in general, multiple equations are used in the block Kaczmarz methods at each iteration. One kind of block Kaczmarz method selects a block of rows from the matrix $A$ and computes the Moore-Penrose pseudoinverse of the block at each iteration. That is to say, starting from an initial guess $x_0$, a general format of the block Kaczmarz (BK) method can be formulated as

$$x_{k+1} = x_k + A_{i_k}^{\dagger}(b_{i_k} - A_{i_k}x_k), \quad k \geq 0 \tag{1.2}$$

if the matrix $A$ and the vector $b$ are partitioned into $A = [A_1^T, A_2^T, \ldots, A_p^T]^T$ and $b = [b_1^T, b_2^T, \ldots, b_p^T]^T$, respectively, for a given integer $1 \leq p \leq m$. In (1.2), $(\cdot)^\dagger$ denotes the Moore-Penrose pseudoinverse of the corresponding matrix, and the block control sequence $\{i_k\}_{k\geq 0}$ is determined by $i_k = (k \bmod p) + 1$, $k = 0, 1, 2, \ldots$. At the $(k+1)$th iteration, the current iteration point $x_k$ is projected onto the hyperplane corresponding to the solution set of $A_{i_k}x = b_{i_k}$. In the case when the coefficient matrix $A$ is nonsingular, Bai and Liu [1] proved the convergence based on the block Meany inequality.

To improve the efficiency of block Kaczmarz methods, looking for appropriate pre-determined partitions of the row indices of the matrix $A$ and block control sequences becomes the main goal of researches. Let $\mathcal{P} = \{\sigma_1, \sigma_2, \ldots, \sigma_p\}$, a partition of $\{1, 2, \ldots, m\}$, be a row paving of the matrix $A$. By choosing $\{\tau_k\}_{k\geq 0}$ uniformly at random from $\mathcal{P}$, Needell and Tropp [18] proposed the randomized block Kaczmarz (RBK) method defined by

$$x_{k+1} = x_k + A_{\tau_k}^{\dagger}(b_{\tau_k} - A_{\tau_k}x_k), \quad k \geq 0 \tag{1.3}$$

for solving linear least-squares problems. In this case, $A_{\tau_k}$ in (1.3) is the row submatrix of $A$ indexed by $\tau_k$, while $b_{\tau_k}$ is the subvector of $b$ with component indices listed in $\tau_k$. In [18], the authors also gave analysis of the expected linear rate of convergence of the RBK method. Based on the block control sequence $\{\tau_k\}_{k\geq 0}$ determined by

$$\tau_k = \left\{ i \;\middle|\; \left|b^{(i)} - A^{(i)}x_k\right|^2 \geq \delta_k \max_{1\leq i\leq m} \left\{ \frac{\left|b^{(i)} - A^{(i)}x_k\right|^2}{\left\|A^{(i)}\right\|_2^2} \right\} \left\|A^{(i)}\right\|_2^2,\ 1 \leq i \leq m \right\} \tag{1.4}$$

with $\delta_k \in (0, 1]$, a pre-defined parameter for each $k \geq 0$, Niu and Zheng [20] proposed the greedy block Kaczmarz (GBK) method. This method does not need to pre-determine a partition of the row indices of the matrix $A$. Clearly, (1.4) is modified from $\mathcal{U}_k$ defined by

$$\mathcal{U}_k = \left\{ i \;\middle|\; |b^{(i)} - A^{(i)}x_k|^2 \geq \epsilon_k \|b - Ax_k\|_2^2 \|A^{(i)}\|_2^2,\ 1 \leq i \leq m \right\} \tag{1.5}$$

with

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

in the greedy randomized Kaczmarz (GRK) method ($\theta = \frac{1}{2}$) [3] and the relaxed greedy randomized Kaczmarz (RGRK) method ($\theta \in [0, 1]$) [4]. Compared with the randomized Kaczmarz (RK) method [24], it is said in [3, 4] that the GRK and RGRK methods use a different but more effective probability criterion.

Recently, Necoara [16] proposed the randomized average block Kaczmarz (RaBK) method defined by

$$x_{k+1} = x_k + \alpha_k \left( \sum_{i \in \tau_k} \omega_i^k \frac{b^{(i)} - A^{(i)} x_k}{\left\| A^{(i)} \right\|_2^2} \left( A^{(i)} \right)^T \right), \quad k \ge 0, \tag{1.6}$$

where $\omega_1^k, \omega_2^k, \ldots, \omega_{|\tau_k|}^k \in [0, 1]$ are weights satisfying $\sum_{i \in \tau_k} \omega_i^k = 1$ and $\alpha_k$ is the stepsize. Here, $|\tau_k|$ denotes the cardinality of the set $\tau_k$. The RaBK method might be second kind of block Kaczmarz method. In (1.6), $x_{k+1}$ can be regarded as an average or a convex combination of the resulting projections of the Kaczmarz single-projection method applied to those rows whose indices appeared in $\tau_k$. This kind of average block Kaczmarz methods can also be traced back to [5, 15, 22]. One can see for details in [5, 15, 16, 22] and the references therein. The RaBK method defined by (1.6) is named as a randomized method in [16], since the block control sequence $\{\tau_k\}_{k \ge 0}$ is selected at random by two sampling methods from a pre-determined partition of the row indices of the coefficient matrix $A$. In [16], Necoara introduced different choices for the stepsize $\alpha_k$ and provided the analysis of the expected linear rate of convergence. These deterministic and randomized block Kaczmarz methods have also been extended to solve the inconsistent linear systems [8, 19, 23].

The Gaussian Kaczmarz (GK) method [11], defined by

$$x_{k+1} = x_k + \frac{\eta^T (b - Ax_k)}{\|A^T \eta\|_2^2} A^T \eta, \quad k \ge 0, \tag{1.7}$$

can be regarded as another kind of block Kaczmarz method that writes directly the increment in the form of a linear combination of all columns of $A^T$ at each iteration, where $\eta$ is a Gaussian vector with mean $0 \in \mathbb{R}^m$ and the covariance matrix $I \in \mathbb{R}^{m \times m}$, i.e., $\eta \sim N(0, I)$. Here $I$ denotes the identity matrix. In (1.7), all columns of $A^T$ are used. The expected linear convergence rate was analyzed in [11] in the case that $A$ is of full column rank.

In this paper, inspired by the GK method [11], and taking the block control sequence $\{\tau_k\}_{k \ge 0}$ determined by

$$\tau_k = \mathcal{U}_k, \quad k \ge 0,$$

where $\mathcal{U}_k$ is defined in (1.5), i.e., based on a greedy criterion of the row selections in [3, 4], we will propose a fast deterministic block Kaczmarz (FDBK) method. At $(k+1)$th iteration, unlike the RK, GRK and RGRK methods [3, 4, 24] choosing one index, the FDBK method uses all indices belonging to $\mathcal{U}_k$. Unlike pre-partitioning the row indices of the coefficient matrix $A$ in BK, RBK and RaBK methods [9, 10, 16, 18], in the FDBK method, the index set $\tau_k$ at each step is selected adaptively and is updated as iterations proceed. Even though this kind of choices of the index sets is similar to that used in the GBK method [20], unlike the GBK method using the Moore-Penrose pseudoinverse of the submatrix $A_{\tau_k}$, the FDBK method needs only to compute a linear combination of all columns of $A_{\tau_k}^T$. Unlike the GK method using the distribution generated by the pre-determined probability, the FDBK method uses the local residual distribution. The iterative format of the FDBK method can also be seen as the case when the RaBK method takes a specific stepsize and weight. Since the GRK method uses one index selected randomly from $\mathcal{U}_k$, while the FDBK method uses all indices in $\mathcal{U}_k$, in a sense, the FDBK method can be regarded as a deterministic block version of the GRK method with certain stepsize. In the extreme case where the set $\tau_k$ only contains a unique element for each $k \geq 0$, the FDBK method, as well as the GRK and RGRK methods, becomes the maximum distance Kaczmarz (MDK) method [21].

In the following, we will prove that the FDBK method converges to the unique least-norm solution $x_\star$ of the consistent linear system (1.1). Numerical experiments on frequently used examples will show that the FDBK method is more efficient than the GBK and the four special cases of the RaBK methods especially in terms of the CPU time.

The paper is organized as follows. In the rest of this section, we describe some notation that are used throughout this paper. In Section 2, we propose the fast deterministic block Kaczmarz method and discuss basic properties. In Section 3, we take the convergence analysis and derive the error estimates. Numerical results are reported in Section 4, and the paper is ended in Section 5 with a few conclusions.

For a matrix $M \in \mathbb{R}^{m\times n}$, we call $M^{(i)}$, $\|M\|_F$, $M^T$, $M^\dagger$ and $\mathscr{R}(M)$ the $i$th row, the Frobenius norm, the transpose, the Moore-Penrose pseudoinverse and the range space of the matrix $M$, respectively, and let $\lambda_{\min}(M^TM)$ and $\lambda_{\max}(M^TM)$ represent the smallest nonzero and the largest eigenvalue of the matrix $M^TM$. For a vector $u \in \mathbb{R}^m$, we use $u^{(i)}$, $u^T$ and $\|u\|_2$ to denote its $i$th entry, the transpose and the Euclidean norm of the vector $u$, respectively. Let $[m]$ represent the set $\{1, 2, \ldots, m\}$. For any index set $\upsilon$, we use $M_\upsilon$, $u_\upsilon$ and $|\upsilon|$ to denote the row submatrix of $M$ indexed by $\upsilon$, the subvector of $u$ with component indices listed in $\upsilon$ and the cardinality of the set $\upsilon$, respectively. Let $I$ and $e_i$ represent the identity matrix and the $i$th column of the identity matrix, respectively.

## 2 The fast deterministic block Kaczmarz method

The fast deterministic block Kaczmarz (FDBK) method for solving the consistent linear system (1.1) is defined as follows.

**Algorithm 1** The Fast Deterministic Block Kaczmarz (FDBK) method.

**Input:** $A$, $b$ and $x_0$

**for** $k = 0, 1, \ldots$ until convergence or $\|b - Ax_k\|_2^2 = 0$ **do**

**Step 1:** Compute

$$\epsilon_k = \frac{1}{2}\left(\frac{1}{\|b - Ax_k\|_2^2} \max_{1\le i\le m}\left\{\frac{\left|b^{(i)} - A^{(i)}x_k\right|^2}{\left\|A^{(i)}\right\|_2^2}\right\} + \frac{1}{\|A\|_F^2}\right). \tag{2.1}$$

**Step 2:** Determine the index set of positive integers

$$\tau_k = \left\{i \;\middle|\; \left|b^{(i)} - A^{(i)}x_k\right|^2 \ge \epsilon_k \|b - Ax_k\|_2^2 \left\|A^{(i)}\right\|_2^2\right\}. \tag{2.2}$$

**Step 3:** Compute

$$\eta_k = \sum_{i\in\tau_k}\left(b^{(i)} - A^{(i)}x_k\right)e_i. \tag{2.3}$$

**Step 4:** Set

$$x_{k+1} = x_k + \frac{\eta_k^T(b - Ax_k)}{\left\|A^T\eta_k\right\|_2^2}A^T\eta_k. \tag{2.4}$$

**end for**

In this section, we want to discuss basic properties of the FDBK method. Let's begin with the problem: Is the FDBK method executable unconditionally? In other words, does the iterative sequence $\{x_k\}_{k\ge 0}$ generated by the FDBK method exist for any initial guess $x_0$?

**Property 1** *The FDBK method described by Algorithm 1 is executable unconditionally.*

*Proof* For any $x_0 \in \mathbb{R}^n$, $\|b - Ax_0\|_2$ equals zero or not. When $\|b - Ax_0\|_2 = 0$, the iterative sequence only contains $x_0$. When $\|b - Ax_0\|_2 \neq 0$, $\tau_0$ is clearly nonempty. In this case, to show $x_1$ defined by (2.4) exists, what we need to do is to prove $\left\|A^T\eta_0\right\|_2 \neq 0$ since only basic algebra operations of matrices are used. If $\left\|A^T\eta_0\right\|_2 = 0$, or equivalently, $A^T\eta_0 = 0$, then

$$\eta_0^T(b - Ax_0) = (A^T\eta_0)^T(x_\star - x_0) = 0 \tag{2.5}$$

according to $Ax_\star = b$. However, by (2.1)–(2.3), we have

$$\eta_0^T(b - Ax_0) = \sum_{i\in\tau_0}\left|b^{(i)} - A^{(i)}x_0\right|^2 \ge \epsilon_0\|b - Ax_0\|_2^2\sum_{i\in\tau_0}\left\|A^{(i)}\right\|_2^2 > 0, \tag{2.6}$$

which is a contradiction to (2.5). Therefore, $\left\|A^T\eta_0\right\|_2 \neq 0$ holds true, and accordingly $x_1$ can be computed via (2.4).

If $x_k$ has been computed for some $k \ge 1$, then repeat the above procedure by substituting $\eta_0$ and $x_0$ with $\eta_k$ and $x_k$, respectively, we can find easily that the iterative sequence is $\{x_0, x_1, \ldots, x_k\}$ when $\|b - Ax_k\|_2 = 0$, and that (2.5) and (2.6) remain

true with $\eta_0, x_0, \tau_0$ and $\epsilon_0$ replaced by $\eta_k, x_k, \tau_k$ and $\epsilon_k$ in the case of $\|b-Ax_k\|_2 \neq 0$. Therefore, when $\|b - Ax_k\|_2 \neq 0$, the contradiction still happens. It follows that $A^T\eta_k \neq 0$, which implies in the same way as described above that $x_{k+1}$ exists.

By the induction method, $\{x_k\}_{k\geq 0}$ exists for any initial point $x_0 \in \mathbb{R}^n$.

This completes the proof. □

**Property 2** *We have $\eta_k \perp b - Ax_{k+1}$ for all $k \geq 0$, where $\eta_k$ is defined by (2.3).*

*Proof* In fact, for any $k \geq 0$, according to the update rule (2.4), we have

$$\begin{aligned}\eta_k^T\,(b - Ax_{k+1}) &= \eta_k^T\left(b - A\left(x_k + \frac{\eta_k^T(b - Ax_k)}{\left\|A^T\eta_k\right\|_2^2}A^T\eta_k\right)\right)\\ &= \eta_k^T\,(b - Ax_k) - \frac{\eta_k^T\,(b - Ax_k)}{\left\|A^T\eta_k\right\|_2^2}\eta_k^T AA^T\eta_k\\ &= \eta_k^T\,(b - Ax_k) - \eta_k^T\,(b - Ax_k)\\ &= 0,\end{aligned}$$

which completes the proof. □

**Property 3** *The format (2.4) can be written in the format of (1.6) with*

$$\alpha_k = \frac{\|\eta_k\|_2^2\|A_{\tau_k}\|_F^2}{\|A^T\eta_k\|_2^2},\quad \omega_i^k = \frac{\|A^{(i)}\|_2^2}{\|A_{\tau_k}\|_F^2},\quad i \in \tau_k,\quad k \geq 0. \tag{2.7}$$

*Here, $\tau_k$ is defined by (2.2).*

*Proof* For each $k \geq 0$, by the straightforward computations, the format (2.4) can be rewritten as:

$$\begin{aligned}x_{k+1} &= x_k + \frac{\eta_k^T(b - Ax_k)}{\|A^T\eta_k\|_2^2}\left(\sum_{i\in\tau_k}\left(b^{(i)} - A^{(i)}x_k\right)\left(A^{(i)}\right)^T\right)\\ &= x_k + \frac{\|\eta_k\|_2^2\|A_{\tau_k}\|_F^2}{\|A^T\eta_k\|_2^2}\left(\sum_{i\in\tau_k}\frac{\|A^{(i)}\|_2^2}{\|A_{\tau_k}\|_F^2}\frac{b^{(i)} - A^{(i)}x_k}{\|A^{(i)}\|_2^2}\left(A^{(i)}\right)^T\right),\end{aligned}$$

which is the format of (1.6). (2.7) follows. The proof is completed. □

## 3 Convergence analysis and error estimates of the FDBK method

In this section, we will devote ourselves to the proof of the following convergence theorem for the FDBK method.

**Theorem 3.1** *Let $A \in \mathbb{R}^{m\times n}$ be a matrix without any zero row, and $b \in \mathbb{R}^m$. The iteration sequence $\{x_k\}_{k=0}^{\infty}$, generated by the FDBK method starting from any initial*

*guess $x_0 \in \mathscr{R}(A^T)$, exists and converges to the unique least-norm solution $x_\star = A^\dagger b$ of the consistent linear system (1.1), with the error estimate*

$$\|x_{k+1} - x_\star\|_2^2 \le \left(1 - \gamma_k \frac{\left\|A_{\tau_k}\right\|_F^2}{\lambda_{\max}\left(A_{\tau_k} A_{\tau_k}^T\right)} \frac{\lambda_{\min}\left(A^T A\right)}{\|A\|_F^2}\right) \|x_k - x_\star\|_2^2, \quad k \ge 0, \tag{3.1}$$

*where*

$$\gamma_k = \frac{1}{2}\left(\frac{\|A\|_F^2}{q_k \left\|A_{\zeta_k}\right\|_F^2 + (1 - q_k)\left\|A_{\tau_k}\right\|_F^2} + 1\right) \ge 1, \quad k \ge 0, \tag{3.2}$$

*with*

$$\zeta_k = \left\{ i \;\middle|\; b^{(i)} - A^{(i)} x_k \ne 0,\ i \in [m] \right\}, \quad k \ge 0, \tag{3.3}$$

*and*

$$q_k = \begin{cases} \dfrac{\max\limits_{i \in \zeta_k \setminus \tau_k} \left\{ \dfrac{\left|b^{(i)} - A^{(i)} x_k\right|^2}{\left\|A^{(i)}\right\|_2^2} \right\}}{\max\limits_{1 \le i \le m} \left\{ \dfrac{\left|b^{(i)} - A^{(i)} x_k\right|^2}{\left\|A^{(i)}\right\|_2^2} \right\}} < 1, & \text{if } \tau_k \ne [m] \text{ and } \zeta_k \ne \tau_k, \quad k \ge 0. \\ 1, & \text{otherwise}, \end{cases} \tag{3.4}$$

*Here the equality in (3.2) holds only when $\tau_k = [m]$ for each $k \ge 0$.*

*Proof* The iteration sequence $\{x_k\}_{k=0}^{\infty}$ exists and is contained in $\mathscr{R}(A^T)$ for any initial guess $x_0 \in \mathscr{R}(A^T)$ via Property 1 and (2.4), respectively. $\{x_k\}_{k \ge 0}$ might be a sequence containing finite members once $\|b - Ax_k\|_2 = 0$ happens for some $0 \le k < +\infty$, or a sequence containing infinite members once $\|b - Ax_k\|_2 \ne 0$ happens for each $k \ge 0$. Without loss of generality, in the following, it is often assumed that $\|b - Ax_k\|_2 \ne 0$ is true for each $k \ge 0$.

For each $k \ge 0$, $P_k$, defined by $P_k = \frac{A^T \eta_k \eta_k^T A}{\|A^T \eta_k\|_2^2}$, exists by Property 1. Then for the least-norm solution $x_\star$ of the consistent linear system (1.1), according to the iterative format (2.4) of the FDBK method, we have

$$\begin{aligned} x_{k+1} - x_\star &= x_k - x_\star + \frac{\eta_k^T (b - Ax_k)}{\left\|A^T \eta_k\right\|_2^2} A^T \eta_k \\ &= x_k - x_\star - \frac{\eta_k^T A (x_k - x_\star)}{\left\|A^T \eta_k\right\|_2^2} A^T \eta_k \\ &= x_k - x_\star - \frac{A^T \eta_k \eta_k^T A}{\|A^T \eta_k\|_2^2} (x_k - x_\star) \\ &= (I - P_k)(x_k - x_\star). \end{aligned}$$

Since $P_k^T = P_k$ and $P_k^2 = P_k$, $P_k$ is an orthogonal projector. It follows that

$$\begin{aligned}\|x_{k+1} - x_\star\|_2^2 &= \|(I - P_k)(x_k - x_\star)\|_2^2 \\ &= \|x_k - x_\star\|_2^2 - \|P_k(x_k - x_\star)\|_2^2 \\ &= \|x_k - x_\star\|_2^2 - \left\| \frac{\eta_k^T A (x_k - x_\star)}{\|A^T \eta_k\|_2^2} A^T \eta_k \right\|_2^2 \\ &= \|x_k - x_\star\|_2^2 - \frac{\left|\eta_k^T (b - Ax_k)\right|^2}{\left\|A^T \eta_k\right\|_2^2}\end{aligned} \tag{3.5}$$

by the Pythagorean Theorem.

Let $E_k \in \mathbb{R}^{m \times |\tau_k|}$ denote the matrix whose columns in turn are composed of all the vectors $e_i \in \mathbb{R}^m$ with $i \in \tau_k$, then, $A_{\tau_k} = E_k^T A$. Denote by $\widehat{\eta}_k = E_k^T \eta_k$, we have

$$\|\widehat{\eta}_k\|_2^2 = \eta_k^T E_k E_k^T \eta_k = \|\eta_k\|_2^2 = \sum_{i \in \tau_k} \left| b^{(i)} - A^{(i)} x_k \right|^2 \tag{3.6}$$

and

$$\left\| A^T \eta_k \right\|_2^2 = \eta_k^T A A^T \eta_k = \widehat{\eta}_k^T E_k^T A A^T E_k \widehat{\eta}_k = \widehat{\eta}_k^T A_{\tau_k} A_{\tau_k}^T \widehat{\eta}_k = \left\| A_{\tau_k}^T \widehat{\eta}_k \right\|_2^2. \tag{3.7}$$

Therefore, we can obtain

$$\left\| A_{\tau_k}^T \widehat{\eta}_k \right\|_2^2 = \widehat{\eta}_k^T A_{\tau_k} A_{\tau_k}^T \widehat{\eta}_k \le \lambda_{\max}\left( A_{\tau_k} A_{\tau_k}^T \right) \|\widehat{\eta}_k\|_2^2 . \tag{3.8}$$

From the definition of $\eta_k$ in (2.3), we have, by (3.6), that

$$\begin{aligned}\eta_k^T (b - Ax_k) &= \left( \sum_{i \in \tau_k} \left( b^{(i)} - A^{(i)} x_k \right) e_i^T \right) (b - Ax_k) \\ &= \sum_{i \in \tau_k} \left( \left( b^{(i)} - A^{(i)} x_k \right) e_i^T (b - Ax_k) \right) \\ &= \sum_{i \in \tau_k} \left| b^{(i)} - A^{(i)} x_k \right|^2 \\ &= \|\widehat{\eta}_k\|_2^2 .\end{aligned} \tag{3.9}$$

Since $x_k, x_\star \in \mathscr{R}(A^T)$ implies $x_k - x_\star \in \mathscr{R}(A^T)$, we can obtain

$$\|b - Ax_k\|_2^2 = \|A(x_k - x_\star)\|_2^2 \ge \lambda_{\min}\left( A^T A \right) \|x_k - x_\star\|_2^2 \tag{3.10}$$

by the following well-known inequality

$$\|Au\|_2^2 \ge \lambda_{\min}\left( A^T A \right) \|u\|_2^2, \quad \forall u \in \mathscr{R}\left( A^T \right).$$

It then holds that

$$\begin{aligned}\frac{\left|\eta_k^T(b-Ax_k)\right|^2}{\left\|A^T\eta_k\right\|_2^2} &= \frac{\left(\sum\limits_{i\in\tau_k}\left|b^{(i)}-A^{(i)}x_k\right|^2\right)\|\widehat{\eta}_k\|_2^2}{\left\|A_{\tau_k}^T\widehat{\eta}_k\right\|_2^2}\\ &\geq \frac{\sum\limits_{i\in\tau_k}\left|b^{(i)}-A^{(i)}x_k\right|^2}{\lambda_{\max}\left(A_{\tau_k}A_{\tau_k}^T\right)}\\ &\geq \frac{\sum\limits_{i\in\tau_k}\left(\epsilon_k\|b-Ax_k\|_2^2\left\|A^{(i)}\right\|_2^2\right)}{\lambda_{\max}\left(A_{\tau_k}A_{\tau_k}^T\right)}\\ &= \frac{\epsilon_k\|b-Ax_k\|_2^2\sum\limits_{i\in\tau_k}\left\|A^{(i)}\right\|_2^2}{\lambda_{\max}\left(A_{\tau_k}A_{\tau_k}^T\right)}\\ &\geq \frac{\epsilon_k\left\|A_{\tau_k}\right\|_F^2\lambda_{\min}\left(A^TA\right)}{\lambda_{\max}\left(A_{\tau_k}A_{\tau_k}^T\right)}\|x_k-x_\star\|_2^2\end{aligned} \tag{3.11}$$

by (3.7)–(3.10) and the definition of $\tau_k$ in (2.2).

For each $k\geq 0$, since

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

by the definition of $\epsilon_k$ in (2.1), we have

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

which implies via (2.2) that $l\in\tau_k$, i.e., $\tau_k\neq\emptyset$, when

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

In the case of $\tau_k\neq[m]$ and $\zeta_k\neq\tau_k$, where $\zeta_k$ is defined by (3.3), $q_k$, defined by (3.4), is a positive number less than 1, i.e., $0<q_k<1$. According to the definition of $\gamma_k$ in (3.2), we have

$$\gamma_k=\frac{1}{2}\left(\frac{\|A\|_F^2}{q_k\left\|A_{\zeta_k}\right\|_F^2+(1-q_k)\left\|A_{\tau_k}\right\|_F^2}+1\right)>\frac{1}{2}\left(\frac{\|A\|_F^2}{\left\|A_{\zeta_k}\right\|_F^2}+1\right)\geq 1. \tag{3.12}$$

Since $i \notin \tau_k$ and $i \in \zeta_k$ if and only if $i \in \zeta_k \backslash \tau_k$, it follows that

$$\begin{aligned}
\|b - Ax_k\|_2^2 &= \sum_{i\in\tau_k} \frac{\left|b^{(i)} - A^{(i)}x_k\right|^2}{\left\|A^{(i)}\right\|_2^2} \left\|A^{(i)}\right\|_2^2 + \sum_{i\in\zeta_k\backslash\tau_k} \frac{\left|b^{(i)} - A^{(i)}x_k\right|^2}{\left\|A^{(i)}\right\|_2^2} \left\|A^{(i)}\right\|_2^2 \\
&\le \max_{1\le i\le m} \left\{ \frac{\left|b^{(i)} - A^{(i)}x_k\right|^2}{\left\|A^{(i)}\right\|_2^2} \right\} \left( \sum_{i\in\tau_k} \left\|A^{(i)}\right\|_2^2 \right) \\
&\quad + \max_{i\in\zeta_k\backslash\tau_k} \left\{ \frac{\left|b^{(i)} - A^{(i)}x_k\right|^2}{\left\|A^{(i)}\right\|_2^2} \right\} \left( \sum_{i\in\zeta_k\backslash\tau_k} \left\|A^{(i)}\right\|_2^2 \right) \\
&= \max_{1\le i\le m} \left\{ \frac{\left|b^{(i)} - A^{(i)}x_k\right|^2}{\left\|A^{(i)}\right\|_2^2} \right\} \left( \sum_{i\in\tau_k} \left\|A^{(i)}\right\|_2^2 + q_k \sum_{i\in\zeta_k\backslash\tau_k} \left\|A^{(i)}\right\|_2^2 \right) \\
&= \max_{1\le i\le m} \left\{ \frac{\left|b^{(i)} - A^{(i)}x_k\right|^2}{\left\|A^{(i)}\right\|_2^2} \right\} \left( q_k \left\|A_{\zeta_k}\right\|_F^2 + (1-q_k) \left\|A_{\tau_k}\right\|_F^2 \right),
\end{aligned}$$

where the last equality holds due to

$$\sum_{i\in\zeta_k\backslash\tau_k} \left\|A^{(i)}\right\|_2^2 = \left\|A_{\zeta_k}\right\|_F^2 - \left\|A_{\tau_k}\right\|_F^2.$$

Thus, we have

$$\begin{aligned}
\epsilon_k \|A\|_F^2 &= \frac{1}{2} \left( \frac{\|A\|_F^2}{\|b - Ax_k\|_2^2} \max_{1\le i\le m} \left\{ \frac{\left|b^{(i)} - A^{(i)}x_k\right|^2}{\left\|A^{(i)}\right\|_2^2} \right\} + 1 \right) \\
&\ge \frac{1}{2} \left( \frac{\|A\|_F^2}{q_k \left\|A_{\zeta_k}\right\|_F^2 + (1-q_k) \left\|A_{\tau_k}\right\|_F^2} + 1 \right) \\
&= \gamma_k.
\end{aligned} \tag{3.13}$$

In the case of $\tau_k \neq [m]$ and $\zeta_k = \tau_k$, i.e., $\zeta_k = \tau_k \subsetneqq [m]$, it holds that

$$\|b - Ax_k\|_2^2 = \sum_{i\in\zeta_k} \frac{\left|b^{(i)} - A^{(i)}x_k\right|^2}{\left\|A^{(i)}\right\|_2^2} \left\|A^{(i)}\right\|_2^2 \le \max_{1\le i\le m} \left\{ \frac{\left|b^{(i)} - A^{(i)}x_k\right|^2}{\left\|A^{(i)}\right\|_2^2} \right\} \left\|A_{\zeta_k}\right\|_F^2$$

and that

$$\epsilon_k \|A\|_F^2 \ge \frac{1}{2} \left( \frac{\|A\|_F^2}{\left\|A_{\zeta_k}\right\|_F^2} + 1 \right) = \gamma_k > 1. \tag{3.14}$$

In the case of $\tau_k = [m]$, it should be that

$$\frac{\left|b^{(1)} - A^{(1)}x_k\right|^2}{\left\|A^{(1)}\right\|_2^2} = \frac{\left|b^{(2)} - A^{(2)}x_k\right|^2}{\left\|A^{(2)}\right\|_2^2} = \cdots = \frac{\left|b^{(m)} - A^{(m)}x_k\right|^2}{\left\|A^{(m)}\right\|_2^2},$$

i.e., $\zeta_k = \tau_k = [m]$. Therefore,

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

Thanks to the definition of $\epsilon_k$ and $\gamma_k$ in (2.1) and (3.2), respectively, we can obtain

$$\epsilon_k\|A\|_F^2=\gamma_k=1. \tag{3.15}$$

Inequalities in (3.12)–(3.15) imply that

$$\epsilon_k\ge\frac{1}{\|A\|_F^2}\gamma_k, \tag{3.16}$$

and that (3.2) is true and the equality in it holds only when $\tau_k=[m]$.

For any nonzero vector $\widehat{y}\in\mathbb{R}^{|\tau_k|}$, we have

$$\lambda_{\min}(A^TA)=\lambda_{\min}(AA^T)\le\frac{(E_k\widehat{y})^TAA^T(E_k\widehat{y})}{(E_k\widehat{y})^T(E_k\widehat{y})}=\frac{\widehat{y}^TA_{\tau_k}A_{\tau_k}^T\widehat{y}}{\widehat{y}^T\widehat{y}}\le\lambda_{\max}(A_{\tau_k}A_{\tau_k}^T). \tag{3.17}$$

Moreover, by (2.2) and (3.16), we have

$$\gamma_k\frac{\left\|A_{\tau_k}\right\|_F^2}{\|A\|_F^2}\le\epsilon_k\left\|A_{\tau_k}\right\|_F^2=\sum_{i\in\tau_k}\epsilon_k\|A^{(i)}\|_2^2\le\sum_{i\in\tau_k}\frac{\left|b^{(i)}-A^{(i)}x_k\right|^2}{\|b-Ax_k\|_2^2}\le 1. \tag{3.18}$$

Since it is always true that

$$0<\frac{\lambda_{\min}\left(A^TA\right)}{\|A\|_F^2}\le 1,\quad \frac{\left\|A_{\tau_k}\right\|_F^2}{\lambda_{\max}\left(A_{\tau_k}A_{\tau_k}^T\right)}\ge 1,$$

it follows from (3.17) and (3.18) that

$$\begin{aligned}0&\le 1-\gamma_k\frac{\left\|A_{\tau_k}\right\|_F^2}{\|A\|_F^2}\frac{\lambda_{\min}\left(A^TA\right)}{\lambda_{\max}\left(A_{\tau_k}A_{\tau_k}^T\right)}=1-\gamma_k\frac{\left\|A_{\tau_k}\right\|_F^2}{\lambda_{\max}\left(A_{\tau_k}A_{\tau_k}^T\right)}\frac{\lambda_{\min}\left(A^TA\right)}{\|A\|_F^2}\\&\le 1-\frac{\lambda_{\min}\left(A^TA\right)}{\|A\|_F^2}<1.\end{aligned} \tag{3.19}$$

Now, to complete the proof, what we need to do is to prove the convergence of the sequence $\{x_k\}_{k\ge0}$.

It follows from (3.5), (3.11) and (3.16) that

$$\|x_{k+1}-x_\star\|_2^2\le\left(1-\gamma_k\frac{\left\|A_{\tau_k}\right\|_F^2}{\lambda_{\max}\left(A_{\tau_k}A_{\tau_k}^T\right)}\frac{\lambda_{\min}\left(A^TA\right)}{\|A\|_F^2}\right)\|x_k-x_\star\|_2^2,\quad k\ge0, \tag{3.20}$$

i.e., (3.1) holds. By (3.19) and (3.20), we can get

$$\|x_{k+1}-x_\star\|_2^2\le\left(1-\frac{\lambda_{\min}\left(A^TA\right)}{\|A\|_F^2}\right)\|x_k-x_\star\|_2^2,\quad k\ge0,$$

which deduces

$$\|x_{k+1} - x_\star\|_2^2 \le \left(1 - \frac{\lambda_{\min}\left(A^T A\right)}{\|A\|_F^2}\right)^{k+1} \|x_0 - x_\star\|_2^2, \quad k \ge 0 \tag{3.21}$$

via the mathematical induction method. Here, $1 - \frac{\lambda_{\min}\left(A^T A\right)}{\|A\|_F^2}$ is a nonnegative number less than 1 by (3.19). (3.21) shows us that the sequence $\{x_k\}_{k\ge 0}$ converges to $x_\star$ as $k \to \infty$. □

## 4 Experimental results

In this section, we will make numerical experiments on consistent linear systems with three types of matrices for comparing the fast deterministic block Kaczmarz (FDBK) method with the following methods:

- RaBK: the randomized average block Kaczmarz method [16];
- GBK: the greedy block Kaczmarz method [20].

For the RaBK method, we consider four cases appeared in [16], and use the same sampling methods and choices of blocks, stepsizes and weights. The details are listed below.

The meanings of the partition sampling, $L_k$ and $\lambda_{\max}^{\text{block}}$ in Table 1 are described in the following.

- The partition sampling means that $\tau_k$ used is selected randomly from the row partition $\mathcal{P}_s = \{\sigma_1, \sigma_2, \ldots, \sigma_s\}$ at $(k+1)$th iteration, where $s = \lceil \|A\|_2^2 \rceil$, and
$$\sigma_i = \left\{ \left\lfloor (i-1)\frac{m}{s} \right\rfloor + 1, \left\lfloor (i-1)\frac{m}{s} \right\rfloor + 2, \ldots, \left\lfloor i\frac{m}{s} \right\rfloor \right\}, \quad i = 1, 2, \ldots, s.$$
- $L_k = \sum_{i\in\tau_k} \omega_i^k \left(A^{(i)} x_k - b^{(i)}\right)^2 \Big/ \left\| \sum_{i\in\tau_k} \omega_i^k \left(A^{(i)} x_k - b^{(i)}\right) (A^{(i)})^T \right\|_2^2$.
- $\lambda_{\max}^{\text{block}} = \max_{\tau_k\in\mathcal{P}} \lambda_{\max}(A_{\tau_k}^T A_{\tau_k})$, where $\mathcal{P}_s$ is the partition defined above.

For the GBK method, we take the parameter $\delta_k$ in (1.4) as the same value as that used in [20] and also use the CGLS algorithm instead of calculating the

**Table 1** The sampling methods and parameters of four cases of the RaBK method in [16]

| Method | Sampling method | Block size $\tau$ | Stepsize $\alpha_k$ | Weight $\omega_i^k$ |
|---|---|---|---|---|
| RaBK-c | Uniform sampling | 10 | 1.95 | $\frac{1}{\tau}$ |
| RaBK-a | Uniform sampling | 10 | $1.95 L_k$ | $\frac{1}{\tau}$ |
| RaBK-e-paved | Partition sampling | $\left\lfloor \frac{m}{\|A\|_2^2} \right\rfloor$ | $1.95\tau / \lambda_{\max}^{\text{block}}$ | $\frac{1}{\tau}$ |
| RaBK-a-paved | Partition sampling | $\left\lfloor \frac{m}{\|A\|_2^2} \right\rfloor$ | $1.95 L_k$ | $\frac{1}{\tau}$ |

Moore-Penrose inverse $A_{\tau_k}^{\dagger}$ at each iteration. The details are described as follows. For each $k \geq 0$,

- $\delta_k = \frac{1}{2} + \frac{1}{2} \frac{\|b - Ax_k\|_2^2}{\|A\|_F^2} \left( \max_{1 \leq i \leq m} \left\{ \frac{|b^{(i)} - A^{(i)} x_k|^2}{\|A^{(i)}\|_2^2} \right\} \right)^{-1}$,
- the stopping criterion of the CGLS algorithm is taken as $10^{-4}$.

The stopping criterion used for the CGLS algorithm is an appropriate error criterion to ensure that the GBK method can be carried out. In fact, on the one hand, the CPU time of the GBK method will increase with the decrease of the stopping criterion of the CGLS algorithm, and on the other hand, when the stopping criterion of the CGLS algorithm is taken as $10^{-3}$, the GBK method cannot converge for the overdetermined matrix generated by using the MATLAB function **randn**(1000,800) and used in the numerical experiments.

Three types of coefficient matrices of the consistent linear systems used in the numerical experiments are

- forty dense and well-conditioned matrices, generated by using the MATLAB function **randn**;
- twenty full-rank sparse matrices, selected from the SuiteSparse Matrix Collection (formerly known as the University of Florida Sparse Matrix Collection) [7];
- six rank-deficient sparse matrices, selected from the SuiteSparse Matrix Collection [7].

Thus, the dense matrices used are generated randomly, while two kinds of sparse matrices used are of deterministic.

In the numerical experiments, all zero rows, which might be appeared in four matrices, will be removed, and the lengths of all nonzero rows in each matrix are normalized to 1.

In our implementations, to ensure that each linear system (1.1) with the selected matrix $A$ is consistent, a vector $y \in \mathbb{R}^n$ with its entries randomly generated by the MATLAB function **randn** is formed at first, and $b \in \mathbb{R}^m$, the right-hand-side of the linear system (1.1) is then set to be $b = Ay$.

For all methods, the iterations are started from $x_0 = 0$, and terminated once the *relative solution error* (RSE) satisfies

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

or the number of iteration steps exceeds 200000.

All numerical experiments are implemented in MATLAB (version R2019b) and executed on an Intel(R) Core(TM) i5-10210U CPU 1.60GHz computer with 12 GB of RAM and Windows 10 operating system.

We report the performance of the methods in terms of the number of iteration steps (denoted by "IT") and the CPU time in seconds (denoted by "CPU") in Tables 2, 3, 4, 5, 6, 7, and 8, where each "IT" and "CPU" of the randomized methods are the arithmetic mean of the results obtained by repeatedly running the corresponding method 50 times.

**Table 2** Numerical results corresponding to the dense matrices $A$ of Type I with $m > n$

| $m$ | | | 1000 | 2000 | 3000 | 4000 | 5000 |
|---|---|---|---|---|---|---|---|
| $n = 100$ | RaBK-c | IT | 588.3 | 483.7 | 445.6 | 426.8 | 433.3 |
| | | CPU | 0.0093 | 0.0083 | 0.0080 | 0.0080 | 0.0091 |
| | RaBK-a | IT | 1504.5 | 1509.2 | 1514.5 | 1514.8 | 1516.9 |
| | | CPU | 0.0165 | 0.0171 | 0.0174 | 0.0184 | 0.0196 |
| | RaBK-e-paved | IT | 35.0 | 29.1 | 27.0 | 25.8 | 24.9 |
| | | CPU | 0.0076 | 0.0123 | 0.0162 | 0.0213 | 0.0259 |
| | RaBK-a-paved | IT | 362.9 | 338.9 | 325.1 | 319.4 | 313.1 |
| | | CPU | 0.0104 | 0.0137 | 0.0140 | 0.0165 | 0.0178 |
| | GBK | IT | 20 | 14 | 12 | 9 | 10 |
| | | CPU | 0.0021 | 0.0026 | 0.0035 | 0.0043 | 0.0051 |
| | FDBK | IT | 23 | 16 | 12 | 12 | 11 |
| | | CPU | 0.0011 | 0.0020 | 0.0023 | 0.0031 | 0.0039 |
| | speed-up_RaBK-c | | 8.45 | 4.15 | 3.48 | 2.58 | 2.33 |
| | speed-up_RaBK-a | | 15.00 | 8.55 | 7.57 | 5.94 | 5.03 |
| | speed-up_RaBK-e-paved | | 6.91 | 6.15 | 7.04 | 6.87 | 6.64 |
| | speed-up_RaBK-a-paved | | 9.45 | 6.85 | 6.09 | 5.32 | 4.56 |
| | speed-up_GBK | | 1.91 | 1.30 | 1.52 | 1.39 | 1.31 |
| $n = 150$ | RaBK-c | IT | 1028.8 | 799.0 | 713.5 | 682.2 | 670.4 |
| | | CPU | 0.0214 | 0.0171 | 0.0169 | 0.0182 | 0.0197 |
| | RaBK-a | IT | 2223.8 | 2220.4 | 2226.9 | 2222.7 | 2223.0 |
| | | CPU | 0.0306 | 0.0314 | 0.0347 | 0.0394 | 0.0440 |
| | RaBK-e-paved | IT | 42.6 | 32.0 | 29.1 | 27.8 | 26.6 |
| | | CPU | 0.0143 | 0.0203 | 0.0295 | 0.0362 | 0.0438 |
| | RaBK-a-paved | IT | 400.3 | 350.1 | 341.2 | 330.7 | 326.1 |
| | | CPU | 0.0269 | 0.0284 | 0.0315 | 0.0345 | 0.0369 |
| | GBK | IT | 28 | 17 | 14 | 12 | 12 |
| | | CPU | 0.0036 | 0.0043 | 0.0052 | 0.0062 | 0.0090 |
| | FDBK | IT | 33 | 21 | 17 | 14 | 14 |
| | | CPU | 0.0025 | 0.0030 | 0.0039 | 0.0053 | 0.0069 |
| | speed-up_RaBK-c | | 8.56 | 5.70 | 4.33 | 3.43 | 2.86 |
| | speed-up_RaBK-a | | 12.24 | 10.47 | 8.90 | 7.43 | 6.38 |
| | speed-up_RaBK-e-paved | | 5.72 | 6.77 | 7.56 | 6.83 | 6.35 |
| | speed-up_RaBK-a-paved | | 10.76 | 9.47 | 8.08 | 6.51 | 5.35 |
| | speed-up_GBK | | 1.44 | 1.43 | 1.33 | 1.17 | 1.30 |

In the following tables, the item "$>$ 200000" represents that the number of iteration steps exceeds 200000. In this case, the corresponding CPU time is expressed as "$--$".

**Table 3** Numerical results corresponding to the dense matrices $A$ of Type I with $m > n$

| $m$ | | | 1000 | 2000 | 3000 | 4000 | 5000 |
|---|---|---|---|---|---|---|---|
| $n = 500$ | RaBK-c | IT | 12143.0 | 4645.9 | 3594.3 | 3211.4 | 2925.3 |
| | | CPU | 1.2105 | 0.6349 | 0.5741 | 0.5421 | 0.5213 |
| | RaBK-a | IT | 7326.4 | 7215.9 | 7199.2 | 7198.6 | 7199.3 |
| | | CPU | 0.4555 | 0.6223 | 0.7497 | 0.7195 | 0.7302 |
| | RaBK-e-paved | IT | 172.0 | 60.5 | 43.8 | 39.1 | 36.1 |
| | | CPU | 0.1343 | 0.1525 | 0.2179 | 0.2831 | 0.3426 |
| | RaBK-a-paved | IT | 532.8 | 440.1 | 407.9 | 391.9 | 383.0 |
| | | CPU | 0.1859 | 0.2199 | 0.2530 | 0.5661 | 0.6019 |
| | GBK | IT | 251 | 69 | 43 | 39 | 33 |
| | | CPU | 0.0784 | 0.0523 | 0.0543 | 0.0634 | 0.0708 |
| | FDBK | IT | 271 | 72 | 45 | 39 | 35 |
| | | CPU | 0.0322 | 0.0379 | 0.0416 | 0.0515 | 0.0563 |
| | speed-up_RaBK-c | | 37.69 | 16.75 | 13.80 | 10.53 | 9.26 |
| | speed-up_RaBK-a | | 14.15 | 16.42 | 18.02 | 13.97 | 12.97 |
| | speed-up_RaBK-e-paved | | 4.17 | 4.02 | 5.24 | 5.50 | 6.09 |
| | speed-up_RaBK-a-paved | | 5.77 | 5.80 | 6.08 | 10.99 | 10.69 |
| | speed-up_GBK | | 2.43 | 1.38 | 1.31 | 1.23 | 1.26 |
| $n = 800$ | RaBK-c | IT | 123550.0 | 13029.5 | 7997.8 | 6561.4 | 5675.4 |
| | | CPU | 23.1951 | 3.4222 | 2.1305 | 1.8914 | 1.7800 |
| | RaBK-a | IT | 15336.0 | 11614.8 | 11499.3 | 11495.0 | 11467.5 |
| | | CPU | 1.7558 | 1.7511 | 1.8781 | 1.7543 | 1.9439 |
| | RaBK-e-paved | IT | 1228.0 | 110.3 | 64.6 | 50.9 | 44.2 |
| | | CPU | 2.0417 | 0.7696 | 0.7859 | 0.7872 | 0.7936 |
| | RaBK-a-paved | IT | 979.7 | 512.1 | 461.5 | 435.6 | 421.8 |
| | | CPU | 1.6381 | 1.2537 | 1.3348 | 1.3066 | 1.3824 |
| | GBK | IT | 2419 | 167 | 83 | 67 | 49 |
| | | CPU | 1.2390 | 0.2141 | 0.1805 | 0.1947 | 0.1962 |
| | FDBK | IT | 2511 | 184 | 91 | 72 | 52 |
| | | CPU | 0.6833 | 0.1702 | 0.1460 | 0.1612 | 0.1592 |
| | speed-up_RaBK-c | | 33.95 | 20.11 | 14.59 | 11.73 | 11.18 |
| | speed-up_RaBK-a | | 2.57 | 10.29 | 12.86 | 10.88 | 12.21 |
| | speed-up_RaBK-e-paved | | 2.99 | 4.52 | 5.38 | 4.88 | 4.98 |
| | speed-up_RaBK-a-paved | | 2.40 | 7.37 | 9.14 | 8.11 | 8.68 |
| | speed-up_GBK | | 1.81 | 1.26 | 1.24 | 1.21 | 1.23 |

To compare the degree of convergence speeds, the speed-up of the FDBK method against the other one, defined by

$$\text{speed-up\_Method} = \frac{\text{CPU of Method}}{\text{CPU of FDBK}},$$

**Table 4** Numerical results corresponding to the dense matrices $A$ of Type I with $m < n$

| $n$ | | | 1000 | 2000 | 3000 | 4000 | 5000 |
|---|---|---|---|---|---|---|---|
| $m = 100$ | RaBK-c | IT | 556.8 | 473.9 | 447.3 | 439.0 | 425.6 |
| | | CPU | 0.0759 | 0.1191 | 0.1658 | 0.2287 | 0.3086 |
| | RaBK-a | IT | 1410.3 | 1410.4 | 1412.0 | 1408.3 | 1405.3 |
| | | CPU | 0.1192 | 0.2082 | 0.3116 | 0.4234 | 0.5697 |
| | RaBK-e-paved | IT | 112.5 | 137.6 | 114.5 | 137.4 | 156.0 |
| | | CPU | 0.1355 | 0.7212 | 1.9664 | 4.2535 | 7.6041 |
| | RaBK-a-paved | IT | 276.4 | 277.0 | 276.3 | 276.7 | 277.9 |
| | | CPU | 0.0497 | 0.0948 | 0.3997 | 0.5240 | 0.6441 |
| | GBK | IT | 37 | 30 | 26 | 25 | 23 |
| | | CPU | 0.0093 | 0.0103 | 0.0110 | 0.0150 | 0.0190 |
| | FDBK | IT | 39 | 29 | 26 | 27 | 24 |
| | | CPU | 0.0015 | 0.0031 | 0.0033 | 0.0043 | 0.0053 |
| | speed-up_RaBK-c | | 50.60 | 38.42 | 50.24 | 53.19 | 58.23 |
| | speed-up_RaBK-a | | 79.47 | 67.16 | 94.42 | 98.47 | 107.49 |
| | speed-up_RaBK-e-paved | | 90.23 | 232.65 | 595.88 | 989.19 | 1434.74 |
| | speed-up_RaBK-a-paved | | 33.13 | 30.58 | 121.12 | 121.86 | 121.53 |
| | speed-up_GBK | | 6.20 | 3.32 | 3.33 | 3.49 | 3.58 |
| $m = 150$ | RaBK-c | IT | 1065.2 | 785.3 | 722.5 | 681.0 | 656.6 |
| | | CPU | 0.1619 | 0.2267 | 0.3402 | 0.4649 | 0.5533 |
| | RaBK-a | IT | 2130.3 | 2117.3 | 2114.3 | 2118.9 | 2118.8 |
| | | CPU | 0.1894 | 0.3498 | 0.5469 | 0.8814 | 1.0814 |
| | RaBK-e-paved | IT | 85.7 | 107.0 | 101.3 | 153.1 | 122.9 |
| | | CPU | 0.1435 | 0.8174 | 2.0681 | 4.4995 | 7.6442 |
| | RaBK-a-paved | IT | 277.1 | 279.1 | 277.5 | 278.1 | 276.7 |
| | | CPU | 0.0815 | 0.4105 | 0.5990 | 0.7888 | 0.9709 |
| | GBK | IT | 51 | 36 | 29 | 29 | 28 |
| | | CPU | 0.0106 | 0.0198 | 0.0209 | 0.0289 | 0.0356 |
| | FDBK | IT | 50 | 36 | 31 | 31 | 28 |
| | | CPU | 0.0025 | 0.0043 | 0.0090 | 0.0126 | 0.0171 |
| | speed-up_RaBK-c | | 64.76 | 52.72 | 37.80 | 36.90 | 32.36 |
| | speed-up_RaBK-a | | 75.76 | 81.35 | 60.77 | 69.95 | 63.24 |
| | speed-up_RaBK-e-paved | | 57.40 | 190.09 | 229.79 | 357.10 | 447.03 |
| | speed-up_RaBK-a-paved | | 32.60 | 95.47 | 66.56 | 62.60 | 56.78 |
| | speed-up_GBK | | 4.24 | 4.60 | 2.32 | 2.29 | 2.08 |

is also reported. Clearly, the greater the "speed-up_Method" is than 1, the faster the convergence speed of the FDBK method against the corresponding "Method" is.

**Dense and well-conditioned matrices** Numerical results corresponding to these forty matrices of Type I are listed in Tables 2, 3, 4, and 5, where Tables 2 and 3 stands for

**Table 5** Numerical results corresponding to the dense matrices $A$ of Type I with $m < n$

| $n$ | | | 1000 | 2000 | 3000 | 4000 | 5000 |
|---|---|---|---|---|---|---|---|
| $m = 500$ | RaBK-c | IT | 11764.0 | 4877.3 | 3575.4 | 3182.7 | 2893.2 |
| | | CPU | 2.6910 | 2.7763 | 2.9999 | 3.7439 | 4.2782 |
| | RaBK-a | IT | 7265.1 | 7141.5 | 7121.2 | 7104.6 | 7108.2 |
| | | CPU | 0.9201 | 2.1601 | 3.4666 | 4.5042 | 5.5678 |
| | RaBK-e-paved | IT | 134.6 | 79.1 | 76.7 | 87.4 | 89.2 |
| | | CPU | 0.4022 | 1.2886 | 2.3977 | 4.9434 | 8.3194 |
| | RaBK-a-paved | IT | 427.3 | 421.9 | 279.0 | 277.5 | 276.9 |
| | | CPU | 0.7075 | 1.3900 | 1.9680 | 2.5381 | 3.1891 |
| | GBK | IT | 302 | 102 | 66 | 56 | 49 |
| | | CPU | 0.1120 | 0.1094 | 0.1340 | 0.1785 | 0.2088 |
| | FDBK | IT | 313 | 106 | 68 | 58 | 51 |
| | | CPU | 0.0327 | 0.0588 | 0.0707 | 0.0889 | 0.1090 |
| | speed-up_RaBK-c | | 82.29 | 47.22 | 42.43 | 42.11 | 39.25 |
| | speed-up_RaBK-a | | 28.14 | 36.74 | 49.03 | 50.67 | 51.08 |
| | speed-up_RaBK-e-paved | | 12.30 | 21.91 | 33.91 | 55.61 | 76.32 |
| | speed-up_RaBK-a-paved | | 21.64 | 23.64 | 27.84 | 28.55 | 29.26 |
| | speed-up_GBK | | 3.43 | 1.86 | 1.90 | 2.01 | 1.92 |
| $m = 800$ | RaBK-c | IT | 120710.5 | 14295.0 | 7697.8 | 6492.2 | 5742.0 |
| | | CPU | 38.5108 | 8.8431 | 6.9098 | 8.2531 | 9.4483 |
| | RaBK-a | IT | 15741.0 | 11549.6 | 11410.3 | 11385.0 | 11382.5 |
| | | CPU | 2.4834 | 3.8037 | 5.4229 | 7.2930 | 9.1952 |
| | RaBK-e-paved | IT | 1261.2 | 96.9 | 80.5 | 92.0 | 60.0 |
| | | CPU | 2.3818 | 1.5653 | 3.4208 | 6.8780 | 8.2795 |
| | RaBK-a-paved | IT | 997.2 | 424.8 | 422.4 | 419.8 | 277.7 |
| | | CPU | 1.9628 | 1.9260 | 2.9184 | 3.7890 | 4.7668 |
| | GBK | IT | 2657 | 244 | 113 | 87 | 70 |
| | | CPU | 1.6345 | 0.3857 | 0.3520 | 0.4195 | 0.4229 |
| | FDBK | IT | 2764 | 243 | 116 | 90 | 73 |
| | | CPU | 0.7717 | 0.2246 | 0.2007 | 0.2181 | 0.2417 |
| | speed-up_RaBK-c | | 49.90 | 39.37 | 34.43 | 37.84 | 39.09 |
| | speed-up_RaBK-a | | 3.22 | 16.94 | 27.02 | 33.44 | 38.04 |
| | speed-up_RaBK-e-paved | | 3.09 | 6.97 | 17.04 | 31.54 | 34.26 |
| | speed-up_RaBK-a-paved | | 2.54 | 8.58 | 14.54 | 17.37 | 19.72 |
| | speed-up_GBK | | 2.12 | 1.72 | 1.75 | 1.92 | 1.75 |

the case of thin (or tall) matrices while Tables 4 and 5 for the case of fat matrices. The numbers of columns of the matrices used in Tables 2 and 3 are fixed with 100, 150, 500 and 800, respectively, and the numbers of rows of the matrices used in Tables 4 and 5 are fixed with 100, 150, 500 and 800, respectively.

From Tables 2, 3, 4, and 5, we can observe the following phenomena:

- FDBK versus GBK: These two methods use the same criterion to construct the block at each iteration. Since the linear combination of columns of $A_{\tau_k}^T$ is used directly in the FDBK method, the decrease of the norm of the residual of the FDBK method might be not greater than that of the GBK method at each iteration, and the number of the iteration steps would be greater than that of the GBK method. But the decrement of the number of floating-point operations leads the FDBK method to have a faster convergence and own less CPU time. More specifically, the speed-up of the FDBK method against the GBK method is distributed from 1.17 to 6.20.
- FDBK versus RaBK: Compared with the four special cases of the RaBK method, data in tables shows us that the FDBK method has great advantages in terms of the CPU time in all cases and the number of iteration steps in most cases. In detail, for the matrices of $1000 \times 500$, $2000 \times 500$, $3000 \times 500$, $2000 \times 800$, $3000 \times 800$, $4000 \times 800$, $5000 \times 800$, $500 \times 1000$, $500 \times 2000$, $800 \times 2000$, $800 \times 3000$ and $800 \times 5000$, the RaBK-e-paved method owns the least number of iteration steps, and for the matrices of $1000 \times 800$ and $800 \times 1000$, the RaBK-a-paved method owns the least number of iteration steps. In these tables, the speed-up of the FDBK method against the RaBK-c method is distributed from 2.33 to 82.29, the speed-up of the FDBK method against the RaBK-a method is distributed from 2.57 to 107.49, the speed-up of the FDBK method against the RaBK-e-paved method is distributed from 2.99 to 1434.74, and the speed-up of the FDBK method against the RaBK-a-paved method is distributed from 2.40 to 121.86. Therefore, the submatrices, stepsizes and weights chosen in the FDBK method are much effective.

**Full-rank sparse matrices selected from** [7] There are five thin (or tall) matrices, five square matrices, and ten fat matrices, which originate in different applications such as linear programming problem, combinatorial problem, least-squares problem, directed weighted graph, weighted bipartite graph and undirected weighted graph [7]. These twenty matrices of Type II are determined uniquely.

Numerical results corresponding to these matrices of Type II are listed in Tables 6 and 7, where "density", the sparsity of these matrices, is computed as usual via the following format:

$$\text{density} = \frac{\text{number of nonzeros of an } m \times n \text{ matrix}}{m \times n},$$

and "cond($A$)", the Euclidean condition number of $A$, is computed by using the matlab function **cond**.

Since the matrix *WorldCities* has zero rows, in the numerical experiments, we removed these zero rows from the matrix. Clearly, this operation does not influence the solution of the corresponding linear system.

From Tables 6 and 7, we can see that the phenomena here are similar to those observed from Tables 2, 3, 4, and 5:

- FDBK versus GBK: The FDBK method performs better than the GBK method especially in terms of the CPU time, although the number of the iteration steps of

**Table 6** Numerical results corresponding to the full-rank sparse matrices $A$ of Type II with $m \geq n$

| name | | ash958 | ash608 | WorldCities[a] | ash219 | Cities |
|---|---|---|---|---|---|---|
| $m \times n$ | | $958 \times 292$ | $608 \times 188$ | $315 \times 100$ | $219 \times 85$ | $55 \times 46$ |
| density | | 0.68% | 1.06% | 23.87% | 2.35% | 53.04% |
| cond(A) | | 3.20 | 3.37 | 66.00 | 3.02 | 207.15 |
| RaBK-c | IT | 2685.0 | 2055.3 | 18133.0 | 798.1 | 109750.2 |
| | CPU | 0.1465 | 0.0534 | 0.2751 | 0.0109 | 1.0085 |
| RaBK-a | IT | 4276.1 | 2746.6 | 5738.1 | 1273.5 | 66455.0 |
| | CPU | 0.1487 | 0.0458 | 0.0628 | 0.0129 | 0.4762 |
| RaBK-e-paved | IT | 318.1 | 225.5 | 16009.0 | 225.0 | $> 200000$ |
| | CPU | 0.0635 | 0.0217 | 0.1641 | 0.0048 | $--$ |
| RaBK-a-paved | IT | 1242.7 | 1069.2 | 7585.9 | 918.7 | $> 200000$ |
| | CPU | 0.1965 | 0.0827 | 0.0731 | 0.0142 | $--$ |
| GBK | IT | 30 | 28 | 990 | 30 | 31748 |
| | CPU | 0.0152 | 0.0034 | 0.0934 | 0.0016 | 2.7097 |
| FDBK | IT | 58 | 43 | 1894 | 38 | 55881 |
| | CPU | 0.0051 | 0.0023 | 0.0336 | 0.0009 | 0.2463 |
| speed-up_RaBK-c | | 28.73 | 23.22 | 8.19 | 12.28 | 4.09 |
| speed-up_RaBK-a | | 29.16 | 19.91 | 1.87 | 14.54 | 1.93 |
| speed-up_RaBK-e-paved | | 12.45 | 9.43 | 4.88 | 5.41 | $--$ |
| speed-up_RaBK-a-paved | | 38.53 | 35.96 | 2.18 | 16.00 | $--$ |
| speed-up_GBK | | 2.98 | 1.48 | 2.78 | 1.80 | 11.00 |
| name | | Trefethen_700 | Trefethen_300 | Trefethen_20 | cage5 | Stranke94 |
| $m \times n$ | | $700 \times 700$ | $300 \times 300$ | $20 \times 20$ | $37 \times 37$ | $10 \times 10$ |
| density | | 2.58% | 5.20% | 39.50% | 17.02% | 90.00% |
| cond(A) | | 4710.39 | 1772.69 | 63.09 | 15.42 | 51.73 |
| RaBK-c | IT | 5850.8 | 7250.3 | 706.8 | 1250.1 | 9496.0 |
| | CPU | 0.7359 | 0.3998 | 0.0051 | 0.0104 | 0.0550 |
| RaBK-a | IT | 9938.0 | 4232.5 | 288.5 | 538.4 | 7348.0 |
| | CPU | 0.7901 | 0.1479 | 0.0018 | 0.0036 | 0.0330 |
| RaBK-e-paved | IT | 194.2 | 198.8 | 262.7 | 353.0 | $> 200000$ |
| | CPU | 0.3546 | 0.0394 | 0.0018 | 0.0026 | $--$ |
| RaBK-a-paved | IT | 420.9 | 417.6 | 423.9 | 571.5 | $> 200000$ |
| | CPU | 0.6014 | 0.0614 | 0.0026 | 0.0042 | $--$ |
| GBK | IT | 35 | 37 | 135 | 132 | 4311 |
| | CPU | 0.0175 | 0.0039 | 0.0014 | 0.0016 | 0.0932 |
| FDBK | IT | 44 | 42 | 132 | 147 | 6977 |
| | CPU | 0.0105 | 0.0023 | 0.0005 | 0.0006 | 0.0169 |
| speed-up_RaBK-c | | 70.09 | 173.83 | 10.81 | 17.13 | 3.25 |
| speed-up_RaBK-a | | 75.25 | 64.30 | 3.82 | 5.93 | 1.95 |
| speed-up_RaBK-e-paved | | 33.77 | 17.13 | 3.82 | 4.28 | $--$ |
| speed-up_RaBK-a-paved | | 57.28 | 26.70 | 5.51 | 6.92 | $--$ |
| speed-up_GBK | | 1.67 | 1.70 | 2.97 | 2.63 | 5.51 |

[a]The matrix *WorldCities* has two zero rows, which are removed

**Table 7** Numerical results corresponding to the full-rank sparse matrices $A$ of Type II with $m < n$

| name | | df2177 | cari | model1 | nemsafm | bibd_17_8 |
|---|---|---|---|---|---|---|
| $m \times n$ | | 630 × 10358 | 400 × 1200 | 362 × 798 | 334 × 2348 | 136 × 24310 |
| density | | 0.34% | 31.83% | 1.05% | 0.36% | 20.59% |
| cond(A) | | 2.01 | 3.13 | 17.57 | 4.77 | 9.04 |
| RaBK-c | IT | 2953.4 | 1641.4 | 12733.0 | 2548.7 | 1417.8 |
| | CPU | 9.8847 | 0.3651 | 1.6393 | 1.3547 | 9.3782 |
| RaBK-a | IT | 8947.2 | 5671.6 | 5230.5 | 4734.6 | 2068.8 |
| | CPU | 13.3825 | 0.7288 | 0.4143 | 1.4091 | 7.9277 |
| RaBK-e-paved | IT | 219.0 | 1120.3 | 240.9 | 169.1 | 1235.5 |
| | CPU | 81.2929 | 1.1274 | 0.1708 | 1.6436 | 12532.2438 |
| RaBK-a-paved | IT | 418.0 | 1419.5 | 423.6 | 416.5 | 4498.1 |
| | CPU | 8.5530 | 0.3169 | 0.1663 | 1.0567 | 8.2525 |
| GBK | IT | 42 | 67 | 357 | 66 | 242 |
| | CPU | 0.5063 | 0.0353 | 0.0702 | 0.0566 | 7.5696 |
| FDBK | IT | 43 | 70 | 379 | 66 | 260 |
| | CPU | 0.2497 | 0.0102 | 0.0184 | 0.0256 | 0.6151 |
| speed-up_RaBK-c | | 39.59 | 35.79 | 89.09 | 52.92 | 15.25 |
| speed-up_RaBK-a | | 53.59 | 71.45 | 22.52 | 55.04 | 12.89 |
| speed-up_RaBK-e-paved | | 325.56 | 110.53 | 9.28 | 64.20 | 20374.32 |
| speed-up_RaBK-a-paved | | 34.25 | 31.07 | 9.04 | 41.28 | 13.42 |
| speed-up_GBK | | 2.03 | 3.46 | 3.82 | 2.21 | 12.31 |
| name | | crew1 | bibd_16_8 | bibd_13_6 | refine | Trec8 |
| $m \times n$ | | 135 × 6469 | 120 × 12870 | 78 × 1716 | 29 × 62 | 23 × 84 |
| density | | 5.38% | 23.33% | 19.23% | 8.51% | 28.43% |
| cond(A) | | 18.20 | 9.54 | 6.27 | 66.67 | 26.89 |
| RaBK-c | IT | 7297.4 | 1407.3 | 736.6 | 1027.6 | 3271.8 |
| | CPU | 9.1950 | 3.8979 | 0.1546 | 0.0105 | 0.0387 |
| RaBK-a | IT | 2206.5 | 1859.4 | 1161.2 | 431.3 | 1525.7 |
| | CPU | 1.4169 | 2.2453 | 0.1447 | 0.0036 | 0.0134 |
| RaBK-e-paved | IT | 1630.5 | 1039.6 | 484.3 | 285.7 | 3781.7 |
| | CPU | 57.8892 | 1634.2695 | 3.0749 | 0.0027 | 0.0283 |
| RaBK-a-paved | IT | 1493.1 | 4543.3 | 2462.5 | 587.0 | 2378.3 |
| | CPU | 1.0218 | 4.5207 | 0.2716 | 0.0049 | 0.0185 |
| GBK | IT | 516 | 268 | 125 | 170 | 688 |
| | CPU | 1.0068 | 3.9287 | 0.0590 | 0.0021 | 0.0122 |
| FDBK | IT | 1023 | 282 | 156 | 176 | 1012 |
| | CPU | 0.3866 | 0.2599 | 0.0049 | 0.0007 | 0.0033 |
| speed-up_RaBK-c | | 23.78 | 15.00 | 31.55 | 14.90 | 11.73 |
| speed-up_RaBK-a | | 3.67 | 8.64 | 29.53 | 5.11 | 4.06 |
| speed-up_RaBK-e-paved | | 149.74 | 6288.07 | 627.53 | 3.83 | 8.58 |
| speed-up_RaBK-a-paved | | 2.64 | 17.39 | 55.43 | 6.95 | 5.61 |
| speed-up_GBK | | 2.60 | 15.12 | 12.04 | 2.98 | 3.70 |

the GBK method is less than that of the FDBK method. Moreover, the speed-up of the FDBK method against the GBK method is distributed from 1.48 to 15.12.

- FDBK versus RaBK: Compared with the four special cases of the RaBK method, data in tables shows us that the FDBK method has great advantages in terms of the CPU time in all cases and the number of iteration steps except the case when the matrix is *model1*. In detail, for the matrix of *model1*, the RaBK-e-paved method owns the least number of iteration steps. For the matrices of *Cities* and *Stranke94*, the RaBK-e-paved and RaBK-a-paved methods failed due to the number of the iteration steps exceeding 200000. Specifically, the speed-up of the FDBK method against the RaBK-c method is distributed from 3.25 to 173.83, and the speed-up of the FDBK method against the RaBK-a method is distributed from 1.87 to 75.25. When the numbers of iteration steps do not exceed 200000, the speed-up of the FDBK method against the RaBK-e-paved method is distributed from 3.82 to 20374.32, and the speed-up of the FDBK method against the RaBK-a-paved method is distributed from 2.18 to 57.28.

**Rank-deficient sparse matrices selected from** [7] The matrices used here, originated in different applications such as combinatorial problem, undirected weighted graph, directed graph and bipartite graph [7], include a thin (or tall) matrix, two square matrices and three fat matrices. As is in the previous part, these six matrices of Type III are determined uniquely.

Numerical results corresponding to these matrices of Type III are listed in Table 8, where "density" and "cond($A$)" are computed via the method in the previous part.

Since the matrices *relat6*, *GD00_a* and *GL7d26* have zero rows, respectively, we removed these zero rows from the matrices in the numerical experiments in the same reason described in the previous part.

From Table 8, we can also observe the similar phenomena to those observed from Tables 2, 3, 4, 5, 6, and 7:

- FDBK versus GBK: The FDBK method performs better than the GBK method especially in terms of the CPU time, although the number of the iteration steps of the FDBK method is greater than that of the GBK method. Moreover, the speed-up of the FDBK method against the GBK method is distributed from 1.45 to 2.51.
- FDBK versus RaBK: The FDBK method outperforms the four special cases of the RaBK method in terms of both the number of iteration steps and the CPU time. Specifically, in the table, the speed-up of the FDBK method against the RaBK-c method is distributed from 5.92 to 56.55, the speed-up of the FDBK method against the RaBK-a method is distributed from 1.52 to 31.80, the speed-up of the FDBK method against the RaBK-e-paved method is distributed from 5.02 to 44.69, and the speed-up of the FDBK method against the RaBK-a-paved method is distributed from 3.07 to 19.79.

All in all, data in Tables 2, 3, 4, 5, 6, 7, and 8 illustrates that the CPU time of the FDBK method is less than those of other tested methods, i.e., the FDBK method performs more efficiently than other tested methods especially in terms of the CPU time.

**Table 8** Numerical results corresponding to the rank-deficient matrices $A$ of Type III

| name | | relat6[a] | GD00_a[a] | Sandi_authors | Sandi_sandi | GL7d26[a] | n4c5-b10 |
|---|---|---|---|---|---|---|---|
| $m \times n$ | | $2340 \times 157$ | $352 \times 352$ | $86 \times 86$ | $314 \times 360$ | $305 \times 2798$ | $120 \times 630$ |
| density | | 2.21% | 0.37% | 3.35% | 0.54% | 0.87% | 1.75% |
| cond(A) | | Inf | Inf | 1.11e+18 | 1.20e+18 | 1.55e+19 | 2.37e+16 |
| RaBK-c | IT | 7449.0 | 8368.4 | 76351.0 | 17420.0 | 1411.6 | 454.0 |
| | CPU | 0.1765 | 0.5174 | 1.0025 | 1.1592 | 0.9594 | 0.0390 |
| RaBK-a | IT | 2110.7 | 1974.5 | 8552.8 | 4083.2 | 3896.3 | 1563.9 |
| | CPU | 0.0453 | 0.0838 | 0.0822 | 0.1768 | 1.4512 | 0.0954 |
| RaBK-e-paved | IT | 5568.0 | 5399.5 | 15393.0 | 853.4 | 172.4 | 221.3 |
| | CPU | 0.1856 | 0.2538 | 0.1396 | 0.1030 | 2.5163 | 0.0635 |
| RaBK-a-paved | IT | 2639.4 | 3409.3 | 8076.3 | 859.2 | 417.3 | 274.4 |
| | CPU | 0.0916 | 0.1481 | 0.0821 | 0.0970 | 1.1144 | 0.0439 |
| GBK | IT | 176 | 184 | 3706 | 212 | 35 | 28 |
| | CPU | 0.0598 | 0.0390 | 0.0565 | 0.0297 | 0.0821 | 0.0057 |
| FDBK | IT | 295 | 427 | 4507 | 388 | 36 | 30 |
| | CPU | 0.0298 | 0.0198 | 0.0225 | 0.0205 | 0.0563 | 0.0030 |
| speed-up_RaBK-c | | 5.92 | 26.13 | 44.56 | 56.55 | 17.04 | 13.00 |
| speed-up_RaBK-a | | 1.52 | 4.23 | 3.65 | 8.62 | 25.78 | 31.80 |
| speed-up_RaBK-e-paved | | 6.23 | 12.82 | 6.20 | 5.02 | 44.69 | 21.17 |
| speed-up_RaBK-a-paved | | 3.07 | 7.48 | 3.65 | 4.73 | 19.79 | 14.63 |
| speed-up_GBK | | 2.01 | 1.97 | 2.51 | 1.45 | 1.46 | 1.90 |

[a]The matrices *relat6*, *GD00_a* and *GL7d26* have zero rows, which are removed in the numerical experiments

## 5 Conclusions

In this paper, we propose the fast deterministic block Kaczmarz (FDBK) method for solving large-scale consistent linear systems. It is proved that the FDBK method converges to the least-norm solutions of the linear systems whether they are overdetermined or underdetermined. Numerical results illustrate that the FDBK method provides significant computational advantages and is more efficient, which means that the FDBK method is a competitive block Kaczmarz-type method for consistent linear system.

**Funding** This work was supported by NSFC under grant number 11871430.

## References

1. Bai, Z.-Z., Liu, X.-G.: On the Meany inequality with applications to convergence analysis of several row-action iteration methods. Numer. Math. **124**(2), 215–236 (2013)

2. Bai, Z.-Z., Wu, W.-T.: On convergence rate of the randomized Kaczmarz method. Linear Algebra Appl. **553**, 252–269 (2018)
3. Bai, Z.-Z., Wu, W.-T.: On greedy randomized Kaczmarz method for solving large sparse linear systems. SIAM J. Sci. Comput. **40**(1), A592–A606 (2018)
4. Bai, Z.-Z., Wu, W.-T.: On relaxed greedy randomized Kaczmarz methods for solving large sparse linear systems. Appl. Math. Lett. **83**, 21–26 (2018)
5. Bauschke, H.H., Combettes, P.L., Kruk, S.G.: Extrapolation algorithm for affine-convex feasibility problems. Numer. Algorithms **41**(3), 239–274 (2006)
6. Censor, Y.: Row-action methods for huge and sparse systems and their applications. SIAM Rev. **23**(4), 444–466 (1981)
7. Davis, T.A., Hu, Y.: The University of Florida sparse matrix collection. ACM Trans. Math. Software **38**(1), 1–25 (2011)
8. Du, K., Si, W.-T., Sun, X.-H.: Randomized extended average block Kaczmarz for solving least squares. SIAM J. Sci. Comput. **42**(6), A3541–A3559 (2020)
9. Eggermont, G.T., Herman, P.P.B., Lent, A.: Iterative algorithms for large partitioned linear systems, with applications to image reconstruction. Linear Algebra Appl. **40**, 37–67 (1981)
10. Elfving, T.: Block-iterative methods for consistent and inconsistent linear equations. Numer. Math. **35**(1), 1–12 (1980)
11. Gower, R.M., Richtárik, P.: Randomized iterative methods for linear systems. SIAM J. Matrix Anal. Appl. **36**(4), 1660–1690 (2015)
12. Kaczmarz, S.: Angenäherte Auflösung von Systemen Linearer Gleichungen. Bull. Int. Acad. Polon. Sci. Lett. A **35**, 355–357 (1937)
13. Leventhal, D., Lewis, A.S.: Randomized methods for linear constraints: convergence rates and conditioning. Math. Oper. Res. **35**(3), 641–654 (2010)
14. Ma, A., Needell, D., Ramdas, A.: Convergence properties of the randomized extended Gauss-Seidel and Kaczmarz methods. SIAM J. Matrix Anal. Appl. **36**(4), 1590–1604 (2015)
15. Merzlyakov, Y.I.: On a relaxation method of solving systems of linear inequalities. USSR Comput. Math. Math. Phys. **2**(3), 504–510 (1963)
16. Necoara, I.: Faster randomized block Kaczmarz algorithms. SIAM J. Matrix Anal. Appl. **40**(4), 1425–1452 (2019)
17. Needell, D.: Randomized Kaczmarz solver for noisy linear systems. BIT **50**(2), 395–403 (2010)
18. Needell, D., Tropp, J.A.: Paved with good intentions: Analysis of a randomized block Kaczmarz method. Linear Algebra Appl. **441**(1), 199–221 (2014)
19. Needell, D., Zhao, R., Zouzias, A.: Randomized block Kaczmarz method with projection for solving least squares. Linear Algebra Appl. **484**, 322–343 (2015)
20. Niu, Y.-Q., Zheng, B.: A greedy block Kaczmarz algorithm for solving large-scale linear systems. Appl. Math. Lett. **104**, 106294 (2020)
21. Nutini, J., Sepehry, B., Laradji, I., Schmidt, M., Koepke, H., Virani, A.: Convergence rates for greedy Kaczmarz algorithms, and faster randomized Kaczmarz rules using the orthogonality graph. arXiv:1612.07838 (2016)
22. Pierra, G.: Decomposition through formalization in a product space. Math. Program. **28**(1), 96–115 (1984)
23. Popa, C.: Extensions of block-projections methods with relaxation parameters to inconsistent and rank-deficient least-squares problems. BIT **38**(1), 151–176 (1998)
24. Strohmer, T., Vershynin, R.: A randomized Kaczmarz algorithm with exponential convergence. J. Fourier Anal. Appl. **15**(2), 262–278 (2009)
25. Zouzias, A., Freris, N.M.: Randomized extended Kaczmarz for solving least-squares. SIAM J. Matrix Anal. Appl. **34**(2), 773–793 (2013)