Mathematics

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

AIMS Mathematics, 7(4): 6792–6806.
DOI:10.3934/math.2022378
Received: 03 November 2021
Revised: 04 January 2022
Accepted: 10 January 2022
Published: 26 January 2022

Research article

A greedy average block Kaczmarz method for the large scaled consistent system of linear equations

Li Wen[1,2], Feng Yin[1,2,*], Yimou Liao[1] and Guangxin Huang[3]

[1] College of Mathematics and Statistics, Sichuan University of Science and Engineering, Zigong 643000, China

[2] Sichuan Province University Key Laboratory of Bridge Non-destruction Detecting and Engineering Computing, Sichuan University of Science and Engineering, Zigong 643000, China

[3] College of Mathematics and Physics, Geomathematics Key Laboratory of Sichuan, Chengdu University of Technology, Chengdu 610059, China

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

Abstract: This paper presents a greedy average block Kaczmarz (GABK) method to solve the large scaled consistent system of linear equations. The GABK method introduces the strategy of extrapolation process to improve the GBK algorithm and to avoid computing the Moore-Penrose inverse of a submatrix of the coefficient matrix determined by the block index set. The GABK method is proved to converge linearly to the least-norm solution of the consistent system of linear equations. Numerical examples show that the GABK method has the best efficiency and effectiveness among all methods compared.

Keywords: block Kaczmarz; extrapolated stepsize; greedy strategy; GABK
Mathematics Subject Classification: 65F10, 65F45

1. Introduction

We are concerned with the numerical solution of the large scaled consistent system of linear equations of the form

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

where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$ are known and $x \in \mathbb{R}^n$ is unknown to be determined.

The Kaczmarz method in [1], which was revised to be applied to image reconstruction in [2] and is called as an algebraic reconstruction technique (ART), is a simple and of high performance for solving the large scaled system of linear Eq (1.1), and has many applications such as image reconstruction in computerized tomography [2–5] and parallel computing [4,6].

The block Kaczmarz methods (BK) have received much attention for its high efficiency for solving (1.1). Elfving [7] and Eggermont et al. [8] first presented block iterative methods to solve (1.1). Needell et al. in [9] proposed a randomized block Kaczmarz (RBK) algorithm to solve the linear least-squares problem by choosing a subsystem from the pre-determined partitions at random, which converges to the least-norm solution of (1.1) with an expected linear rate of convergence. Needell et al. in [10] further presented a randomized double block Kaczmarz (RDBK) to solve an inconsistent linear system (1.1). Chen and Huang [11] improved the error estimate in expectation and obtained a much better upper bound of the error estimate than that in [10]. Gower and Richtárik in [12] presented a Gaussian Kaczmarz (GK) method for (1.1). Necoara in [13] developed a unified framework of randomized average block Kaczmarz (RABK) algorithms with the iteration of the form

$$x_{k+1} = x_k + \alpha_k \sum_{i\in\mathcal{J}_k} \omega_i^k \frac{b^{(i)} - A^{(i)}x_k}{\|A^{(i)}\|_2^2}(A^{(i)})^T \tag{1.2}$$

for the consistent system (1.1), where $A^{(i)}$ denotes the ith row of the matrix A and $b^{(i)}$ denotes the ith entry of the vector b, ω_i^k presents the weight of the ith row of the matrix A at k step iteration. Niu and Zheng in [14] simplified the greed strategy in [15] to produce a greedy probability criterion

$$\mathcal{J}_k = \{i \| b^{(i)} - A^{(i)}x_k|^2 \geq \varepsilon_k \|A^{(i)}\|^2\} \tag{1.3}$$

where

$$\varepsilon_k = \eta \max_{1\leq i\leq m}\{\frac{|b^{(i)} - A^{(i)}x_k|^2}{\|A^{(i)}\|_2^2}\} \tag{1.4}$$

with $\eta \in (0,1)$, and proposed a greedy block Kaczmarz algorithm (GBK) with the iteration

$$x_{k+1} = x_k + A^{\dagger}_{\mathcal{J}_k}(b_{\mathcal{J}_k} - A_{\mathcal{J}_k}x_k). \tag{1.5}$$

It is proved that the GBK method converges linearly to the unique minimum norm least-squares solution of (1.1). We refer [15–18] more recent work on block Kaczmarz methods. The GBK algorithm in [14] needs to compute the Moore-Penrose inverse $A^{\dagger}_{\mathcal{J}_k}$ of $A_{\mathcal{J}_k}$ at each step and may be expensive when $\mathcal{J}_k$ is large enough.

In this paper, we improve the GBK method in [14] by introducing the strategy of extrapolation process proposed in [13] to avoid the computing of $A^{\dagger}_{\mathcal{J}_k}$ in (1.5). The proposed method is called a greedy average block Kaczmarz algorithm, which is abbreviated as GABK. Numerical examples in Section 3 shows the GABK method has the best efficiency by the running time and the number of iterations and the best effectiveness by the convergence of the relative solution error among all methods compared.

The rest of this paper is organized as follows. Section 2 presents a greedy average block Kaczmarz algorithm and proves its convergence. Several examples are shown in Section 3 and some conclusions are drawn in Section 4.

2. A greedy average block Kaczmarz algorithm

We introduce the strategy of extrapolation process proposed in [13] for the GBK algorithm in (1.5) to avoid the computing of $A^{\dagger}_{\mathcal{J}_k}$ in (1.5) and propose a greedy average block Kaczmarz (GABK) method. Algorithm 1 summarizes the GABK algorithm. Steps 4 and 5 determine the control index set. Step 6 computes the stepsize which is used to determine the stepsize adaptively and Step 7 presents the iteration process which avoids the computing of $A^{\dagger}_{\mathcal{J}_k}$ in (1.5), where the weight ω_i^k in (2.1) is chosen such that $0 < \omega_i^k < 1$ and $\sum_{i\in\mathcal{J}_k} \omega_i^k = 1$. We set $\omega_i^k = 1/\mathcal{J}_k$ in Section 3.

We consider the convergence of the GABK algorithm 1. We first need the following results presented in [17] to prove the convergence of Algorithm 1.

Algorithm 1: The greedy average block Kaczmarz algorithm (GABK).

```
Input: $A, b, x_0, \zeta \in (0,1], \delta \in (0,1]$.
Output: the approximation solution $x_{k+1}$ of the consistent system (1.1).
for $k = 0, 1, \dots$ until converge do
    Compute $\varepsilon_k = \zeta \max_{1\le i\le m}\{\gamma_k(i)\}$, where $\gamma_k(i) = \frac{|b^{(i)}-A^{(i)}x_k|_2^2}{\|A^{(i)}\|_2^2}$.
    Determine the control index set $\mathcal{J}_k = \{i \mid |\gamma_k(i) \ge \varepsilon_k\}$.
    Compute
        $$\alpha_k = (2-\delta)\frac{\sum_{i\in\mathcal{J}_k} \frac{\omega_i^k (b^{(i)}-A^{(i)}x_k)^2}{\|A^{(i)}\|_2^2}}{\|\sum_{i\in\mathcal{J}_k} \frac{\omega_i^k (b^{(i)}-A^{(i)}x_k)(A^{(i)})^T}{\|A^{(i)}\|_2^2}\|_2^2}. \quad (2.1)$$
    Update $x_{k+1} = x_k + \alpha_k \sum_{i\in\mathcal{J}_k} \omega_i^k \frac{b^{(i)}-A^{(i)}x_k}{\|A^{(i)}\|_2^2}(A^{(i)})^T$.
end
```

Lemma 2.1. *If $A \in \mathcal{R}^{m\times n}$ is a nonzero real matrix, then it holds that*

$$\sigma_{min}^2(A)\|x\|_2^2 \le \|Ax\|_2^2 \le \sigma_{max}^2(A)\|x\|_2^2$$

for any $x \in range(A^T)$, where $\sigma_{min}^2(A)$ and $\sigma_{max}^2(A)$ denote the minimum and maximum singular value of A, respectively.

Theorem 2.1. *Assume the system of linear Eq* (1.1) *is consistent and let x^* be a solution of* (1.1). *Let $\{x_k\}_{k\ge 0}$ be generated by Algorithm 1 with $x_0 \in range(A^T)$. Assume that the weights fulfill $\omega_i^k \in (0,1)$ for all $i \in \mathcal{J}_k$ and k. Denote by $\omega_{min} = \min_{i\in\mathcal{J}_k, k\ge 0}\{\omega_i^k\}$ and $\omega_{max} = \max_{i\in\mathcal{J}_k, k\ge 0}\{\omega_i^k\}$. Then it holds that*

$$\|x_{k+1} - x^*\|_2^2 \le (1 - \frac{\delta(2-\delta)\zeta|\mathcal{J}_k|\omega_{min}\sigma_{min}^2(A)}{\omega_{max}\lambda_{max}^{block}\|A\|_F^2})\|x_k - x^*\|_2^2, \quad (2.2)$$

where $\lambda_{max}^{block} = \max_{\mathcal{J}_k}\{\lambda_{max}(D)\}$ with $D = A_{\mathcal{J}_k}^T diag(1/\|A^{(i)}\|_2^2, i \in \mathcal{J}_k)A_{\mathcal{J}_k}$.

Proof. With the consistency assumption of the system of linear Eq (1.1), we have $Ax^* = b$, and

$$\langle x_k - x^*, (A^{(i)}x_k - b^{(i)})(A^{(i)})^T\rangle = (A^{(i)}x_k - b^{(i)})^2.$$

According to the update rule of GABK and using the notation of $\overline{\omega}_i^k = \frac{\omega_i^k}{\|A^{(i)}\|_2^2}$, where α_k is defined in (2.1), we have

$$\begin{aligned}
\|x_{k+1} - x^*\|_2^2 &= \|x_k - \alpha_k \sum_{i\in\mathcal{J}_k} \overline{\omega}_i^k (A^{(i)}x_k - b^{(i)})(A^{(i)})^T - x^*\|_2^2 \\
&= \|x_k - x^*\|_2^2 - 2\alpha_k \sum_{i\in\mathcal{J}_k} \overline{\omega}_i^k (A^{(i)}x_k - b^{(i)})^2 \\
&+ \alpha_k^2 \| \sum_{i\in\mathcal{J}_k} \overline{\omega}_i^k (A^{(i)}x_k - b^{(i)})(A^{(i)})^T\|_2^2 \\
&= \|x_k - x^*\|_2^2 - 2(2-\delta)\frac{(\sum\limits_{i\in\mathcal{J}_k} \overline{\omega}_i^k (A^{(i)}x_k - b^{(i)})^2)^2}{\|\sum\limits_{i\in\mathcal{J}_k} \overline{\omega}_i^k (A^{(i)}x_k - b^{(i)})(A^{(i)})^T\|_2^2} \\
&+ (2-\delta)^2 \frac{(\sum\limits_{i\in\mathcal{J}_k} \overline{\omega}_i^k (A^{(i)}x_k - b^{(i)})^2)^2}{\|\sum\limits_{i\in\mathcal{J}_k} \overline{\omega}_i^k (A^{(i)}x_k - b^{(i)})(A^{(i)})^T\|_2^2} \\
&= \|x_k - x^*\|_2^2 - \delta(2-\delta)L_k \sum_{i\in\mathcal{J}_k} \overline{\omega}_i^k (A^{(i)}x_k - b^{(i)})^2, \qquad (2.3)
\end{aligned}$$

in which

$$L_k = \frac{\sum\limits_{i\in\mathcal{J}_k} \overline{\omega}_i^k (A^{(i)}x_k - b^{(i)})^2}{\|\sum\limits_{i\in\mathcal{J}_k} \overline{\omega}_i^k (A^{(i)}x_k - b^{(i)})(A^{(i)})^T\|_2^2}.$$

Now we consider the bound of L_k. According to (2.1), we have

$$\begin{aligned}
L_k &= \frac{\|diag(\sqrt{\overline{\omega}_i^k}, i\in\mathcal{J}_k)(A_{\mathcal{J}_k}x_k - b_{\mathcal{J}_k})\|_2^2}{\|A_{\mathcal{J}_k}^T diag(\overline{\omega}_i^k, i\in\mathcal{J}_k)(A_{\mathcal{J}_k}x_k - b_{\mathcal{J}_k})\|_2^2} \\
&\geq \frac{1}{\lambda_{max}(diag(\sqrt{\overline{\omega}_i^k}, i\in\mathcal{J}_k)A_{\mathcal{J}_k}A_{\mathcal{J}_k}^T diag(\sqrt{\overline{\omega}_i^k}, i\in\mathcal{J}_k))} \\
&= \frac{1}{\lambda_{max}(A_{\mathcal{J}_k}^T diag(\overline{\omega}_i^k, i\in\mathcal{J}_k)A_{\mathcal{J}_k})} \\
&\geq \frac{1}{(\max\limits_{i\in\mathcal{J}_k} \omega_i^k)\lambda_{max}(A_{\mathcal{J}_k}^T diag(\frac{1}{\|A^{(i)}\|_2^2}, i\in\mathcal{J}_k)A_{\mathcal{J}_k})} \\
&\geq \frac{1}{\omega_{max}\lambda_{max}^{block}}. \qquad (2.4)
\end{aligned}$$

Substituting the bound (2.4) into (2.3) results in

$$\|x_{k+1} - x^*\|_2^2 \leq \|x_k - x^*\|_2^2 - \delta(2-\delta)\frac{1}{\omega_{max}\lambda_{max}^{block}} \sum_{i\in\mathcal{J}_k} \omega_i^k \frac{(A^{(i)}x_k - b^{(i)})^2}{\|A^{(i)}\|_2^2}$$

$$\leq \|x_k - x^*\|_2^2 - \delta(2-\delta)\frac{\omega_{min}\varepsilon_k|J_k|}{\omega_{max}\lambda_{max}^{block}}, \tag{2.5}$$

where the last inequality holds because of the choice of the index in Step 5 of Algorithm 1, i.e., $\gamma_k(i) = \frac{(A^{(i)}x_k - b^{(i)})^2}{\|A^{(i)}\|_2^2} \geq \varepsilon_k$ for $i \in \mathcal{J}_k$. For $k = 1, 2, \ldots$, since

$$\begin{aligned}\|b - Ax_k\|_2^2 &= \sum_{i\in[m]} |b^{(i)} - A^{(i)}x_k|^2 = \sum_{i\in[m]} \frac{|b^{(i)} - A^{(i)}x_k|^2}{\|A^{(i)}\|_2^2}\|A^{(i)}\|_2^2 \\ &\leq \max_{1\leq i\leq m} \gamma_k(i)\|A\|_F^2\},\end{aligned}$$

where $[m] = \{1, 2, \ldots, m\}$, then

$$\varepsilon_k = \zeta \max_{1\leq i\leq m} \gamma_k(i) \geq \zeta\frac{\|b - Ax_k\|_2^2}{\|A\|_F^2} \geq \zeta\frac{\sigma_{min}^2(A)\|x_k - x^*\|_2^2}{\|A\|_F^2}, \tag{2.6}$$

where the last inequality holds by the lemma 2.1. Thus, substituting (2.6) into (2.5) results in (2.2). This completes the proof. □

We remark that under the conditions of Theorem 2.1, if the matrix A is a normalized matrix, i.e., $\|A^{(i)}\|_2^2 = 1$ for all $i \in [m]$, then the convergence rate of GABK, which is determined by $\rho_{GABK} = \|x_{k+1} - x^*\|_2 / \|x_k - x^*\|_2$, is subject to $0 < \rho_{GABK} < 1$. In fact,

$$\begin{aligned}1 - \frac{\delta(2-\delta)|\mathcal{J}_k|\omega_{min}\sigma_{min}^2(A)}{\omega_{max}\lambda_{max}^{block}\|A\|_F^2} &= 1 - \frac{\delta(2-\delta)\omega_{min}}{\omega_{max}}\frac{\|A_{\mathcal{J}_k}\|_F^2}{\sigma_{min}^2(A_{\mathcal{J}_k})}\frac{\sigma_{min}^2(A)}{\|A\|_F^2} \\ &\leq 1 - \frac{\delta(2-\delta)\omega_{min}}{\omega_{max}}\frac{\sigma_{min}^2(A)}{\|A\|_F^2},\end{aligned}$$

the inequality holds because of the fact $\frac{\|A_{\mathcal{J}_k}\|_F^2}{\sigma_{min}^2(A_{\mathcal{J}_k})} \geq 1$. Moreover, $0 < \delta(2-\delta) \leq 1$ since $\delta \in (0, 1]$, and $0 < \frac{\sigma_{min}^2(A)}{\|A\|_F^2} \leq 1$. Then it holds that

$$0 < \frac{\delta(2-\delta)\omega_{min}}{\omega_{max}}\frac{\sigma_{min}^2(A)}{\|A\|_F^2} < 1,$$

Thus $0 < \rho_{GABK} < 1$, which means that Algorithm 1 has a linear convergence.

We give a special selection of the parameters in Theorem 2.1 as follows, which will be used in Section 3.

Corollary 2.1. *If* $\delta = 1$, $\omega_i^k = \frac{1}{|\mathcal{J}_k|}$ *and* $\|A^{(i)}\|_2^2 = 1$ *for all k and i, then it holds that*

$$\|x_{k+1} - x^*\|_2^2 \leq (1 - \frac{\zeta|\mathcal{J}_k|\lambda_{min}^{nz}(A^TA)}{m\lambda_{max}(A_{\mathcal{J}_k}^TA_{\mathcal{J}_k})})\|x_k - x^*\|_2^2.$$

Proof. Since $\|A^{(i)}\|_2^2 = 1$, then $\|A\|_F^2 = m$ and $\lambda_{max}^{block} = \lambda_{max}(A_{\mathcal{J}_k}^TA_{\mathcal{J}_k})$, substituting these results and the assumption that $\delta = 1$ and $\omega_i^k = \frac{1}{|\mathcal{J}_k|}$ into the right-hand side of (2.2) results in

$$1 - \frac{\delta(2-\delta)|\mathcal{J}_k|\omega_{min}\sigma_{min}^2(A)}{\omega_{max}\lambda_{max}^{block}\|A\|_F^2} = 1 - \frac{\zeta|\mathcal{J}_k|\lambda_{min}^{nz}(A^TA)}{m\lambda_{max}(A_{\mathcal{J}_k}^TA_{\mathcal{J}_k})},$$

which implies (2.2). This completes the proof. □

We complete this section by analyzing the arithmetic complexity of Algorithm 1. It needs about $(nnz(A)+nnz(A_{\mathcal{J}_k})+m+2n+2)$ complex flops at the kth iteration of the GABK method, where $nnz(A)$ and $nnz(A_{\mathcal{J}_k})$ denote the number of nonzero elements of A and the submatrix $A_{\mathcal{J}_k}$ respectively. Moreover, the approximate computing cost of the k-step iteration of the GBK method is $(2nnz(A_{\mathcal{J}_k})+3n+2|\mathcal{J}_k|)K+n+2n|\mathcal{J}_k|$ complex flops, where $|\mathcal{J}_k|$ is the cardinality of the set $\mathcal{J}_k$, and in GBK method the computation of the pseudoinverse is approximated by performing K number of iterations using conjugate gradients CGLS.

3. Numerical experiments

In this section, several different kinds of examples are given to show the efficiency and effectiveness of Algorithm 1 (GABK) for solving the consistent system of linear Eq (1.1). The GABK method is compared with the RABK method in [13], the GBK method in [14] and the fast deterministic block Kaczmarz (FDBK) in [16]. We run all examples by the soft of Matlab with the R2019b version on a personal computer with 2.0 GHz Inter(R) Core(TM) i7-8565U CPU processing unit, 8 GB memory, and 64 bit Windows 10 operating system.

For the RABK method, we consider two cases used in [13], and use the same sampling methods and choices of blocks, stepsizes and weights, which is listed in Table 1.

Table 1. The sampling methods and parameters of two cases of the RABK method in [13].

Method	Sampling method	Block size τ	Stepsize α_k	Weights ω_i^k
RABK_a	Uniform sampling	10	L_k	$\frac{1}{\lvert\mathcal{J}_k\rvert}$
RABK_a_paved	Partition sampling	$\lfloor \frac{m}{\|A\|_2^2} \rfloor$	L_k	$\frac{1}{\lvert\mathcal{J}_k\rvert}$

The partition sampling in Table 1 means that selects randomly from the row partition $P_s = \sigma_1, \sigma_2, \ldots, \sigma_s$ at $(k+1)$th iteration, where $s = \lceil \|A\|_2^2 \rceil$, $\sigma_i = \{\lfloor (i-1)\frac{m}{s} \rfloor + 1, \lfloor (i-1)\frac{m}{s} \rfloor + 2, \ldots, \lfloor i\frac{m}{s} \rfloor\}$, $i = 1, 2, \ldots, s$, and $|\mathcal{J}_k|$ is the size of the block control set $\mathcal{J}_k$ at the kth iteration.

For the GBK method, we use the parameter $\eta = \frac{1}{2}(\max\limits_{i\in[m]}\{\varepsilon_k(i)\} + \frac{\|b-Ax_k\|_2^2}{\|A\|_F^2})$ in (1.4), and the block control set in [16] for the FDBK method. The CGLS algorithm is used to calculate the Moore-Penrose inverse $A_{\mathcal{J}_k}^{\dagger}$ at each iteration of both algorithms. For the GABK method, we set $\zeta = 0.2$ in (1.3) to grasp more rows from the matrix A, $\delta = 1$ and the weights $\omega_i^k = 1/|\mathcal{J}_k|$ in (2.1).

Three types of coefficient matrices A are considered to construct the consistent systems (1.1), i.e., overdetermined or underdetermined dense matrices with normally distribution produced by the Matlab function **randn(m,n)**, large full rank sparse matrices and rank-deficient sparse matrices from the suitesparse matrix collection in [19]. We let $b \in \mathbb{R}^m$ in (1.1) be generated by Ax^*, where $x^* \in \mathbb{R}^n$ represents the exact solution produced by the Matlab function **randn**. The performance of the GABK and other methods are evaluated in terms of *efficiency* and *effectiveness*. The efficiency is defined by the iteration number denoted by 'IT' and the CPU time in seconds by 'CPU'. The effectiveness is determined by *the relative solution error (RSE)* defined by

$$RSE = \frac{\|x_k - x_*\|_2^2}{\|x_*\|_2^2}.$$

The initial solution x_0 is set as 0 in all experiments, and all algorithms do not stop until the RSE satisfies $RSE < 10^{-6}$. All numerical results reported as follows are arithmetical average quantities with respect to 50 repeated trials of each method. The *speed-up* of GABK against other methods is defined by

$$speed-up\_method = \frac{CPU\,of\,a\,method}{CPU\,of\,GABK}.$$

3.1. Example 1. consistent overdetermined systems

Table 2. *IT*, *CPU* and the *speed-up* of Algorithm 1 (GABK) compared with RABK [13], GBK [14] and FDBK [16] for solving the consistent system of linear Eq (1.1) with a dense overdetermined matrix A.

m			$1*10^3$	$2*10^3$	$3*10^3$	$4*10^3$	$5*10^3$
n=100	RABK_a	IT	155.4	147.4	144.7	145.1	142.2
		CPU	0.0044	0.0059	0.0067	0.0099	0.0104
	RABK_a_paved	IT	32.8	27.4	25.6	25.4	24.5
		CPU	0.0014	0.0018	0.0020	0.0023	0.0029
	GBK	IT	24	14	14	11	11
		CPU	0.0013	0.0014	0.0018	0.0023	0.0030
	FDBK	IT	22	13	12	11	10
		CPU	0.0012	0.0013	0.0016	0.0024	0.0028
	GABK	IT	9	7	6	6	5
		CPU	5.98e-04	7.14e-04	7.91e-04	0.0013	0.0015
	speed-up_RABK_a		7.36	8.26	8.47	7.62	6.93
	speed-up_RABK_a_paved		2.34	2.52	2.53	1.77	1.93
	speed-up_GBK		2.20	1.97	2.26	1.86	2.00
	speed-up_FDBK		2.07	1.87	2.01	1.94	1.87
n=500	RABK_a	IT	2644.8	1085.6	860.3	809.2	779.5
		CPU	0.1335	0.0932	0.0875	0.0861	0.0919
	RABK_a_paved	IT	180.4	57.1	42.6	37.4	34.2
		CPU	0.0512	0.0404	0.0412	0.0485	0.0501
	GBK	IT	270	80	51	40	35
		CPU	0.0632	0.0519	0.0636	0.0732	0.0817
	FDBK	IT	273	76	49	40	33
		CPU	0.0526	0.0497	0.0634	0.0813	0.0874
	GABK	IT	72	24	16	12	11
		CPU	0.0201	0.0231	0.0228	0.0228	0.0266
	speed-up_RABK_a		6.64	4.04	3.84	3.78	3.46
	speed-up_RABK_a_paved		2.55	1.75	1.81	2.13	1.88
	speed-up_GBK		3.14	2.25	2.78	3.21	3.07
	speed-up_FDBK		2.61	2.15	2.78	3.57	3.29

In this example, we consider the solution of the consistent overdetermined system of linear Eq (1.1) with a dense overdetermined ($m \geq n$) matrix $A \in \mathbb{R}^{m\times n}$, which has normal distribution. The coefficient matrix $A \in \mathbb{R}^{m\times n}$ with different size combined with $m = i * 10^3$ $(i = 1, 2, ..., 5)$ and $n = j * 10^2$ $(j = 1, 5)$ is produced by the Matlab function **randn(m,n)**. Table 2 shows *IT* and *CPU* together with the *speed-up* for Algorithm 1 for solving (1.1), and those compared with the RABK method in [13], the GBK method in [14] and the FDBK method in [16]. Figure 1 plots RSE versus IT (left) and CPU (right) of different method for solving (1.1) with the matrix $A = randn(4000, 500)$.

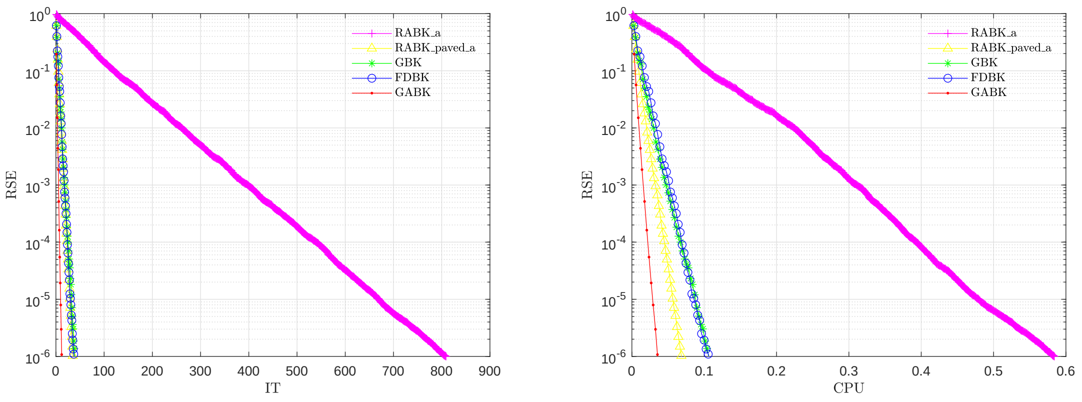

Figure 1. RSE versus IT (left) and CPU (right) of different method for solving (1.1) with the matrix $A = randn(4000, 500)$.

We can see from Table 2 that the GABK method needs the least number of iterations and least CPU time among all methods. This result benefits from the greater block control set and an extrapolated stepsize in each iteration in Algorithm 1. Figure 1 shows that GABK has the fastest convergence of RSE both on *IT* and *CPU* among all methods.

3.2. *Example 2. consistent underdetermined systems*

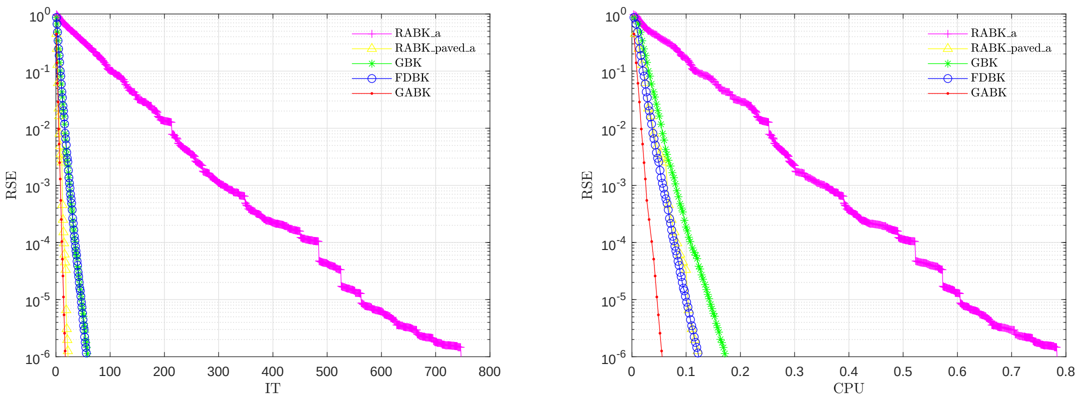

Figure 2. RSE versus IT (left) and CPU (right) of different method for solving (1.1) with the matrix $A = randn(500, 4000)$.

We use Algorithm 1 for solving the consistent underdetermined system of linear Eq (1.1) with different underdetermined ($m < n$) dense matrices $A \in \mathbb{R}^{m\times n}$, and compare it with RABK [13], GBK [14] and FDBK [16]. Table 3 lists different size of $A \in \mathbb{R}^{m\times n}$ combined with $m = i * 10^2$ ($i = 1, 5$) and $n = j * 10^3$ ($j = 1, 2, ..., 5$) and *IT*, *CPU* and *speed-up* of different methods. Figure 2 plots RSE versus IT (left) and CPU (right) of different method for solving (1.1) with the matrix $A = randn(500, 4000)$.

Similar results on *IT*, *CPU* and the *speed-up* to Table 2 in Example 1 are derived for Algorithm 1 from Table 3. It is observed from Figure 2 that the GABK method has the fastest convergence among all methods.

Table 3. *IT*, *CPU* and the *speed-up* of Algorithm 1 compared with RABK [13], GBK [14] and FDBK [16] for solving the consistent system of linear Eq (1.1) with a dense underdetermined matrix A.

n			$1 * 10^3$	$2 * 10^3$	$3 * 10^3$	$4 * 10^3$	$5 * 10^3$
m=100	RABK_a	IT	129.7	109	106.5	97.3	95.5
		CPU	0.0106	0.0145	0.0207	0.0253	0.0310
	RABK_a_paved	IT	17.4	11.9	10.6	9.2	8.6
		CPU	0.0029	0.0081	0.0115	0.0142	0.0156
	GBK	IT	37	30	27	25	26
		CPU	0.0049	0.0062	0.0081	0.0093	0.0120
	FDBK	IT	38	31	28	25	25
		CPU	0.0031	0.0038	0.0049	0.0054	0.0067
	GABK	IT	14	11	10	10	11
		CPU	0.0016	0.0019	0.0026	0.0032	0.0055
	speed-up_RABK_a		6.63	7.63	7.96	7.91	5.64
	speed-up_RABK_a_paved		1.81	4.26	4.42	4.44	2.84
	speed-up_GBK		3.04	3.25	3.12	2.91	2.18
	speed-up_FDBK		1.93	1.98	1.89	1.69	1.22
m=500	RABK_a	IT	2145	976.8	819.5	748.7	703.5
		CPU	0.2560	0.3704	0.6150	0.8577	1.0738
	RABK_a_paved	IT	109.8	39.6	26.9	21.6	19.1
		CPU	0.1018	0.0998	0.1152	0.1279	0.1468
	GBK	IT	307	98	69	51	50
		CPU	0.0984	0.0994	0.1574	0.1525	0.2041
	FDBK	IT	305	95	68	55	49
		CPU	0.0681	0.0716	0.1165	0.1249	0.1596
	GABK	IT	76	27	20	17	16
		CPU	0.0305	0.0399	0.0468	0.0523	0.0659
	speed-up_RABK_a		8.39	9.28	13.14	16.40	16.29
	speed-up_RABK_a_paved		3.34	2.94	2.46	2.45	2.23
	speed-up_GBK		3.23	2.49	3.37	2.91	3.10
	speed-up_FDBK		2.23	1.80	2.49	2.39	2.42

3.3. Example 3. linear systems with sparse full rank matrices

This example shows the application of Algorithm 1 to solve the consistent linear systems (1.1) with full rank sparse overdetermined or underdetermined matrices $A \in \mathbb{R}^{m\times n}$ from the Suite Sparse Matrix Collection in [19].

Tables 4 and 5 display the size, the sparsity (*density*) and the Euclidean condition number (*cond(A)*) of $A \in \mathbb{R}^{m\times n}$, where *density* of A is defined by the ratio of the number of nozeros of A to the total number of A.

Table 4. *IT*, *CPU* and the *speed-up* of Algorithm 1 (GABK) compared with RABK [13], GBK [14] and FDBK [16] for solving the consistent system of linear Eq (1.1) with a sparse overdetermined matrix A with full rank.

different A		ash219	ash608	ash958	Trefethen_700	ch7-8-b1
$m \times n$		219×85	608×188	958×292	700×700	1176×56
Density		2.35%	1.06%	0.86%	2.58%	3.57%
cond(A)		3.02	3.37	3.20	4710.39	4.79e+14
RABK_a	IT	185.2	374.2	563.7	3444	77.2
	CPU	0.0018	0.0046	0.0092	0.1585	0.0012
RABK_a_paved	IT	55.9	52.9	63.1	47	20.4
	CPU	7.26e-04	0.0012	0.0019	0.0079	3.73e-04
GBK	IT	41	50	59	107	15
	CPU	5.18e-04	9.15e-04	0.0014	0.0089	3.90e-04
FDBK	IT	48	48	57	104	17
	CPU	4.26e-04	5.76e-04	8.61e-04	0.0065	3.44e-04
GABK	IT	23	23	25	50	5
	CPU	2.76e-04	3.91e-04	5.68e-04	0.0028	1.57e-04
speed-up_RABK_a		6.53	11.76	16.19	56.61	7.63
speed-up_RABK_a_paved		2.63	3.07	3.34	2.82	2.37
speed-up_GBK		1.88	2.34	2.51	3.21	2.48
speed-up_FDBK		1.54	1.47	1.52	2.33	2.19
Name		ch7-9-b2	Franz10	cari^T	crew1^T	df2177^T
$m \times n$		17640×1512	19588×4164	1200×400	6469×135	10358×630
Density		3.45e-05	6.23e-06	31.83%	0.11%	0.34%
cond(A)		1.61e+15	1.27e+16	3.13	18.20	2.01
RABK_a	IT	2029.9	6726.3	609.3	970.5	891.6
	CPU	0.6775	4.1036	0.0917	0.0930	0.1398
RABK_a_paved	IT	24.4	219.3	692.1	861.5	36.9
	CPU	0.0132	0.1496	0.0940	0.0894	0.0079
GBK	IT	31	175	36	346	31
	CPU	0.0230	0.1981	0.0211	0.1433	0.0125
FDBK	IT	40	198	44	265	42
	CPU	0.0217	0.1650	0.0244	0.0943	0.0128
GABK	IT	14	65	22	167	14
	CPU	0.0101	0.0664	0.0103	0.0434	0.0047
speed-up_RABK_a		67.08	61.80	8.90	2.14	29.75
speed-up_RABK_a_paved		1.31	2.25	9.13	2.06	1.68
speed-up_GBK		2.27	2.98	2.04	3.30	2.65
speed-up_FDBK		2.14	2.49	2.36	2.17	2.71

The *IT* and *CPU* together with the *speed-up* of Algorithm 1 for solving (1.1) with different sparse overdetermined full rank matrix $A \in \mathbb{R}^{m\times n}$ and with different sparse underdetermined full rank matrix $A \in \mathbb{R}^{m\times n}$ are listed in Tables 4 and 5, respectively.

Tables 4 and 5 show very similar results to Table 2 in Example 1 for the GABK method. It can be seen from Figures 3 and 4 that the GABK method converges fastest among all methods for both overdetermined and underdetermined cases.

GABK is compared with the RABK [13], GBK [14] and FDBK [16] methods. Figures 3 and 4 show RSE versus IT (left) and CPU (right) of different method for solving (1.1) with *df2177* and *df2177^T*, respectively.

Table 5. *IT*, *CPU* and the *speed-up* of Algorithm 1 (GABK) compared with RABK [13], GBK [14] and FDBK [16] for solving the consistent system of linear Eq (1.1) with a sparse underdetermined matrix A with full rank.

different A		df2177	cari	ch7-6-b1^T	model1	nemsafm
$m \times n$		630×10358	400×1200	42×630	362×798	334×2348
Density		0.34%	31.83%	0.98%	1.05%	0.36%
cond(A)		2.01	3.13	1.16e+15	17.57	4.77
RABK_a	IT	783.4	468.3	33.5	1594.7	541.8
	CPU	0.2359	0.0836	7.35e-04	0.0390	0.0205
RABK_a_paved	IT	29.5	105.5	6.4	112.7	35.6
	CPU	0.0199	0.0583	3.17e-04	0.0074	0.0023
GBK	IT	42	86	23	297	65
	CPU	0.0256	0.0448	5.35e-04	0.0106	0.0034
FDBK	IT	42	74	23	294	64
	CPU	0.0130	0.0411	3.09e-04	0.0058	0.0018
GABK	IT	14	33	9	85	22
	CPU	0.0058	0.0155	1.62e-04	0.0026	8.12e-04
speed-up_RABK_a		40.67	5.39	4.54	15.00	25.26
speed-up_RABK_a_paved		3.43	3.76	1.96	2.85	2.83
speed-up_GBK		4.44	2.89	3.31	4.02	4.19
speed-up_FDBK		2.25	2.64	1.91	2.120	2.20
Name		crew1	GL7d25	abtaha1^T	ash958^T	Franz10^T
$m \times n$		135×6469	2798×21074	209×14596	292×958	4164×19588
Density		5.38%	1.14e-05	1.01e-04	0.68%	6.23e-06
cond(A)		18.20	1.42e+19	12.23	3.20	1.27e+16
RABK_a	IT	1171.8	3575.2	1940.5	396.7	4949.1
	CPU	0.3156	3.6454	1.1722	0.0089	4.2183
RABK_a_paved	IT	667.1	23.5	195.6	23.1	39.9
	CPU	0.2039	0.0509	0.2216	0.0011	0.0676
GBK	IT	896	59	355	54	70
	CPU	0.4992	0.1169	0.3853	0.0021	0.1014
FDBK	IT	759	61	353	53	69
	CPU	0.2602	0.0678	0.1862	9.38e-04	0.0487
GABK	IT	272	32	105	16	24
	CPU	0.1021	0.0386	0.0799	4.62e-04	0.0233
speed-up_RABK_a		3.09	94.44	14.67	19.26	184.04
speed-up_RABK_a_paved		2.00	1.32	2.77	2.38	2.90
speed-up_GBK		4.89	3.03	4.82	4.56	4.35
speed-up_FDBK		2.55	1.76	2.33	2.03	2.09

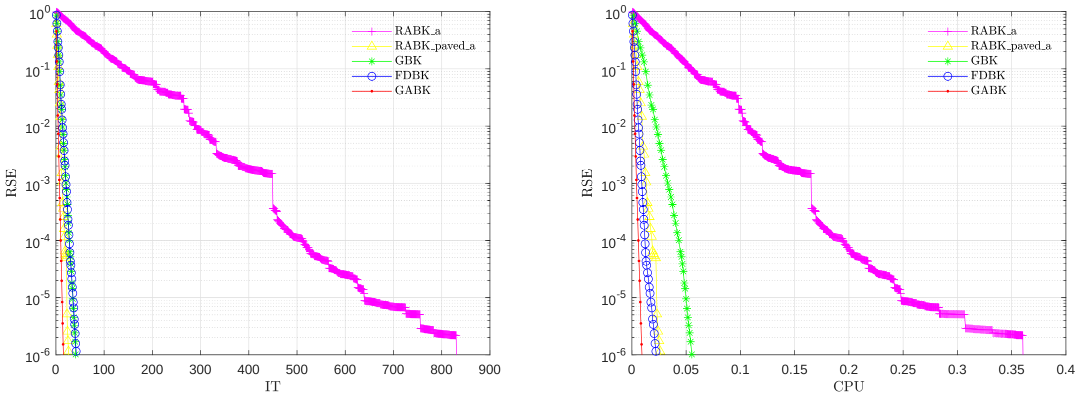

Figure 3. RSE versus IT (left) and CPU (right) of different method for solving (1.1) with the matrix *df2177*.

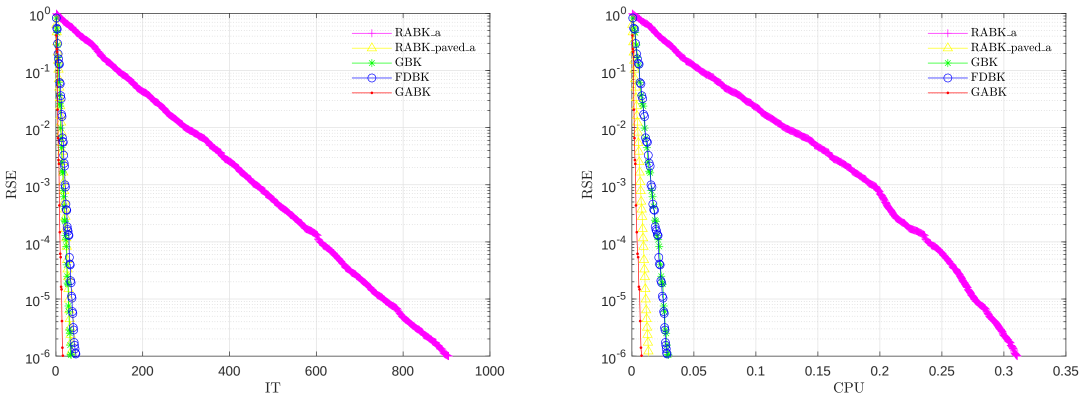

Figure 4. RSE versus IT (left) and CPU (right) of different method for solving (1.1) with the matrix *df2177*^{T}.

3.4. Example 4. linear systems with sparse rank-deficient matrices

We test the consistent system of linear Eq (1.1) with the sparse rank-deficient matrices from [19], which originate from different field of application such as directed graph and combinatorial problem. We remove zero rows of the matrices relat6, GD00_a and GL7d26 before running all methods. Table 6 shows *IT*, *CPU* and *speed-up* of Algorithm 1 for solving (1.1) compared with those of the RABK [13], GBK [14] and FDBK [16] methods. Figure 5 plots RSE versus IT (left) and CPU (right) of different methods for solving (1.1) with a sparse rank-deficient matrix *relat6*.

Table 6. *IT*, *CPU* and the *speed-up* of Algorithm 1 compared with RABK [13], GBK [14] and FDBK [16] for solving the consistent system of linear Eq (1.1) with a sparse matrix A with deficient rank.

Name		relat6	GD00_a	Sandi_authors	us04	GL7D26	flower_5_1
$m \times n$		2340×157	352×352	86×86	163×28016	305×2798	211×201
Density		0.34%	31.83%	1.05%	1.18e-04	4.83e-04	1.42%
cond(A)		2.01	3.13	1.79e+18	inf	6.02e+18	inf
RABK_a	IT	860.1	1423.6	3393.4	1980.4	365.1	2225.5
	CPU	0.0245	0.0162	0.0298	4.7449	0.0286	0.0248
RABK_a_paved	IT	248.6	1437.8	2745.5	1480.2	20.1	238
	CPU	0.0085	0.0170	0.0237	3.7823	0.0036	0.0045
GBK	IT	184	387	1702	626	37	545
	CPU	0.0103	0.0070	0.0201	1.9183	0.0051	0.0088
FDBK	IT	173	310	1582	556	37	488
	CPU	0.0079	0.0035	0.0124	1.1454	0.0026	0.0052
GABK	IT	91	133	906	278	17	132
	CPU	0.0049	0.0018	0.0074	0.7613	0.0018	0.0021
speed-up_RABK_a		5.00	9.00	4.03	6.23	15.89	11.81
speed-up_RABK_a_paved		1.74	9.44	3.20	4.97	2.00	2.14
speed-up_GBK		2.09	3.88	2.73	2.52	2.88	4.14
speed-up_FDBK		1.60	1.95	1.69	1.51	1.44	2.44

Table 6 illustrates that Algorithm 1 needs the least CPU time and the least number of iterations among all methods. Figure 5 shows that the GABK method converges fastest among all methods for solving (1.1) with the sparse rank-deficient matrix *relat6*.

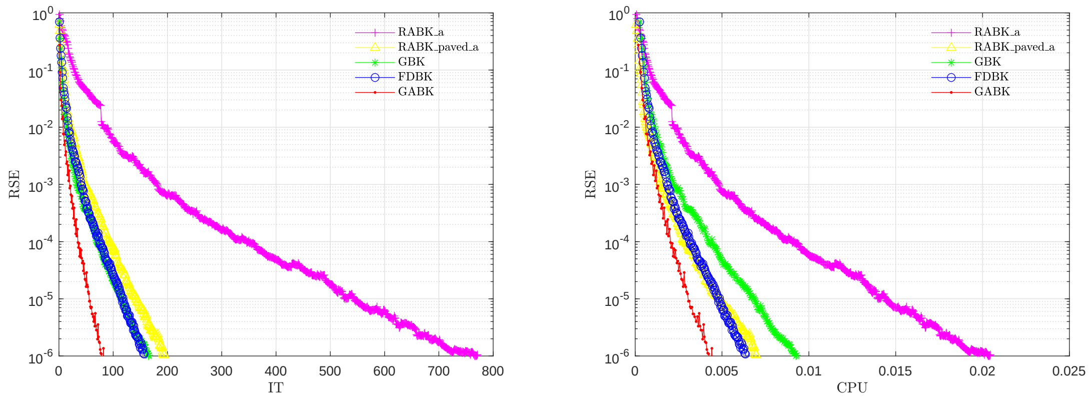

Figure 5. RSE versus IT (left) and CPU (right) of different method for solving (1.1) with the sparse rank-deficient matrix *relat6*.

4. Conclusions

We propose a greedy average block Kaczmarz (GABK) approach to solve the large scaled consistent system of linear equations. We consider both dense and sparse systems. The tested systems are over-determined and under-determined with full rank or rank-deficient matrix. The GABK method is proved to converge linearly to the least-norm solution of the underlying linear system. Numerical examples show that the GABK method has the best efficiency and effectiveness among all the methods compared.

Acknowledgments

The authors are thankful to the referees for helpful comments. Research by F. Yin was partially supported by NNSF (grant 11501392), research by G.X. Huang was supported in part by Application Fundamentals Foundation of STD of Sichuan (grant 2020YJ0366) and Key Laboratory of bridge nondestructive testing and engineering calculation Open fund projects (grant 2020QZJ03), and research by Y.M. Liao was partially supported by the Innovation Fund of Postgraduate of SUSE (grant y2021101).

Conflict of interest

The authors declare no conflicts of interest in this paper.

References

1. S. Kaczmarz, Angenäherte auflösung von systemen linearer gleichungen, *International Bulletin of the Polish Academy of Sciences, Letters A*, **35** (1937), 355–357.
2. R. Gordon, R. Bender, G. Herman, Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography, *J. Theor. Biol.*, **29** (1970), 471–481. http://dx.doi.org/10.1016/0022-5193(70)90109-8
3. C. Byrne, A unified treatment of some iterative algorithms in signal processing and image reconstruction, *Inverse Probl.*, **20** (2004), 103. http://dx.doi.org/10.1088/0266-5611/20/1/006
4. Y. Censor, Parallel application of block-iterative methods in medical imaging and radiation therapy, *Math. Program.*, **42** (1988), 307–325. http://dx.doi.org/10.1007/BF01589408
5. G. Herman, *Image reconstruction from projections: the fundamentals of computerized tomography*, New York: Academic Press, 1980.
6. J. Elble, N. Sahinidis, P. Vouzis, Gpu computing with Kaczmarz's and otheriterative algorithms for linear systems, *Parallel Comput.*, **36** (2010), 215–231. http://dx.doi.org/10.1016/j.parco.2009.12.003
7. T. Elfving, Block-iterative methods for consistent and inconsistent linear equations, *Numer. Math.*, **35** (1980), 1–12. http://dx.doi.org/10.1007/BF01396365
8. P. Eggermont, G. Herman, A. Lent, Iterative algorithms for large partitioned linear systems, with applications to image reconstruction, *Linear Algebra Appl.*, **40** (1981), 37–67. http://dx.doi.org/10.1016/0024-3795(81)90139-7
9. D. Needell, J. Tropp, Paved with good intentions: analysis of a randomized block Kaczmarz method, *Linear Algebra Appl.*, **441** (2014), 199–221. http://dx.doi.org/10.1016/j.laa.2012.12.022
10. D. Needell, R. Zhao, A. Zouzias, Randomized block Kaczmarz method with projection for solving least squares, *Linear Algebra Appl.*, **484** (2015), 322–343. http://dx.doi.org/10.1016/j.laa.2015.06.027
11. J. Chen, Z. Huang, On the error estimate of the randomized double block Kaczmarz method, *Appl. Math. Comput.*, **370** (2019), 124907. http://dx.doi.org/10.1016/j.amc.2019.124907

12. R. Gower, P. Richtárik, Rndomized iterative methods for linear systems, *SIAM J. Matrix Anal. Appl.*, **36** (2015), 1660–1690. http://dx.doi.org/10.1137/15M1025487

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

14. Y. Niu, B. Zheng, A greedy block Kaczmarz algorithm for solving large-scale linear systems, *Appl. Math. Lett.,* **104** (2020), 106294. http://dx.doi.org/10.1016/j.aml.2020.106294

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

16. J. Chen, Z. Huang, On a fast deterministic block Kaczmarz method for solving large-scale linear systems, *Numer. Algor.*, in press. http://dx.doi.org/10.1007/s11075-021-01143-4

17. K. Du, W. Si, X. Sun, Randomized extended average block Kaczmarz for solving least squares, *SIAM J. Sci. Comput.,* **42** (2020), A3541–A3559. http://dx.doi.org/10.1137/20M1312629

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

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