A NOVEL GREEDY KACZMARZ METHOD FOR SOLVING CONSISTENT LINEAR SYSTEMS

HANYU LI† AND YANJUN ZHANG‡

Abstract. With a quite different way to determine the working rows, we propose a novel greedy Kaczmarz method for solving consistent linear systems. Convergence analysis of the new method is provided. Numerical experiments show that, for the same accuracy, our method outperforms the greedy randomized Kaczmarz method and the relaxed greedy randomized Kaczmarz method introduced recently by Bai and Wu [Z.Z. BAI AND W.T. WU, On greedy randomized Kaczmarz method for solving large sparse linear systems, SIAM J. Sci. Comput., 40 (2018), pp. A592–A606; Z.Z. BAI AND W.T. WU, On relaxed greedy randomized Kaczmarz methods for solving large sparse linear systems, Appl. Math. Lett., 83 (2018), pp. 21–26] in term of the computing time.

Key words. greedy Kaczmarz method, greedy randomized Kaczmarz method, greedy strategy, iterative method, consistent linear systems

AMS subject classifications. 65F10, 65F20, 65K05, 90C25, 15A06

1. Introduction. We consider the following consistent linear systems

\[ Ax = b, \]

where \( A \in \mathbb{C}^{m \times n} \), \( b \in \mathbb{C}^m \), and \( x \) is the \( n \)-dimensional unknown vector. As we know, the Kaczmarz method [10] is a popular so-called row-action method for solving the systems (1.1). In 2009, Strohmer and Vershynin [18] proved the linear convergence of the randomized Kaczmarz (RK) method. Following that, Needell [13] found that the RK method is not converge to the ordinary least squares solution when the system is inconsistent. To overcome it, Zouzias and Freris [20] extended the RK method to the randomized extended Kaczmarz (REK) method. Later, Ma, Needell, and Ramdas [12] provided a unified theory of these related iterative methods in all possible system settings. Recently, many works on Kaczmarz methods were reported; see for example [1, 2, 7, 6, 19, 4, 15] and references therein.

In 2018, Bai and Wu [1] first constructed a greedy randomized Kaczmarz (GRK) method by introducing an efficient probability criterion for selecting the working rows from the coefficient matrix \( A \), which avoids a weakness of the one adopted in the RK method. As a result, the GRK method is faster than the RK method in terms of the number of iterations and computing time. Subsequently, based on the GRK method, a so-called relaxed greedy randomized Kaczmarz (RGRK) method was proposed in [2] by introducing a relaxation parameter \( \theta \), which makes the convergence factor of the RGRK method be smaller than that of the GRK method when it is in \([\frac{1}{2}, 1]\), and the convergence factor reaches the minimum when \( \theta = 1 \). For the latter case, i.e., \( \theta = 1 \), Du and Gao [7] called it the maximal weighted residual Kaczmarz method and carried out extensive experiments to test this method. By the way, the idea of greed applied in [1, 2] also has wide applications, see for example [8, 14, 11, 3] and references therein.

In the present paper, we propose a novel greedy Kaczmarz (GK) method. Unlike the GRK and RGRK methods, the new method adopts a quite different way to determine the working rows of the
matrix $A$ and hence needs less computing time in each iteration; see the detailed analysis before Algorithm 3.1 below. Consequently, the GK method can outperform the GRK and RGRK methods in term of the computing time. This result is confirmed by extensive numerical experiments, which show that, for the same accuracy, the GK method requires almost the same number of iterations as those of the GRK and RGRK methods, but spends less computing time. In addition, we also prove the convergence of the GK method in theory.

The rest of this paper is organized as follows. In Section 2, notation and some preliminaries are provided. We present our novel GK method and its convergence properties in Section 3. Experimental results are given in Section 4.

2. Notation and preliminaries. For a vector $z \in \mathbb{C}^n$, $z^{(j)}$ represents its $j$th entry. For a matrix $G = (g_{ij}) \in \mathbb{C}^{m \times n}$, $G^{(i)}$, $\|G\|_2$, and $\|G\|_F$ denote its $i$th row, spectral norm, and Frobenius norm, respectively. In addition, we denote the smallest positive eigenvalue of $G^*G$ by $\lambda_{\min}(G^*G)$, where $(\cdot)^*$ denotes the conjugate transpose of a vector or a matrix, and the number of elements of a set $W$ by $|W|$.

In what follows, we use $x_\star = A^\dagger b$, with $A^\dagger$ being the Moore-Penrose pseudoinverse, to denote the least-Euclidean-norm solution to the systems (1.1). For finding this solution, Bai and Wu [1] proposed the GRK method listed as follows, where $r_k = b - Ax_k$ denotes the residual vector.

**Algorithm 2.1** The GRK method

**Input:** $A \in \mathbb{C}^{m \times n}$, $b \in \mathbb{C}^m$, $\ell$, initial estimate $x_0$

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

Compute

$$
\epsilon_k = \frac{1}{2} \left( \frac{1}{\|r_k\|_2^2} \max_{1 \leq i_k \leq m} \left\{ \frac{|r_k^{(i_k)}|^2}{\|A^{(i_k)}\|_2^2} \right\} + \frac{1}{\|A\|_F^2} \right).
$$

Determine the index set of positive integers

$$
U_k = \left\{ i_k \left| \frac{|r_k^{(i_k)}|^2}{\|r_k\|_2^2} \geq \epsilon_k \frac{\|r_k\|_2^2 \|A^{(i_k)}\|_2^2}{\|A\|_F^2} \right. \right\}.
$$

Compute the $i$th entry $\tilde{r}_k^{(i)}$ of the vector $\tilde{r}_k$ according to

$$
\tilde{r}_k^{(i)} = \begin{cases} 
  r_k^{(i)} & \text{if } i \in U_k, \\
  0 & \text{otherwise}.
\end{cases}
$$

Select $i_k \in U_k$ with probability $\Pr (\text{row} = i_k) = \frac{|r_k^{(i_k)}|^2}{\|r_k\|_2^2}$. Set

$$
x_{k+1} = x_k + \frac{r_k^{(i_k)}}{\|A^{(i_k)}\|_2^2} (A^{(i_k)})^*.
$$

end for
From the definitions of $\epsilon_k$ and $U_k$ in Algorithm 2.1, we have that if $\ell \in U_k$, then

$$\frac{|r_k^{(\ell)}|^2}{\|A^{(\ell)}\|_2^2} \geq \frac{1}{2} \max_{1 \leq i_k \leq m} \left\{ \frac{|r_k^{(i_k)}|^2}{\|A^{(i_k)}\|_2^2} \right\} + \frac{\|r_k\|_2^2}{\|A\|_2^2}.$$  

Note that

$$\max_{1 \leq i_k \leq m} \left\{ \frac{|r_k^{(i_k)}|^2}{\|A^{(i_k)}\|_2^2} \right\} \geq \sum_{i_k=1}^{m} \frac{\|A^{(i_k)}\|_2^2}{\|A\|_2^2} \frac{|r_k^{(i_k)}|^2}{\|A^{(i_k)}\|_2^2} = \frac{\|r_k\|_2^2}{\|A\|_2^2}.$$  

Thus, we can’t conclude that if $\ell \in U_k$, then

$$\frac{|r_k^{(\ell)}|^2}{\|A^{(\ell)}\|_2^2} \geq \max_{1 \leq i_k \leq m} \left\{ \frac{|r_k^{(i_k)}|^2}{\|A^{(i_k)}\|_2^2} \right\}, \text{ i.e., } \frac{|r_k^{(\ell)}|^2}{\|A^{(\ell)}\|_2^2} = \max_{1 \leq i_k \leq m} \left\{ \frac{|r_k^{(i_k)}|^2}{\|A^{(i_k)}\|_2^2} \right\}.$$  

As a result, there may exist some $\ell \in U_k$ such that

$$(2.1) \quad \frac{|r_k^{(\ell)}|^2}{\|A^{(\ell)}\|_2^2} < \max_{1 \leq i_k \leq m} \left\{ \frac{|r_k^{(i_k)}|^2}{\|A^{(i_k)}\|_2^2} \right\}.$$  

Meanwhile, from the update formula, for any $i_k \in U_k$, we have

$$(2.2) \quad \|x_{k+1} - x_k\|_2^2 = \frac{|r_k^{(i_k)}|^2}{\|A^{(i_k)}\|_2^2}.$$  

Thus, combining (2.1) and (2.2), we can find that we can’t make sure any row with the index from the index set $U_k$ make the distance between $x_{k+1}$ and $x_k$ be the largest when finding $x_{k+1}$. Moreover, to compute $\epsilon_k$, we have to compute the norm of each row of the matrix $A$.

Based on the GRK method, Bai and Wu [2] further designed the RGRK method by introducing a relaxation parameter, which is listed in Algorithm 2.2.

It is easy to see that when $\theta = \frac{1}{2}$, the RGRK method is just the GRK method. Bai and Wu [2] showed that the convergence factor of the RGRK method is smaller than that of the GRK method when $\theta \in \left[\frac{1}{2}, 1\right)$, and the convergence factor reaches the minimum when $\theta = 1$. For the latter case, we have that if $\ell \in V_k$, then

$$\frac{|r_k^{(\ell)}|^2}{\|A^{(\ell)}\|_2^2} = \max_{1 \leq i_k \leq m} \left\{ \frac{|r_k^{(i_k)}|^2}{\|A^{(i_k)}\|_2^2} \right\}.$$  

From the analysis following the Algorithm 2.2, in this case, the row with the index from the index set $V_k$ can make the distance between $x_{k+1}$ and $x_k$ be the largest for any possible $x_{k+1}$. However, we still need to compute the norm of each row of the matrix $A$ when computing $\epsilon_k$.

3. A novel greedy Kaczmarz method. On basis of the analysis of the GRK and RGRK methods and inspired by some recent works on selection strategy for working index based on the maximum residual [16, 9, 17], we design a new method for solving consistent linear systems which
Algorithm 2.2 The RGRK method

Input: $A \in \mathbb{C}^{m \times n}$, $b \in \mathbb{C}^m$, $\theta \in [0, 1]$, $\ell$, initial estimate $x_0$

Output: $x_\ell$

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

Compute

$$
\varepsilon_k = \frac{\theta}{\|r_k\|_2^2} \max_{1 \leq i_k \leq m} \left\{ \frac{|r_k^{(i_k)}|^2}{\|A^{(i_k)}\|_2^2} \right\} + 1 - \theta \frac{\|A\|_F^2}{\|r_k\|_2^2}.
$$

Determine the index set of positive integers

$$
\mathcal{V}_k = \left\{ i_k \bigg| \frac{|r_k^{(i_k)}|^2}{\|r_k\|_2^2} \geq \varepsilon_k \frac{\|r_k\|_2^2}{\|A^{(i_k)}\|_2^2} \right\}.
$$

Compute the $i$th entry $\tilde{r}_k^{(i)}$ of the vector $\tilde{r}_k$ according to

$$
\tilde{r}_k^{(i)} = \begin{cases} 
  r_k^{(i)}, & \text{if } i \in \mathcal{V}_k, \\
  0, & \text{otherwise}.
\end{cases}
$$

Select $i_k \in \mathcal{V}_k$ with probability

$$
\Pr(\text{row } = i_k) = \frac{|r_k^{(i_k)}|^2}{\|r_k\|_2^2}.
$$

Set

$$
x_{k+1} = x_k + \frac{r_k^{(i_k)}}{\|A^{(i_k)}\|_2^2} (A^{(i_k)})^*.
$$

end for

includes two main steps. In the first step, we use the maximum entries of the residual vector $r_k$ to determine an index set $\mathcal{R}_k$ whose specific definition is given in Algorithm 3.1. In the second step, we capture an index from the set $\mathcal{R}_k$ with which we can make sure the distance between $x_k$ and $x_{k+1}$ be the largest for any possible $x_{k+1}$. On a high level, the new method seems to change the order of the two main steps of Algorithm 2.1 or Algorithm 2.2. However, comparing with the GRK and RGRK methods, we do not need to calculate the norm of each row of the matrix $A$ any longer in Algorithm 3.1, and, like the RGRK method with $\theta = 1$, our method always makes the distance between $x_k$ and $x_{k+1}$ be the largest when finding $x_{k+1}$. In fact, the new method combines the maximum residual rule and the maximum distance rule. These characters make the method reduce the computation cost at each iteration and hence behaves better in the computing time, which is confirmed by numerical experiments given in Section 4.

Based on the above introduction, we propose the following algorithm, i.e., Algorithm 3.1.

Remark 3.1. Note that if

$$
|r_k^{(i_k)}| = \max_{1 \leq i \leq m} |r_k^{(i)}|,
$$

then $i_k \in \mathcal{R}_k$. So the index set $\mathcal{R}_k$ in Algorithm 3.1 is nonempty for all iteration index $k$. 
Algorithm 3.1 The GK method

Input: $A \in \mathbb{C}^{m \times n}$, $b \in \mathbb{C}^m$, $\ell$, initial estimate $x_0$
Output: $x_\ell$

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

Determine the index set of positive integers
\[ R_k = \left\{ \tilde{i}_k \mid \tilde{i}_k = \arg \max_{1 \leq i \leq m} r_{ik} \right\}. \]

Compute
\[ i_k = \arg \max_{\tilde{i}_k \in R_k} \left\{ \frac{r_{ik}^2}{\|A(\tilde{i}_k)\|_2^2} \right\}. \]

Set
\[ x_{k+1} = x_k + \frac{r_{ik}}{\|A(i_k)\|_2^2} (A(i_k))^\ast. \]
end for

Remark 3.2. Like Algorithm 2.1 or Algorithm 2.2, we can use the values of $\frac{r_{ik}^2}{\|A(i_k)\|_2^2}$ for $\tilde{i}_k \in R_k$ as a probability selection criterion to devise a randomized version of Algorithm 3.1. In this case, the convergence factor may be a little worse than that of Algorithm 3.1 because, for the latter, the index is selected based on the largest value of $\frac{r_{ik}^2}{\|A(i_k)\|_2^2}$ for $\tilde{i}_k \in R_k$, which make the distance between $x_{k+1}$ and $x_k$ be the largest for any possible $x_{k+1}$.

Remark 3.3. To the best of our knowledge, the idea of Algorithm 3.1 is brand new in the fields of designing greedy Kaczmarz type algorithms and we don’t find it in any work on greedy Gauss-Seidel methods either. So it is interesting to apply this idea to Gauss-Seidel methods to devise some new greedy Gauss-Seidel algorithms for solving other problems like large linear least squares problems. We will consider this topic in a subsequent paper.

In the following, we give the convergence theorem of the GK method.

Theorem 3.4. The iteration sequence $\{x_k\}_{k=0}^{\infty}$ generated by Algorithm 3.1, starting from an initial guess $x_0 \in \mathbb{C}^n$ in the column space of $A^\ast$, converges linearly to the least-Euclidean-norm solution $x^\ast = A^\dagger b$ and

\[
\|x_1 - x^\ast\|_2^2 \leq \left( 1 - \frac{1}{|R_0|} \sum_{i_0 \in R_0} \frac{1}{\|A(i_0)\|_2^2} \cdot \frac{1}{m} \cdot \lambda_{\min}(A^\ast A) \right) \cdot \|x_0 - x^\ast\|_2^2.
\]
which implies that
\[
\|x_{k+1} - x_\star\|^2 \leq \left(1 - \frac{1}{|\mathcal{R}_k|} \cdot \sum_{i_k \in \mathcal{R}_k} \frac{1}{\|A(i_k)\|^2} \cdot \frac{1}{m-1} \cdot \lambda_{\min}(A^*A) \right) \cdot \|x_k - x_\star\|^2, \quad k = 1, 2, \ldots
\]

Moreover, let \(\alpha = \max\{|\mathcal{R}_k|\}, \beta = \max\{\sum_{i_k \in \mathcal{R}_k} \|A(i_k)\|^2\}, k = 0, 1, 2, \ldots\). Then,
\[
\|x_k - x_\star\|^2 \leq \left(1 - \frac{\lambda_{\min}(A^*A)}{\alpha \cdot \beta \cdot (m-1)} \right)^{k-1} \left(1 - \frac{\lambda_{\min}(A^*A)}{|\mathcal{R}_0|} \cdot \sum_{i_0 \in \mathcal{R}_0} \|A(i_0)\|^2 \cdot m \right) \cdot \|x_0 - x_\star\|^2, \quad k = 1, 2, \ldots
\]

**Proof.** From the update formula in Algorithm 3.1, we have
\[
x_{k+1} - x_k = \frac{r_{(i_k)}^k}{\|A(i_k)\|^2} (A(i_k))^\star,
\]
which implies that \(x_{k+1} - x_k\) is parallel to \((A(i_k))^\star\). Meanwhile,
\[
A(i_k)(x_{k+1} - x_\star) = A(i_k) \left( x_k - x_\star + \frac{r_{(i_k)}^k}{\|A(i_k)\|^2} (A(i_k))^\star \right)
= A(i_k)(x_k - x_\star) + r_{(i_k)}^k,
\]
which together with the fact \(Ax_\star = b\) gives
\[
A(i_k)(x_{k+1} - x_\star) = (A(i_k)x_k - b(i_k)) + (b(i_k) - A(i_k)x_k) = 0.
\]
Then \(x_{k+1} - x_\star\) is orthogonal to \(A(i_k)\). Thus, the vector \(x_{k+1} - x_k\) is perpendicular to the vector \(x_{k+1} - x_\star\). By the Pythagorean theorem, we get
\[
\|x_{k+1} - x_\star\|^2 = \|x_k - x_\star\|^2 - \|x_{k+1} - x_k\|^2.
\]

On the other hand, from Algorithm 3.1, we have
\[
|r_{k}^{(i)}| = \max_{1 \leq i \leq m} |r_{k}^{(i)}| \quad \text{and} \quad \frac{|r_{k}^{(i)}|^2}{\|A(i_k)\|^2} = \max_{i \in \mathcal{R}_k} \frac{|r_{k}^{(i)}|^2}{\|A(i)\|^2}.
\]

Then
\[
\|x_{k+1} - x_k\|^2 = \frac{|r_{k}^{(i_k)}|^2}{\|A(i_k)\|^2} \geq \sum_{i_k \in \mathcal{R}_k} \frac{|r_{k}^{(i_k)}|^2}{\|A(i_k)\|^2} \cdot \frac{|r_{k}^{(i)}|^2}{\|A(i)\|^2} \geq \sum_{i_k \in \mathcal{R}_k} \frac{1}{|\mathcal{R}_k|} \cdot \frac{|r_{k}^{(i_k)}|^2}{\|A(i_k)\|^2} \geq \sum_{i_k \in \mathcal{R}_k} \frac{1}{|\mathcal{R}_k|} \cdot \frac{|r_{k}^{(i_k)}|^2}{\|A(i_k)\|^2} \geq \frac{\max_{1 \leq i \leq m} |r_{k}^{(i)}|^2}{\|A(i_k)\|^2}.
\]

(3.5)
Thus, substituting (3.5) into (3.4), we obtain

\[(3.6) \quad \|x_{k+1} - x^*\|_2^2 \leq \|x_k - x^*\|_2^2 - \sum_{i_k \in \mathcal{K}_k} \frac{1}{|\mathcal{R}_k|} \max_{1 \leq i \leq m} \left| r_k^{(i)} \right|^2 \frac{\max_{1 \leq i \leq m} \left| r_k^{(i)} \right|^2}{\|A(i_k)^*\|_2^2}. \]

For \(k = 0\), we have

\[\max_{1 \leq i \leq m} \left| r_0^{(i)} \right|^2 = \max_{1 \leq i \leq m} \left| r_0^{(i)} \right|^2 \cdot \frac{\|r_0\|_2^2}{\sum_{i=1}^m \left| r_0^{(i)} \right|^2} \geq \frac{1}{m} \cdot \|r_0\|_2^2, \]

which together with a result from [1]:

\[(3.7) \quad \|Ax\|_2^2 \geq \lambda_{\min}(A^*A) \|x\|_2^2 \]

is valid for any vector \(x\) in the column space of \(A^*\), implies

\[(3.8) \quad \max_{1 \leq i \leq m} \left| r_0^{(i)} \right|^2 \geq \frac{1}{m} \cdot \lambda_{\min}(A^*A) \cdot \|x_* - x_0\|_2^2. \]

Thus, substituting (3.8) into (3.6), we obtain

\[\|x_1 - x_*\|_2^2 \leq \|x_0 - x_*\|_2^2 - \sum_{i_0 \in \mathcal{K}_0} \frac{1}{|\mathcal{R}_0|} \cdot \frac{1}{\|A(i_0)^*\|_2^2} \cdot \frac{1}{m} \cdot \lambda_{\min}(A^*A) \cdot \|x_0 - x_*\|_2^2 \]

\[= \left(1 - \frac{1}{|\mathcal{R}_0|} \cdot \sum_{i_0 \in \mathcal{K}_0} \|A(i_0)^*\|_2^2 \right) \cdot \frac{1}{m} \cdot \lambda_{\min}(A^*A) \cdot \|x_0 - x_*\|_2^2, \]

which is just the estimate (3.1).

For \(k \geq 1\), we have

\[\max_{1 \leq i \leq m} \left| r_k^{(i)} \right|^2 = \max_{1 \leq i \leq m} \left| r_k^{(i)} \right|^2 \cdot \frac{\|r_k\|_2^2}{\sum_{i=1}^m \left| r_k^{(i)} \right|^2}. \]

Note that, according to the update formula in Algorithm 3.1, it is easy to obtain

\[r_k^{(i_k-1)} = b^{(i_k-1)} - A^{(i_k-1)} x_k \]

\[= b^{(i_k-1)} - A^{(i_k-1)} \left(x_{k-1} + \frac{r_k^{(i_k-1)}}{\|A^{(i_k-1)}\|_2}\left(A^{(i_k-1)}\right)^* \right) \]

\[= b^{(i_k-1)} - A^{(i_k-1)} x_{k-1} - r_k^{(i_k-1)} \]

\[(3.9) \quad = 0. \]

Then

\[\max_{1 \leq i \leq m} \left| r_k^{(i)} \right|^2 = \max_{1 \leq i \leq m} \left| r_k^{(i)} \right|^2 \cdot \frac{\|r_k\|_2^2}{\sum_{i=1}^m \left| r_k^{(i)} \right|^2} \geq \frac{1}{m-1} \cdot \|r_k\|_2^2, \]
which together with (3.7) yields
\begin{equation}
\max_{1 \leq i \leq m} \| p_k^{(i)} \|^2 \geq \frac{1}{m-1} \cdot \lambda_{\min} (A^*A) \| x_* - x_k \|^2.
\end{equation}
(3.10)

Thus, substituting (3.10) into (3.6), we get
\begin{equation}
\| x_{k+1} - x_* \|^2 \leq \| x_k - x_* \|^2 - \sum_{i=1}^{\infty} \frac{1}{|R_k|} \cdot \| A \|^2 \cdot \frac{1}{m-1} \cdot \lambda_{\min} (A^*A) \| x_k - x_* \|^2
\end{equation}
(3.11)

So the estimate (3.2) is obtained. By induction on the iteration index \( k \), we can get the estimate (3.3).

\[ \square \]

Remark 3.5. Since \( 1 \leq \alpha \leq m \) and \( \min_{1 \leq i \leq m} \| A^{(i)} \|^2 \leq \beta \leq \| A \|^2 \), it holds that
\[ \left( 1 - \frac{\lambda_{\min} (A^*A)}{\min_{1 \leq i \leq m} \| A^{(i)} \|^2 \cdot (m - 1)} \right) \leq \left( 1 - \frac{\lambda_{\min} (A^*A)}{\alpha \cdot \beta \cdot (m - 1)} \right) \leq \left( 1 - \frac{\lambda_{\min} (A^*A)}{m \cdot \| A \|^2 \cdot (m - 1)} \right). \]

Hence, the convergence factor of the GK method is small when the parameters \( \alpha \) and \( \beta \) are small. So, the smaller size of \( |R_k| \) is, the better convergence factor of the GK method is when \( \beta \) is fixed. From the definitions of \( U_k \), \( V_k \), and \( R_k \), we can find that the size of \( |R_k| \) may be smaller than those of \( |U_k| \) and \( |V_k| \). This is one of the reasons that our algorithm behaves better in computing time.

Remark 3.6. If \( \alpha = 1 \) and \( \beta = \min_{1 \leq i \leq m} \| A^{(i)} \|^2 \), the right side of (3.2) is smaller than
\[ \left( 1 - \frac{1}{\min_{1 \leq i \leq m} \| A^{(i)} \|^2 \cdot (m - 1)} \lambda_{\min} (A^*A) \right) \| x_k - x_* \|^2. \]

Since
\begin{equation}
\min_{1 \leq i \leq m} \| A^{(i)} \|^2 \cdot (m - 1) \leq \| A \|^2 - \min_{1 \leq i \leq m} \| A^{(i)} \|^2 < \| A \|^2,
\end{equation}
(3.12)

which implies
\[ \frac{1}{\min_{1 \leq i \leq m} \| A^{(i)} \|^2 \cdot (m - 1)} > \frac{1}{\| A \|^2} \left( \frac{1}{\| A \|^2 - \min_{1 \leq i \leq m} \| A^{(i)} \|^2} + \frac{1}{\| A \|^2} \right). \]

we have
\[ \left( 1 - \frac{1}{\min_{1 \leq i \leq m} \| A^{(i)} \|^2 \cdot (m - 1)} \lambda_{\min} (A^*A) \right) \| x_k - x_* \|^2 \]
(3.13)

< \left( 1 - \frac{1}{\| A \|^2 - \min_{1 \leq i \leq m} \| A^{(i)} \|^2} + \frac{1}{\| A \|^2} \right) \lambda_{\min} (A^*A) \| x_k - x_* \|^2.
Note that the error estimate in expectation of the GRK method in \([1]\) is
\[
E_k \|x_{k+1} - x^*\|_2^2 \leq \left( 1 - \frac{1}{2} \left( \frac{1}{\|A\|_F^2} - \min_{1 \leq i \leq m} \|A(i)\|_2^2 + \frac{1}{\|A\|_F^2} \right) \right) \lambda_{\min}(A^*A) \|x_k - x^*\|_2^2,
\]
where \(k = 1, 2, \ldots\). So the convergence factor of GK method is slightly better for the above case.

For the RGRK method, its error estimate in expectation given in \([2]\) is
\[
E_k \|x_{k+1} - x^*\|_2^2 \leq \left( 1 - \left( \frac{\theta}{\|A\|_F^2} - \min_{1 \leq i \leq m} \|A(i)\|_2^2 + 1 - \theta \right) \lambda_{\min}(A^*A) \right) \|x_k - x^*\|_2^2,
\]
where \(k = 1, 2, \ldots\). The estimate attains its minimum at \(\theta = 1\), which is
\[
\left( 1 - \frac{1}{\|A\|_F^2} - \min_{1 \leq i \leq m} \|A(i)\|_2^2 \lambda_{\min}(A^*A) \right) \|x_k - x^*\|_2^2.
\]
Considering (3.12) and similar to derivation of (3.13), we can get that when \(\alpha = 1\) and \(\beta = \min_{1 \leq i \leq m} \|A(i)\|_2^2\), the convergence factor of GK method is also slightly better than that of the RGRK method.

4. Experimental results. In this section, we compare the GRK, RGRK and GK methods with the matrix \(A \in \mathbb{C}^{m \times n}\) from two sets. One is generated randomly by using the MATLAB function `randn`, and the other includes some full-rank sparse matrices (e.g., ch7-8-b1, ch8-8-b1, model1, Trec8, Stranke94 and mycielskian5) and some rank-deficient sparse matrices (e.g., flower5_1, relat6, D_11, Sandi_sandi, GD01_c and GD02_a) originating in different applications from [5]. They possess certain structures, such as square \((m = n)\) (e.g., Stranke94, mycielskian5, GD01_c and GD02_a), thin \((m > n)\) (e.g., ch7-8-b1, ch8-8-b1, flower5_1 and relat6 ) or fat \((m < n)\) (e.g., model1, Trec8, D_11 and Sandi_sandi), and some properties, such as symmetric (e.g., Stranke94 and mycielskian5) or nonsymmetric (e.g., GD01_c and GD02_a).

We compare the three methods mainly in terms of the iteration numbers (denoted as “IT”) and the computing time in seconds (denoted as “CPU”). It should be pointed out here that the IT and CPU listed in our numerical results denote the arithmetical averages of the required iteration numbers and the elapsed CPU times with respect to 50 times repeated runs of the corresponding methods, and we always set \(\theta = 1\) in the RGRK method in our experiments since the convergence factor attains its minimum in this case. To give an intuitive compare of the three methods, we also present the iteration number speed-up of GK against GRK, which is defined as
\[
\text{IT speed-up}_1 = \frac{\text{IT of GRK}}{\text{IT of GK}},
\]
the iteration number speed-up of GK against RGRK, which is defined as
\[
\text{IT speed-up}_2 = \frac{\text{IT of RGRK}}{\text{IT of GK}}.
\]
HANYU LI AND YANJUN ZHANG

Table 1

| m x n | 1000 x 50 | 2000 x 50 | 3000 x 50 | 4000 x 50 | 5000 x 50 |
|-------|-----------|-----------|-----------|-----------|-----------|
| IT    | GRK       | 88.7600   | 79.3200   | 75.4200   | 74.1200   | 72.3000   |
|       | RGRK      | 67.0000   | 57.0000   | 50.0000   | 51.0000   | 48.0000   |
|       | GK        | 77.0000   | 64.0000   | 58.0000   | 54.0000   | 52.0000   |
| speed-up | 1.1927   | 1.2394   | 1.3003   | 1.3726   | 1.3904   |
| CPU   | GRK       | 0.0475    | 0.0606    | 0.0681    | 0.1241    | 0.1416    |
|       | RGRK      | 0.0300    | 0.0353    | 0.0394    | 0.0862    | 0.0928    |
|       | GK        | 0.0066    | 0.0084    | 0.0094    | 0.0223    | 0.0278    |
| speed-up | 7.2381   | 7.1852   | 7.2667   | 5.5915   | 5.0899   |

Table 2

| m x n | 1000 x 100 | 2000 x 100 | 3000 x 100 | 4000 x 100 | 5000 x 100 |
|-------|------------|------------|------------|------------|------------|
| IT    | GRK        | 205.0400   | 167.8400   | 157.2600   | 152.0000   | 146.8600   |
|       | RGRK       | 177.0000   | 129.0000   | 120.0000   | 114.0000   | 110.0000   |
|       | GK         | 183.0000   | 137.0000   | 122.0000   | 122.0000   | 113.0000   |
| speed-up | 11.2894 | 12.2511 | 12.6900 | 12.9559 | 12.9986 |
| CPU   | GRK        | 0.0959    | 0.0972    | 0.1163    | 0.2744    | 0.3409    |
|       | RGRK       | 0.0844    | 0.0747    | 0.0912    | 0.2172    | 0.2512    |
|       | GK         | 0.0187    | 0.0256    | 0.0291    | 0.0663    | 0.0791    |
| speed-up | 5.1167   | 4.7927   | 4.1415   | 4.3123   | 4.3123   |
| speed-up | 4.5714   | 4.1852   | 4.2000   | 3.8873   | 3.3771   |

the computing time speed-up of GK against GRK, which is defined as

\[
\text{CPU speed-up}_1 = \frac{\text{CPU of GRK}}{\text{CPU of GK}},
\]

and the computing time speed-up of GK against RGRK, which is defined as

\[
\text{CPU speed-up}_2 = \frac{\text{CPU of RGRK}}{\text{CPU of GK}}.
\]

In addition, for the sparse matrices from [5], we define the density as follows

\[
\text{density} = \frac{\text{number of nonzero of an } m \times n \text{ matrix}}{mn},
\]

and use \(\text{cond}(A)\) to represent the Euclidean condition number of the matrix \(A\).

In our specific experiments, the solution vector \(x_\star\) is generated randomly by the MATLAB function \texttt{randn} and we set the right-hand side \(b = Ax_\star\). All the test problems are started from an initial zero vector \(x_0 = 0\) and terminated once the relative solution error (RES), defined by

\[
\text{RES} = \frac{\|x_k - x_\star\|^2_2}{\|x_\star\|^2_2},
\]

satisfies \(\text{RES} \leq 10^{-6}\) or the number of iteration exceeds 200,000.

For the first class of matrices, that is, the randomly generated matrices, the numerical results on IT and CPU are listed in Tables 1 to 4 when \(m > n\), and in Tables 5 to 8 when \(m < n\). From Tables 1 to 8, we see that the GK method requires almost the same number of iterations as those
Table 3

| m x n          | 1000 x 150 | 2000 x 150 | 3000 x 150 | 4000 x 150 | 5000 x 150 |
|----------------|------------|------------|------------|------------|------------|
| IT GRK         | 364.6800   | 276.0200   | 249.5800   | 233.6800   | 226.4000   |
| RGRK           | 318        | 239        | 199        | 189        | 179        |
| GK             | 321        | 245        | 202        | 192        | 183        |
| speed-up 1     | 1.1961     | 1.1205     | 1.1455     | 1.2174     | 1.2394     |
| speed-up 2     | 0.9097     | 0.9755     | 0.9851     | 0.9844     | 0.9781     |
| CPU GRK        | 0.1906     | 0.1734     | 0.2712     | 0.6228     | 0.8194     |
| RGRK           | 0.1675     | 0.1572     | 0.2081     | 0.3209     | 0.6447     |
| GK             | 0.0462     | 0.0500     | 0.0737     | 0.1556     | 0.2391     |
| speed-up 1     | 4.1216     | 3.4687     | 3.0750     | 4.0920     | 4.3778     |
| speed-up 2     | 3.6216     | 3.1437     | 2.8220     | 3.4474     | 2.7386     |

Table 4

| m x n          | 1000 x 200 | 2000 x 200 | 3000 x 200 | 4000 x 200 | 5000 x 200 |
|----------------|------------|------------|------------|------------|------------|
| IT GRK         | 557.7000   | 398.6800   | 351.0400   | 328.3800   | 312.2400   |
| RGRK           | 517        | 341        | 294        | 277        | 257        |
| GK             | 504        | 334        | 294        | 284        | 258        |
| speed-up 1     | 1.1065     | 1.1337     | 1.1940     | 1.2139     | 1.2102     |
| speed-up 2     | 1.0258     | 1.0210     | 1        | 1.0492     | 0.9961     |
| CPU GRK        | 0.2706     | 0.2797     | 0.4300     | 1.1756     | 1.3834     |
| RGRK           | 0.2066     | 0.2425     | 0.4034     | 1.0497     | 1.1747     |
| GK             | 0.0741     | 0.0793     | 0.1197     | 0.3753     | 0.5031     |
| speed-up 1     | 3.6450     | 3.5375     | 3.5927     | 3.1407     | 2.7492     |
| speed-up 2     | 3.6641     | 3.0672     | 3.3708     | 2.7098     | 2.3438     |

of the GRK and RGRK methods but the GK method is more efficient in term of the computing time. The computing time speed-up of GK against GRK is at least 1.6951 (see Table 8 for the 200 x 4000 matrix) and at most 7.2667 (see Table 1 for the 3000 x 50 matrix), and the computing time speed-up of GK against RGRK is at least 1.6144 (see Table 8 for the 200 x 5000 matrix) and at most 4.5714 (see Table 1 for the 1000 x 50 matrix).

For the second class of matrices, that is, the sparse matrices from [5], the numerical results on IT and CPU are listed in Table 9 when the matrices are full-rank with different m and n, and in Table 10 when the matrices are rank-deficient with different m and n. In both tables, the iteration numbers of the GRK, RGRK and GK methods are almost the same, but again the CPUs of the GK method are smaller than those of the other two methods, with the CPU speed-up of GK against GRK being at least 2.3475 (the matrix D_11 in Table 10) and at most 6.0586 (the matrix Strank94 in Table 9), and the CPU speed-up of GK against RGRK being at least 2.3136 (the matrix D_11 in Table 10) and at most 4.3711 (the matrix Strank94 in Table 9).

Therefore, in all the cases, although the GK method requires almost the same number of iterations as those of the GRK and RGRK methods, our method outperforms the others in term of the computing time, which is consistent with the analysis before Algorithm 3.1.
Table 5

| m x n    | 50 x 1000 | 50 x 2000 | 50 x 3000 | 50 x 4000 | 50 x 5000 |
|----------|-----------|-----------|-----------|-----------|-----------|
| IT       |           |           |           |           |           |
| GRK      | 127.6600  | 114.3800  | 100.9000  | 97.7000   | 95.2000   |
| RGRK     | 127       | 118       | 99        | 91        | 91        |
| GK       | 126       | 117       | 98        | 92        | 91        |
| speed-up | 1.0132    | 0.9776    | 1.0300    | 1.0620    | 1.0462    |
| CPU      |           |           |           |           |           |
| GRK      | 0.0594    | 0.0669    | 0.0650    | 0.1247    | 0.1553    |
| RGRK     | 0.0581    | 0.0625    | 0.0638    | 0.1166    | 0.1412    |
| GK       | 0.0172    | 0.0275    | 0.0313    | 0.0625    | 0.0766    |
| speed-up | 3.4545    | 2.4318    | 2.0800    | 1.9950    | 2.0408    |

Table 6

| m x n    | 100 x 1000 | 100 x 2000 | 100 x 3000 | 100 x 4000 | 100 x 5000 |
|----------|------------|------------|------------|------------|------------|
| IT       |            |            |            |            |            |
| GRK      | 285.1800   | 264.6200   | 232.9000   | 217.3200   | 212.3800   |
| RGRK     | 276        | 255        | 232        | 215        | 208        |
| GK       | 268        | 256        | 226        | 214        | 209        |
| speed-up | 1.0641    | 1.0337    | 1.0305    | 1.0155    | 1.0211    |
| CPU      |            |            |            |            |            |
| GRK      | 0.1412     | 0.1638     | 0.2197     | 0.4278     | 0.5150     |
| RGRK     | 0.1375     | 0.1497     | 0.2172     | 0.4253     | 0.5011     |
| GK       | 0.0431     | 0.0622     | 0.0788     | 0.1747     | 0.2122     |
| speed-up | 3.2754    | 2.6332    | 2.7897    | 2.4490    | 2.4271    |

Table 7

| m x n    | 150 x 1000 | 150 x 2000 | 150 x 3000 | 150 x 4000 | 150 x 5000 |
|----------|------------|------------|------------|------------|------------|
| IT       |            |            |            |            |            |
| GRK      | 589.7600   | 441.2800   | 364.5600   | 342.4400   | 340.4400   |
| RGRK     | 580        | 432        | 355        | 331        | 330        |
| GK       | 586        | 427        | 358        | 338        | 321        |
| speed-up | 1.0064    | 1.0334    | 1.0377    | 0.9770    | 1.0211    |
| CPU      |            |            |            |            |            |
| GRK      | 0.3700     | 0.3922     | 0.4500     | 0.9569     | 1.2394     |
| RGRK     | 0.3531     | 0.3503     | 0.4416     | 0.9291     | 1.1884     |
| GK       | 0.0975     | 0.1291     | 0.250      | 0.5281     | 0.7097     |
| speed-up | 3.7949    | 3.0387    | 2.7143    | 1.9625    | 1.7592    |

Table 8

| m x n    | 200 x 1000 | 200 x 2000 | 200 x 3000 | 200 x 4000 | 200 x 5000 |
|----------|------------|------------|------------|------------|------------|
| IT       |            |            |            |            |            |
| GRK      | 940.4600   | 641.0400   | 540.6600   | 497.5000   | 471.0200   |
| RGRK     | 932        | 636        | 514        | 492        | 455        |
| GK       | 962        | 628        | 521        | 499        | 452        |
| speed-up | 0.9888    | 1.0268    | 1.0377    | 0.9770    | 1.0352    |
| CPU      |            |            |            |            |            |
| GRK      | 0.6241     | 0.6269     | 0.7834     | 1.7025     | 2.1916     |
| RGRK     | 0.6288     | 0.5909     | 0.7381     | 1.6322     | 2.0866     |
| GK       | 0.1766     | 0.2372     | 0.3756     | 1.0944     | 1.2925     |
| speed-up | 3.5345    | 2.7594    | 2.0857    | 1.6951    | 1.6096    |

IT and CPU of GRK, RGRK and GK for m-by-n matrices A with m = 50 and different n.

IT and CPU of GRK, RGRK and GK for m-by-n matrices A with m = 100 and different n.

IT and CPU of GRK, RGRK and GK for m-by-n matrices A with m = 150 and different n.

IT and CPU of GRK, RGRK and GK for m-by-n matrices A with m = 200 and different n.
### Table 9
**IT and CPU of GRK, RGRK and GK for m-by-n matrices A with different m and n.**

| name  | ch7-8-b1 | ch8-8-b1 | model1 | Trec8 | Strankes94 | mycielskian5 |
|-------|----------|----------|--------|-------|------------|-------------|
| m × n |          |          |        |       |            |             |
| full rank |         |          |        |       |            |             |
| density | 3.57%   | 3.13%    | 1.05%  | 28.42%| 90.00%     | 26.84%      |
| cond(A) | 4.79e+14| 3.48e+14 | 17.57  | 26.89 | 51.73      | 27.64       |
| IT     |         |          |        |       |            |             |
| GRK    | 103.9800| 113.0800 | 4.7484e+03| 1.7655e+03| 5.6706e+03| 4.2651e+03 |
| RGRK   | 87.0600 | 89.4504  | 16.46  | 41.61 | 36.36      | 41.69        |
| GK     | 87      | 89       | 41.83  | 16.81 | 36.36      | 41.69        |
| speed-up | 1.1922 | 1.2706   | 1.4342 | 1.6882| 1.9096     | 2.0239       |
| speed-up | 1.0007 | 1.0767   | 1.1436 | 1.2837| 1.3822     | 1.4237       |
| CPU    |         |          |        |       |            |             |
| GRK    | 0.0178  | 0.0200   | 0.5381 | 0.1619| 0.4847     | 0.3816       |
| RGRK   | 0.0116  | 0.0119   | 0.5128 | 0.1500| 0.3997     | 0.3700       |
| GK     | 0.0047  | 0.0047   | 0.1953 | 0.0441| 0.0800     | 0.0872       |
| speed-up | 3.8000 | 4.2667   | 4.7052 | 3.6738| 6.0586     | 4.4763       |
| speed-up | 2.4667 | 2.5333   | 2.6256 | 3.4083| 4.3711     | 4.2437       |

### Table 10
**IT and CPU of GRK, RGRK and GK for m-by-n matrices A with different m and n.**

| name  | flower 5_1 | related | D_IT | Sandi_pandil | CD01_c | CD02_c |
|-------|-------------|---------|------|--------------|-------|-------|
| m × n |             |         |      |              |       |       |
| full rank |         |         |      |              |       |       |
| density | 1.42 % | 2.21 % | 3.79 % | 0.54 % | 12.40 % | 16.45 % |
| cond(A) | 2.00e+16 | Inf     | 2.21e+17 | 1.47e+17 | Inf     | Inf   |
| IT     |           |         |      |              |       |       |
| GRK    | 9.8127e+03 | 1.6099e+03 | 682.8200 | 1756 | 1.9329e+03 | 1.392e+03 |
| RGRK   | 9.8616e+03 | 1.5199e+03 | 668  | 1.6864e+03 | 1819 | 1469  |
| GK     | 10521 | 1510 | 690 | 1787 | 1823 | 1228  |
| speed-up | 0.9327 | 1.1061 | 0.9896 | 0.9827 | 1.0663 | 1.1342 |
| speed-up | 0.9373 | 1.0965 | 0.9681 | 0.9437 | 0.9978 | 1.1963 |
| CPU    |           |         |      |              |       |       |
| GRK    | 1.0197 | 0.2550 | 0.0866 | 0.1812 | 0.1663 | 0.1241 |
| RGRK   | 0.9653 | 0.2353 | 0.0853 | 0.1741 | 0.1538 | 0.1219 |
| GK     | 0.3906 | 0.0766 | 0.0309 | 0.0600 | 0.0378 | 0.0303 |
| speed-up | 3.3919 | 3.3096 | 2.3475 | 3.0208 | 4.3967 | 4.0928 |
| speed-up | 3.2140 | 3.0738 | 2.3136 | 2.9870 | 4.0661 | 4.0256 |
REFERENCES

[1] Z. Z. Bai and W. T. Wu, On greedy randomized Kaczmarz method for solving large sparse linear systems, SIAM J. Sci. Comput., 40 (2018), pp. A592–A606.
[2] Z. Z. Bai and W. T. Wu, On relaxed greedy randomized Kaczmarz methods for solving large sparse linear systems, Appl. Math. Lett., 83 (2018), pp. 21–26.
[3] Z. Z. Bai and W. T. Wu, On greedy randomized coordinate descent methods for solving large linear least-squares problems, Numer. Linear Algebra Appl., 26 (2019), pp. 1–15.
[4] J. Q. Chen and Z. D. Huang, On the error estimate of the randomized double block Kaczmarz method, Appl. Math. Comput., 370 (2020), p. 124907.
[5] T. A. Davis and Y. F. Hu, The university of Florida sparse matrix collection, ACM. Trans. Math. Softw., 38 (2011), pp. 1–25.
[6] K. Du, Tight upper bounds for the convergence of the randomized extended Kaczmarz and Gauss-Seidel algorithms, Numer. Linear Algebra Appl., 26 (2019), p. e2233.
[7] K. Du and H. Gao, A new theoretical estimate for the convergence rate of the maximal weighted residual Kaczmarz algorithm, Numer. Math. Theor. Meth. Appl., 12 (2019), pp. 627–639.
[8] M. Griebel and P. Oswald, Greedy and randomized versions of the multiplicative Schwarz method, Linear Algebra Appl., 437 (2012), pp. 1596–1610.
[9] J. Haddock and D. Needell, On Motzkin’s method for inconsistent linear systems, BIT Numer. Math., 59 (2019), pp. 387–401.
[10] S. Kaczmarz, Angenäherte auflösung von systemen linearer gleichungen, Bull. Int. Acad. Pol. Sci. Lett. A, 35 (1937), pp. 355–357.
[11] Y. Liu and C. Q. Gu, Variant of greedy randomized Kaczmarz for ridge regression, Appl. Numer. Math., 143 (2019), pp. 223–246.
[12] A. Ma, D. Needell, and A. Ramdas, Convergence properties of the randomized extended Gauss-Seidel and Kaczmarz methods, SIAM J. Matrix Anal. Appl., 36 (2015), pp. 1590–1604.
[13] D. Needell, Randomized Kaczmarz solver for noisy linear systems, BIT Numer. Math., 50 (2010), pp. 395–403.
[14] N. Nguyen, D. Needell, and T. Woolf, Linear convergence of stochastic iterative greedy algorithms with sparse constraints, IEEE Trans. Inf. Theory, 63 (2017), pp. 6869–6895.
[15] Y. Q. Niu and B. Zheng, A greedy block Kaczmarz algorithm for solving large-scale linear systems, Appl. Math. Lett., 104 (2020), p. 106294.
[16] J. Nutini, Greed is Good: Greedy Optimization Methods for Large-scale Structured Problems, PhD thesis, University of British Columbia, 2018.
[17] E. Rebrova and D. Needell, Sketching for Motzkin’s iterative method for linear systems, Proc. 50th Asilomar Conf. on Signals, Systems and Computers, 2019.
[18] T. Strohmer and R. Vershynin, A randomized Kaczmarz algorithm with exponential convergence, J. Fourier Anal. Appl., 15 (2009), pp. 262–278.
[19] N. C. Wu and H. Xiang, Projected randomized Kaczmarz methods, J. Comput. Appl. Math., 372 (2020), p. 112672.
[20] A. Zouzias and M. N. Freris, Randomized extended Kaczmarz for solving least squares, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 773–793.