Journal of Applied Analysis and Computation
Volume 15, Number 5, October 2025, 2935–2958
Website:http://www.jaac-online.com
DOI:10.11948/20240569

A CLASS OF COORDINATE DESCENT METHODS WITH ANGLE PROBABILITY FOR SOLVING LINEAR SYSTEMS*

Wenjun Yu[1], Hailong Shen[1,†] and Xinhui Shao[1]

Abstract In order to resolve large-scale linear systems, we construct the new probability criterion based on the angle between hyperplanes and propose the randomized coordinate descent method with the angle probability criterion. Furthermore, we propose the randomized extended coordinate descent method with the angle probability criterion, which allows to solve underdetermined linear systems. The convergence properties of the two methods are analyzed from the expectation perspective and upper bounds on their convergence rate are derived when the matrix is full rank. Lastly, we conduct numerical experiments to verify the efficacy of our new methods.

Keywords Linear systems, randomized coordinate descent, randomized extended coordinate descent, convergence property, probability distribution.

MSC(2010) 65F10, 65K05, 68W20.

1. Introduction

We consider solving the large-scale linear system

$$Ax = b,\ A \in \mathbb{R}^{m\times n},\ b \in \mathbb{R}^m, \tag{1.1}$$

where matrix A is either full row rank or full column rank, b is a m dimensional vector and $x \in \mathbb{R}^n$ is an unknown vector. Due to the wide range of applications of this linear system in various fields, finding the solution of (1.1) has become a hot research topic. When the overdetermined linear system ($m > n$) is consistent, $x_\star = (A^TA)^{-1}A^Tb$ being the unique solution is expected to be found, when the overdetermined linear system is inconsistent, which means that there is no exact solution, we are committed to finding the least squares solution

$$x_{\mathrm{LS}} = \arg\min_x \|b - Ax\|_2^2, \tag{1.2}$$

and when the linear system is underdetermined ($m < n$), there are infinitely many solutions, so without special restrictions we usually would like to pick the least Euclidean norm solution $x_{\mathrm{LN}} = A^T(AA^T)^{-1}b$, where $\|A\|_2$, A^T and A^{-1} respectively represent the Euclidean norm, the transpose and the inverse of the matrix A.

†The corresponding author.
[1]Department of Mathematics, School of Science, Northeastern University, Shenyang 110819, China
*The authors were supported by Fundamental Research Funds for the Central Universities (N2224005-1).
Email: yuwenjun2000@126.com(W. Yu), hailong_shen@126.com(H. Shen), xinhui1002@126.com(X. Shao)

When dealing with large-scale linear systems, randomized iterative methods, like the randomized kaczmarz (RK) method [20] and randomized coordinate descent (RCD) method [9], are favoured for their fewer storage, lower computational complexity, and higher stability, whereas randomized coordinate descent is widely studied owing to its extensive applications in the fields of biology [4], signal processing [16], operational research [13], neural networks [23], and so on. In the RCD method, the update of the iterative solution is realized by conducting the line search along a search direction, where the search direction is randomly selected from the coordinate directions with a specific probability criterion. In addition, the coordinate descent (CD) method also can be obtained by application of classical Gauss-Seidel method on normal equation

$$A^T Ax = A^T b,$$

so that the RCD method has also been known as the randomized Gauss-Seidel (RGS) method.

Leventhal and Lewis [9] presented the RCD method in 2010, which is similar to the RK method in that it makes full use of the norm information of the matrix to select the search direction and converges linearly in expectation. Furthermore, Bai et al. [1] succeeded in deriving the precise closed form of the mean-squared residual produced by RCD method, and developed a more precise prediction of the upper bound on convergence rate for RCD method. For the large-scale optimization problem, in 2012, Liu [14] proposed a random coordinate descent method with the worst-case efficiency estimates. In order to decrease the computation cost in iterations, many researchers considered the block versions of the RCD method [11, 17, 24], where the core idea is to randomly select a portion of the columns of the matrix to form a block for iteration, for more research on block versions, please refer to [7, 10]. In addition, to further accelerate the total computation time, Rodomanov and Kropotov [18] applied a randomized coordinate selection strategy for volume sampling in the RCD method, as well as Niu and Zheng [15] provided a more efficient probability criterion based on residual vectors and developed the new randomised coordinate descent (NRCD) method.

When the matrix is normalised in advance, the columns of the coefficient matrix will be selected with the same probability in the RCD method, to address this shortcoming, the greedy randomized coordinate descent (GRCD) method was developed by Bai and Wu [3], which preferentially eliminates greater entries of residual vectors during each iteration to optimize the solution process. In addition, Zhang and Guo [25] further introduced a relaxation factor to achieve more flexible iterative adjustment in performing the greedy strategy, thus developed relaxed greedy randomized coordinate descent (RGRCD) methods. While Zhang et al. [21] improved the utilisation of the iteration product by adopting a multi-step strategy, which led to the proposal of multi-step greedy randomized coordinate descent (MGRCD) method. The numerical experimental results indicated that all of the above greedy versions of the RCD method are superior to the typical RCD method.

However, although the RCD method is effective for overdetermined linear systems, whether consistent or inconsistent, the RCD method adds components which orthogonal to the row span of the matrix A to the iterative solution x_k in the iterations, so that for underdetermined linear system, the RCD method cannot converge to the lest norm solution x_{LN}. In order to solve this problem, Ma et al. [12] used an extension strategy on the RCD method thus obtaining the randomized extended coordinate descent (RECD) method as well as established the theory for conver-

gence. Subsequently, Du [6] provided more tightly upper bounds of convergence for the REK method and the RECD method, in addition to the fact that these bounds hold true for various types of linear systems (1.1), while he gave the equivalent version of RECD showing that the RECD method establishes the connection between the RK and RCD methods. Recently, Wu and Xiang proposed a general version of randomized extended coordinate descent (GRECD) method, which can be achieved by converting rows or columns randomly selected in iterations into matrix form, and used two different sampling strategies, discrete sampling and Gaussian sampling, to accelerate the GRECD method in [22].

In this work, we deeply investigate the meaning of the geometric projection for the RCD method, aiming to select a hyperplane with a larger angle to the current hyperplane for projection in each iteration. By using the iterative solutions and residual vectors to construct the sine value of the angle and constructing a new probability criterion, we further propose the randomized coordinate descent method with angle probability (RCDA). Subsequently, we apply the new probability criterion to the first component update process in the RECD method as well as propose the randomized extended coordinate descent method with angle probability (RECDA). We also explore the convergence of these two new methods theoretically and fully verify the effectiveness of them through a series of numerical experiments.

The rest sections of this work are structured as below. We briefly introduced the classical RCD method and the RECD method in Section 2. Later, the RCDA method is presented and the upper bound on the convergence rate of this method is derived in Section 3. In Section 4, the RECDA method is developed and we derive the theory of convergence. Furthermore, we design and perform a series of numerical experiments to verify the superiority for RCDA and RECDA methods in Section 5. Finally, we summarize the main results of this work in Section 6.

2. The RCD and RECD methods

In this section, we firstly present notations used in this paper and then describe the RCD method [9], the RECD method [12] and an equivalent variant [6] as well as their convergence theorems.

Algorithm 1 The RCD Method

Input: A, b, K, x_0 and $r_0 = b - Ax_0$

Output: x_K

1: **for** $k = 0, 1 \ldots K-1$ **do**

2: Select $j_k \in \{1, 2, \ldots, n\}$ with probability

$$\Pr(\text{column} = j_k) = \frac{\left\|A_{(j_k)}\right\|_2^2}{\|A\|_F^2}$$

3: Compute $\alpha_k = \frac{\langle A_{(j_k)}, r_k \rangle}{\left\|A_{(j_k)}\right\|_2^2}$

4: Update $x_{k+1} = x_k + \alpha_k e_{j_k}$

5: Update $r_{k+1} = r_k - \alpha_k A_{(j_k)}$

6: **end for**

For any vector $v, w \in \mathbb{R}^n$, we represent their inner product with $\langle v, w\rangle$ and use w^T represent the transpose of w. For a matrix $D = (d_{ij}) \in \mathbb{R}^{m\times n}$, the $D^{(i)}$, $D_{(j)}$, $D^\dagger$, D^T, $\|D\|_2 = \max_{x\neq 0}\frac{\|Dx\|_2}{\|x\|_2}$ and $\|D\|_F = \sqrt{\sum_{j=1}^{n}\sum_{i=1}^{m}|d_{ij}|^2}$ respectively represent the ith row, the jth column, the Moore-Penrose, the transpose, the Euclidean norm, and the Frobenius norm of matrix D. Furthermore, we use $\sigma_{\min}(D)$, null(D), $R(D)$ and range(D) to represent the smallest nonzero singular value, the null space, the rank and the column space of matrix D. Besides, the $\|\mathrm{x}\|_D = \sqrt{\mathrm{x}^T D\mathrm{x}}$ denote the energy norm for the symmetric positive-define matrix $D \in \mathbb{R}^{n\times n}$, the e_i denote a column vector where the value at position i is 1 and the values at the other positions are 0.

Let $\mathbb{E}_{k+1}[\cdot]$ or $\mathbb{E}_{k+1}\cdot$ represent the expected value conditional on first $k+1$ iterations, namely,

$$\mathbb{E}_{k+1}[\cdot] = \mathbb{E}[\cdot \mid i_0, j_0, i_1, j_1, \ldots, i_k, j_k],$$

where $i_s, j_s (t = 0, 1, \ldots, k)$ represent row and column indices that are selected in the sth iteration, and $\mathbb{E}[\mathbb{E}_{k+1}[\cdot]] = \mathbb{E}[\cdot]$, $\mathbb{E}_{k+1}[\cdot] = \mathbb{E}^i_{k+1}[\mathbb{E}^j_{k+1}[\cdot]]$ are valid since the law of iterated expectation.

For overdetermined linear systems, the RCD method as presented by Leventhal and Lewis in [9] can be obtained by using a probability criterion constructed based on the Euclidean norm of matrix column in CD method, and the following Theorem 2.1 displays the convergence theory for RCD method in detail.

Algorithm 2 The RECD Method

Input: $A, b, K, x_0 \in \mathbb{R}^n$ and $z_0 \in x_0 + \text{range}(A^T)$
Output: x_K^{LN}

1: **for** $k = 0, 1, \ldots, K-1$ **do**
2: Select $j_k \in \{1, 2, \ldots, n\}$with probability

$$\Pr(\text{column} = j_k) = \frac{\|A_{(j_k)}\|_2^2}{\|A\|_F^2}$$

3: Update $x_{k+1} = x_k + \frac{A_{(j_k)}^T(b - Ax_k)}{\|A_{(j_k)}\|_2^2} e_{j_k}$
4: Select $i_k \in \{1, 2, \ldots, m\}$with probability

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

5: Set $P_i = I - \frac{\left(A^{(i_k)}\right)^T A^{(i_k)}}{\left\|A^{(i_k)}\right\|_2^2}$
6: Update $z_{k+1} = P_i\left(z_k + x_{k+1} - x_k\right)$
7: Update $x_{k+1}^{LN} = x_{k+1} - z_{k+1}$
8: **end for**

Theorem 2.1. [9] *For linear system (1.1), where $A \in \mathbb{R}^{m\times n}(m \geq n)$ and $R(A) = n$, let the initial guess vector $x_0 \in \mathbb{R}^n$, then the iteration sequence $\{x_k\}_{k=0}^{\infty}$ produced by the RCD method converges to the unique least squares solution $x_\star$ in expectation.*

Furthermore, the solution error of the sequence $\{x_k\}_{k=0}^{\infty}$ *is subject to*

$$\mathbb{E}\|x_k - x_\star\|_{A^TA}^2 \leq \left(1 - \frac{\sigma_{\min}^2(A)}{\|A\|_F^2}\right)^k \|x_0 - x_\star\|_{A^TA}^2, \quad k = 1, 2, \dots .$$

However, for the underdetermined linear systems, the RCD method is invalid since there is no way to converge to the corresponding least norm solution x_{LN}, to solve this problem, MA et al. extended RCD method as well as provided a theory of convergence of this method in [12]. They used randomized orthogonal projection for obtaining component in x that is orthogonal to the row space of matrix A and removed the component by iteration so that the iterative solution converges to least Euclidean norm solution. Specifically, the RECD method can be described as Algorithm 2.

Algorithm 3 The RECD-E Method

Input: $A, b, K, x_0 \in \mathbb{R}^n$ and $z_0 \in \text{range}(A^T)$
Output: z_K
1: **for** $k = 0, 1, \dots, K-1$ **do**
2: Select $j_k \in \{1, 2, \dots, n\}$with probability

$$\Pr(\text{column} = j_k) = \frac{\|A_{(j_k)}\|_2^2}{\|A\|_F^2}$$

3: Update $x_{k+1} = x_k + \frac{A_{(j_k)}^T(b - Ax_k)}{\|A_{(j_k)}\|_2^2} e_{j_k}$
4: Select $i_k \in \{1, 2, \dots, m\}$with probability

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

5: Update $z_{k+1} = z_k + \frac{\left(A^{(i_k)}\right)^T(x_{k+1} - z_k)}{\left\|A^{(i_k)}\right\|_2^2} \left(A^{(i_k)}\right)^T$
6: **end for**

Subsequently, in order to discuss the convergence for RECD method more conveniently, Du [6] provided a mathematically equivalent method to the RECD method, known as the RECD-E method. In the $k+1$ th iteration, the vector z_{k+1} in the RECD-E method is equivalent to x_{k+1}^{LN} in the RECD method, and at the same time, z_{k+1} can be regarded as the iterative solution obtained by applying the RK method on linear system $Az = Ax_{k+1}$.

The tighter upper bound for convergence of RECD method has been further developed by Du, which is suitable for linear systems of various types (underdetermined or overdetermined, inconsistent or consistent). The REGS-E method, as well as its convergence theorem, can be summarised as Algorithm 3 and Theorem 2.2.

Theorem 2.2. [6] *For linear system (1.1), where the matrix* A *is of full rank, let the initial guess vector* $x_0 \in \mathbb{R}^n$ *and* $z_0 \in \text{range}(A^T)$*, then the iteration sequence* $\{z_k\}_{k=0}^{\infty}$ *produced by the RECD method converges to the* $A^\dagger b$ *in expectation.*

Furthermore, the solution error of the sequence $\{z_k\}_{k=0}^{\infty}$ *is subject to*

$$\mathbb{E}\left\|z_{k+1}-A^{\dagger}b\right\|_2^2 \leq \varphi^{k+1}\left\|z_0-A^{\dagger}b\right\|_2^2+\frac{(k+1)\varphi^{k+1}}{\|A\|_F^2}\left\|Ax_0-AA^{\dagger}b\right\|_2^2,$$

where $\varphi=1-\frac{\sigma_{\min}^2(A)}{\|A\|_F^2}$.

Theorem 2.1 and Theorem 2.2 indicate that if matrix is reasonably well conditioned, meaning that its singular values are significantly far from the origin, the RCD method and the RECD method may achieve rapid convergence.

3. The RCDA method

Within the current section, we present the randomized coordinate descent method with angle probability (RCDA), which has a probability criterion with respect to the angle between the two hyperplanes, and analyse its convergence.

Used as a projection method, the RCD method projects the iterative solution x_k from hyperplane $A_{(j_{k-1})}^T Ax = A_{(j_{k-1})}^T b$ to $A_{(j_k)}^T Ax = A_{(j_k)}^T b$ at kth iteration to obtain until the iterative solution approximates the least squares solution x_{k+1}. Note that the x_{LS} is at the intersection between hyperplanes $\mathbb{H}_j,\ j=1,2\ldots,n$, here we define $\mathbb{H}_j$ as the hyperplane determined by the equation $A_{(j)}^T Ax = A_{(j)}^T b$, in order to achieve more rapid convergence, we want the iterative solution x_{k+1} to project onto the intersection of the two hyperplanes, which means that the chosen hyperplane $\mathbb{H}_{j_k}$ should be orthogonal to the hyperplane $\mathbb{H}_{j_{k-1}}$. However, in practical arithmetic, such a choice is almost non-existent, so it is a good strategy to choose the hyperplane $\mathbb{H}_{j_k}$ with a larger angle to hyperplane $\mathbb{H}_{j_{k-1}}$. Besides, it is also a good idea to choose a hyperplane that has a larger angle with residual vector $r_k = b - Ax_k$.

Let θ_{j_k} represents the angle between the hyperplane $\mathbb{H}_{j_{k-1}}$ and the residual vector r_k, define the probability as

$$p_{j_k}=\frac{\sin^2\theta_{j_k}}{\sum_{j_k=1}^{n}\sin^2\theta_{j_k}},\quad j_k=1,2,\ldots,n. \tag{3.1}$$

It is obvious that $\sum_{j_k=1}^{n} p_{j_k}=1$, so p_{j_k} can be used as the probability criterion select the j_kth column. By the construction of p_{j_k}, it follows that the probability that the j_kth column is chosen in the current iteration is proportional to the square of sine value for angle between hyperplane $\mathbb{H}_{j_{k-1}}$ and the residual vector r_k. When $\sin^2\theta_{j_{k1}} \geq \sin^2\theta_{j_{k2}}$, the probability of selecting j_{k1}th column is greater than the probability of selecting j_{k2}th column. In other words, the larger the angle between hyperplane $\mathbb{H}_{j_{k-1}}$ and the residual vector r_k, the probability that j_{k1}th column will be selected is larger.

Then the RCDA method can be described as following Algorithm 4.

From the iterative formula of RCDA method, it follows that

$$A(x_{k+1}-x_k)=\frac{A_{(j_k)}^T r_{j_k}}{\|A_{(j_k)}\|_2^2}A_{(j_k)} \tag{3.2}$$

Algorithm 4 The RCDA Method

Input: A, b, K, x_0 and $r_0 = b - Ax_0$
Output: x_K
1: **for** $k = 0, 1 \ldots K-1$ **do**
2: Select $j_k \in \{1, 2, \ldots, n\}$ with probability

$$\Pr(\text{column} = j_k) = \frac{\sin^2 \theta_{j_k}}{\sum_{j_k=1}^{n} \sin^2 \theta_{j_k}}$$

3: Compute $\alpha_k = \frac{\langle A_{(j_k)}, r_k \rangle}{\|A_{(j_k)}\|_2^2}$
4: Update $x_{k+1} = x_k + \alpha_k e_{j_k}$
5: Update $r_{k+1} = r_k - \alpha_k A_{(j_k)}$
6: **end for**

and

$$\begin{aligned}
\left\langle A_{(j_k)}, A(x_{k+1} - x_{LS}) \right\rangle &= A_{(j_k)}^T A(x_{k+1} - x_{LS}) \\
&= A_{(j_k)}^T A \left(x_k + \frac{\langle A_{(j_k)}, r_k \rangle}{\|A_{(j_k)}\|_2^2} e_{j_k} - x_{LS} \right) \\
&= A_{(j_k)}^T \left[A(x_k - x_{LS}) + \frac{A_{(j_k)}^T A(x_{LS} - x_k)}{\|A_{(j_k)}\|_2^2} A_{(j_k)} \right] \qquad (3.3) \\
&= A_{(j_k)}^T \left(I - \frac{A_{(j_k)} A_{(j_k)}^T}{\|A_{(j_k)}\|_2^2} \right) A(x_k - x_{LS}) \\
&= 0,
\end{aligned}$$

the equation (3.2) means that the vector $A(x_{k+1} - x_k)$ is parallel to the vector $A_{(j_k)}$ and the equation (3.3) means that the vector $A_{(j_k)}$ is orthogonal to the vector $A(x_{k+1} - x_{LS})$ respectively. Then it holds that the $A(x_{k+1} - x_k)$ is orthogonal to the $A(x_{k+1} - x_{LS})$ and according to the Pythagorean theorem, it follows that

$$\|A(x_k - x_{LS})\|_2^2 = \|A(x_{k+1} - x_k)\|_2^2 + \|A(x_{k+1} - x_{LS})\|_2^2. \qquad (3.4)$$

Based on the above analysis and the fact $\|A(x_k - x_\star)\|_2^2 = \|r_k\|_2^2$, it is clear that

$$\sin^2 \theta_{j_k} = \frac{\|A(x_{k+1} - x_k)\|_2^2}{\|A(x_k - x_{LS})\|_2^2} = \frac{A_{j_k}^T r_k}{\|A_{j_k}\|_2^2 \|r_k\|_2^2}. \qquad (3.5)$$

Remark 3.1. If we define θ_{j_k} as the angle between hyperplane $\mathbb{H}_{j_{k-1}}$ and hyperplane $\mathbb{H}_{j_k}$, then the equation (3.1) can still be used as the probability criterion to select the column in the current iteration, but different from the calculation of equation (3.5), we use the normal vectors of the hyperplanes to calculate the sine of the angle between two hyperplanes. However, in practical numerical experiments, it seems that the experimental results obtained from such calculations are not satisfactory, so how to better represent the angle between two hyperplanes seems to be a research worth doing in the future.

Remark 3.2. In practical calculations, the value of $\|r_k\|_2^2$ can be calculated without being explicitly calculated in the process of calculating equation (3.1), and combining equation (3.1) with (3.5), it follows that

$$p_{j_k} = \frac{\frac{\left|A_{(j_k)}^T r_k\right|^2}{\|A_{(j_k)}\|_2^2}}{\sum_{j_k=1}^{n} \frac{\left|A_{(j_k)}^T r_k\right|^2}{\|A_{(j_k)}\|_2^2}}, \quad j_k = 1, 2, \ldots, n. \tag{3.6}$$

Next, for analyzing the convergence theory for RCDA method, some lemmas are given as follows.

Lemma 3.1 (Chebyshev's sum inequality, [19])**.** *For two real number sequences* $a_n = \{a_1, a_2, ..., a_n\}$, $b_n = \{b_1, b_2, ..., b_n\}$, *if* $a_1 \geq a_2 \geq \cdots \geq a_n$ *and* $b_1 \geq b_2 \geq \cdots \geq b_n$ *are valid, then the following inequality*

$$\frac{1}{n}\sum_{i=1}^{n} a_i b_i \geq (\frac{1}{n}\sum_{i=1}^{n} a_i)(\frac{1}{n}\sum_{i=1}^{n} b_i)$$

holds true.

Lemma 3.2. [8] *For* $n \in \mathbb{N}^+$, $x_i \geq 0$, $y_i > 0$, $\varrho > 0$, *and* $j = 1, 2, \ldots, n$, *the inequality as follows is valid:*

$$\sum_{j=1}^{n} \frac{x_j^{\varrho+1}}{y_j^{\varrho}} \geq \frac{\left(\sum_{j=1}^{n} x_j\right)^{\varrho+1}}{\left(\sum_{j=1}^{n} y_j\right)^{\varrho}},$$

and the equality holds if and only if $\frac{x_1}{y_1} = \cdots = \frac{x_n}{y_n}$.

Lemma 3.3. [2] *For matrix* $A \in \mathbb{R}^{m\times n}$ *and any vector* $w \in \text{range}(A^T)$, *it follows that*

$$\|Aw\|_2^2 \geq \sigma_{\min}^2(A)\,\|w\|_2^2.$$

For the theory of convergence for RCDA method, the subsequent theorem can be developed.

Theorem 3.1. *For linear system (1.1), where* $A \in \mathbb{R}^{m\times n}(m \geq n)$ *and* $R(A) = n$, *let the initial guess vector* $x_0 \in \mathbb{R}^n$, *then the iteration sequence* $\{x_k\}_{k=0}^{\infty}$ *produced from the RCDA method converges to unique least squares solution* x_{LS} *in expectation. Furthermore, the solution error in expectation of sequence* $\{x_k\}_{k=0}^{\infty}$ *is subject to*

$$\mathbb{E}\|x_1 - x_{LS}\|_{A^TA}^2 \leq \left(1 - \frac{\sigma_{\min}^2(A)}{n\|A\|_F^2}\right)\|x_0 - x_{LS}\|_{A^TA}^2 \tag{3.7}$$

and

$$\mathbb{E}\|(x_{k+1} - x_{LS})\|_{A^TA}^2 \leq \left[1 - \frac{\sigma_{\min}^2(A)}{(n-1)\gamma}\right]\|(x_k - x_{LS})\|_{A^TA}^2, \tag{3.8}$$

where $\gamma = \|A\|_F^2 - \min_{1\leq j\leq n}\|A_j\|_2^2$.

Proof. Based on the step 3 and step 5 of Algorithm 4, for $k = 1, 2, \ldots$, it follows that

$$\begin{aligned} A_{(j_{k-1})}^T r_k &= A_{(j_{k-1})}^T \left(r_{k-1} - \frac{\left\langle A_{(j_{k-1})}, r_{k-1} \right\rangle}{\left\| A_{(j_{k-1})} \right\|_2^2} A_{(j_{k-1})} \right) \\ &= A_{(j_{k-1})}^T r_{k-1} - \frac{\left\langle A_{(j_{k-1})}, r_{k-1} \right\rangle}{\left\| A_{(j_{k-1})} \right\|_2^2} \left\| A_{(j_{k-1})} \right\|_2^2 \\ &= A_{(j_{k-1})}^T r_{k-1} - \left\langle A_{(j_{k-1})}, r_{k-1} \right\rangle \\ &= 0. \end{aligned}$$

Then, from the notion of expectation and the probability criterion equation (3.6), we have

$$\begin{aligned} \mathbb{E}_k \|x_{k+1} - x_k\|_2^2 = & \sum_{\substack{j_k = 1 \\ j_k \neq j_{k-1}}}^{n} \frac{\frac{\left| A_{(j_k)}^T r_k \right|^2}{\left\| A_{(j_k)} \right\|_2^2}}{\sum_{j_k=1}^{n} \frac{\left| A_{(j_k)}^T r_k \right|^2}{\left\| A_{(j_k)} \right\|_2^2}} \cdot \left\| \frac{\left\langle A_{(j_k)}, r_k \right\rangle}{\|A_{(j_k)}\|_2^2} A_{(j_k)} \right\|_2^2 \\ = & \sum_{\substack{j_k = 1 \\ j_k \neq j_{k-1}}}^{n} \frac{\frac{\left| A_{(j_k)}^T r_k \right|^2}{\left\| A_{(j_k)} \right\|_2^2}}{\sum_{j_k=1, j_k \neq j_{k-1}}^{n} \frac{\left| A_{(j_k)}^T r_k \right|^2}{\left\| A_{(j_k)} \right\|_2^2}} \cdot \frac{\left| A_{(j_k)}^T r_k \right|^2}{\|A_{(j_k)}\|_2^2} \\ \geq & \frac{1}{n-1} \cdot \frac{\sum_{j_k=1, j_k \neq j_{k-1}}^{n} \frac{\left| A_{(j_k)}^T r_k \right|^2}{\left\| A_{(j_k)} \right\|_2^2}}{\sum_{j_k=1, j_k \neq j_{k-1}}^{n} \frac{\left| A_{(j_k)}^T r_k \right|^2}{\left\| A_{(j_k)} \right\|_2^2}} \cdot \sum_{\substack{j_k = 1 \\ j_k \neq j_{k-1}}}^{n} \frac{\left| A_{(j_k)}^T r_k \right|^2}{\|A_{(j_k)}\|_2^2} \\ = & \frac{1}{n-1} \cdot \sum_{\substack{j_k = 1 \\ j_k \neq j_{k-1}}}^{n} \frac{\left| A_{(j_k)}^T r_k \right|^2}{\|A_{(j_k)}\|_2^2} \cdot \frac{\left| A_{(j_k)}^T r_k \right|^2}{\left| A_{(j_k)}^T r_k \right|^2} \\ \geq & \frac{1}{n-1} \cdot \frac{\left(\sum_{j_k=1, j_k \neq j_{k-1}}^{n} \left| A_{(j_k)}^T r_k \right|^2 \right)^2}{\sum_{j_k=1, j_k \neq j_{k-1}}^{n} \left\| A_{(j_k)} \right\|_2^2 \cdot \left| A_{(j_k)}^T r_k \right|^2} \\ \geq & \frac{1}{n-1} \cdot \frac{\left(\left\| A^T r_k \right\|_2^2 \right)^2}{\left\| A^T r_k \right\|_2^2 \cdot \sum_{j_k=1, j_k \neq j_{k-1}}^{n} \left\| A_{(j_k)} \right\|_2^2} \\ = & \frac{1}{n-1} \cdot \frac{\left\| A^T r_k \right\|_2^2}{\gamma}, \end{aligned} \tag{3.9}$$

here, the second equality holds true due to $A^T_{(j_{k-1})} r_k = 0$, and since $\{a_n\} = \left\{ \frac{\left|A^T_{(j_k)} r_k\right|^2}{\|A_{(j_k)}\|_2^2} \right\}_{j_k=1}^{n}$ and $\{b_n\} = \left\{ \frac{\left|A^T_{(j_k)} r_k\right|^2}{\|A_{(j_k)}\|_2^2} \right\}_{j_k=1}^{n}$ share the same order, we can re-order them to obtain

$$\frac{\left|A^T_{(\tilde{j_{k_1}})} r_k\right|^2}{\|A_{(\tilde{j_{k_1}})}\|_2^2} \geq \frac{\left|A^T_{(\tilde{j_{k_2}})} r_k\right|^2}{\|A_{(\tilde{j_{k_2}})}\|_2^2} \geq \cdots \geq \frac{\left|A^T_{(\tilde{j_{k_n}})} r_k\right|^2}{\|A_{(\tilde{j_{k_n}})}\|_2^2}$$

and

$$\sum_{j_k=1}^{n} \frac{\left|A^T_{(j_k)} r_k\right|^2}{\|A_{(j_k)}\|_2^2} \cdot \frac{\left|A^T_{(j_k)} r_k\right|^2}{\|A_{(j_k)}\|_2^2} = \sum_{i=1}^{n} \frac{\left|A^T_{(\tilde{j_{k_i}})} r_k\right|^2}{\|A_{(\tilde{j_{k_i}})}\|_2^2} \cdot \frac{\left|A^T_{(\tilde{j_{k_i}})} r_k\right|^2}{\|A_{(\tilde{j_{k_i}})}\|_2^2},$$

then the first inequality can be established by using the Lemma 3.1. Furthermore, we use the Lemma 3.2 for obtaining the second inequality and the third inequality is obviously obtained.

Besides, base on equation (3.9) and by concurrently applying conditional expectation for each side of the equation (3.4) with some calculations, it holds that

$$\begin{aligned}
\mathbb{E}_k \|A(x_{k+1} - x_{LS})\|_2^2 &= \mathbb{E}_k \|A(x_k - x_{LS})\|_2^2 - \mathbb{E}_k \|A(x_{k+1} - x_k)\|_2^2 \\
&= \|A(x_k - x_{LS})\|_2^2 - \mathbb{E}_k \|A(x_{k+1} - x_k)\|_2^2 \\
&\leq \|A(x_k - x_{LS})\|_2^2 - \frac{1}{n-1} \cdot \frac{\left\|A^T r_k\right\|_2^2}{\gamma} \\
&\leq \|A(x_k - x_{LS})\|_2^2 - \frac{\sigma_{\min}^2(A)}{(n-1)\gamma} \|r_k\|_2^2 \\
&= \left[1 - \frac{\sigma_{\min}^2(A)}{(n-1)\gamma}\right] \|A(x_k - x_{LS})\|_2^2, \qquad (3.10)
\end{aligned}$$

where the first and second equality are valid with the use of the linear property for conditional expectation and definition for $\mathbb{E}_k[\cdot]$, respectively, as for the second inequality, the equation $\|Az\|_2^2 \geq \sigma_{\min}^2(A) \|z\|_2^2$ in Lemma 3.3 is used.

Then by applying the full expectation for each side of above equation and using the definition of energy paradigm, it results that

$$\mathbb{E} \|(x_{k+1} - x_{LS})\|_{A^T A}^2 \leq \left[1 - \frac{\sigma_{\min}^2(A)}{(n-1)\gamma}\right] \|(x_k - x_{LS})\|_{A^T A}^2.$$

Analogous to equation (3.9) and (3.10), for $k = 0$, the following inequality can be obtained:

$$\mathbb{E}_k \|A(x_1 - x_{LS})\|_2^2 \geq \frac{\sigma_{\min}^2(A)}{n \|A\|_F^2} \|x_0 - x_{LS}\|_2^2.$$

Finally, we can get equation (3.8), that is,

$$E \|(x_1 - x_{LS})\|_{A^T A}^2 \leq \left(1 - \frac{\sigma_{\min}^2(A)}{n \|A\|_F^2}\right) \|(x_0 - x_{LS})\|_{A^T A}^2.$$

□

4. The RECDA method

Within the current section, the randomized extended coordinate descent method with angle probability (RECDA) is proposed, which also uses the same probability criterion as in Algorithm 4 to select the columns during the iteration process, and we develop a convergence theorem for the new algorithm.

Just as described in Section 2, when the number of rows of matrix $A \in \mathbb{R}^{m\times n}$ is less than the number of columns, i.e. $m < n$, the RGS method fails to converge to corresponding least Euclidean norm solution x_{LN} of linear system (1.1). With the purpose of solving this problem, Ma et al. [12] proposed the RECD method, where in each iteration, for the update of the first and second components, the RECD method constructs the probability criterion used to select rows and columns of matrix A by utilizing Euclidean norm of matrix A, respectively.

However, such criteria only take into account the information related to the norm for matrix A but ignores products such as residual vectors that are generated in each iteration. In addition, the RECD method has the same shortcoming as the RCD method, if the matrix is scaled in advance by using the diagonal matrix, i.e. making Euclidean norm of columns the same constant, then the columns will be chosen with uniform probability in each iteration, which leads to a slower rate of convergence.

Based on the above analysis, we consider using an angle-based probability criterion in the first part of the REGS method without changing the probability criterion in the second part of the REGS method. As shown in the analysis in Algorithm 4, we construct the probability criterion by using the square of the sin value of angle θ_{j_k} between the hyperplane $\mathbb{H}_{j_{k-1}}$ and residual vector r_k. Now, we describe the RECDA method as Algorithm 5.

Algorithm 5 The RECDA Method

Input: $A, b, K, x_0 \in \mathbb{R}^n$ and $z_0 \in \text{range}(A^T)$
Output: z_K

1: **for** $k = 0, 1, \ldots, K-1$ **do**
2: Select $j_k \in \{1, 2, \ldots, n\}$with probability

$$\Pr(\text{column} = j_k) = \frac{\sin^2\theta_{j_k}}{\sum_{j_k=1}^{n}\sin^2\theta_{j_k}}$$

3: Update $x_{k+1} = x_k + \frac{A_{(j_k)}^T(b-Ax_k)}{\|A_{(j_k)}\|_2^2}e_{j_k}$
4: Select $i_k \in \{1, 2, \ldots, m\}$with probability

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

5: Update $z_{k+1} = z_k + \frac{\left(A^{(i_k)}\right)^T(x_{k+1}-z_k)}{\|A^{(i_k)}\|_2^2}\left(A^{(i_k)}\right)^T$
6: **end for**

Remark 4.1. Similarly to the RCDA method, there is no consideration of constructing the probability criterion using the sine values of the angles between hy-

perplanes, since the same as probability criterion based on the Euclidean norm, such a criterion does not take into account the iterative products such as residual vectors.

Next, we first present Lemma 4.1 and then establish convergence theory for RECDA method shown as following Theorem 4.1.

Lemma 4.1. [6] *For matrix $A \in \mathbb{R}^{m\times n}$ and any vector $w \in range(A)$, it follows that*

$$w^T\left(I - \frac{AA^T}{\|A\|_F^2}\right) w \le \left(1 - \frac{\sigma_{\min}^2(A)}{\|A\|_F^2}\right)\|w\|_2^2, \tag{4.1}$$

where the equality holds when $\sigma_{\min}(A) = \sigma_{\max}(A)$.

Theorem 4.1. *For linear system (1.1), where $A \in \mathbb{R}^{m\times n}$ is of full rank, let the initial guess vector $x_0 \in \mathbb{R}^n$ and $z_0 \in \mathrm{range}(A^T)$, then the iteration sequence $\{z_k\}_{k=0}^{\infty}$ produced from the RECDA method converges to the corresponding solution $x_\star$ in expectation. Furthermore, it holds that*

$$\mathbb{E}\|z_{k+1} - x_\star\|_2^2 \le \alpha^{k+1}\|z_0 - x_\star\|_2^2 + \frac{(k+1)\,\beta^k\delta}{\|A\|_F^2}\|x_0 - x_\star\|_{A^TA}^2, \tag{4.2}$$

where $\alpha = 1 - \frac{\sigma_{\min}^2(A)}{\|A\|_F^2}$, $\beta = 1 - \frac{\sigma_{\min}^2(A)}{(n-1)\gamma}$, $\delta = 1 - \frac{\sigma_{\min}^2(A)}{n\|A\|_F^2}$ and $\gamma = \|A\|_F^2 - \min_{1\le j\le n}\|A_j\|_2^2$.

Proof. From the description of the RECDA method, it is clear that

$$\begin{aligned} z_{k+1} - x_\star &= z_k + \frac{\left(A^{(i_k)}\right)^T(x_{k+1} - z_k)}{\left\|A^{(i_k)}\right\|_2^2}\left(A^{(i_k)}\right)^T - x_\star \\ &= \left(I - \frac{\left(A^{(i_k)}\right)^T A^{(i_k)}}{\|A^{(i_k)}\|_2^2}\right) z_k + \frac{\left(A^{(i_k)}\right)^T A^{(i_k)}}{\|A^{(i_k)}\|_2^2} x_{k+1} - x_\star \\ &= \left(I - \frac{\left(A^{(i_k)}\right)^T A^{(i_k)}}{\|A^{(i_k)}\|_2^2}\right)(z_k - x_\star) + \frac{\left(A^{(i_k)}\right)^T A^{(i_k)}}{\|A^{(i_k)}\|_2^2}(x_{k+1} - x_\star). \end{aligned}$$

At the same time, we note that

$$\begin{aligned} &\left\langle \left(I - \frac{\left(A^{(i_k)}\right)^T A^{(i_k)}}{\|A^{(i_k)}\|_2^2}\right)(z_k - x_\star), \frac{\left(A^{(i_k)}\right)^T A^{(i_k)}}{\|A^{(i_k)}\|_2^2}(x_{k+1} - x_\star)\right\rangle \\ &= (x_{k+1} - x_\star)^T \frac{\left(A^{(i_k)}\right)^T A^{(i_k)}}{\|A^{(i_k)}\|_2^2}\left(I - \frac{\left(A^{(i_k)}\right)^T A^{(i_k)}}{\|A^{(i_k)}\|_2^2}\right)(z_k - x_\star) \\ &= (x_{k+1} - x_\star)^T\left(\frac{\left(A^{(i_k)}\right)^T A^{(i_k)}}{\|A^{(i_k)}\|_2^2} - \frac{\left(A^{(i_k)}\right)^T A^{(i_k)}}{\|A^{(i_k)}\|_2^2}\right)(z_k - x_\star) \\ &= 0, \end{aligned}$$

which means that vector $\left(I - \frac{\left(A^{(i_k)}\right)^T A^{(i_k)}}{\|A^{(i_k)}\|_2^2}\right)(z_k - x_\star)$ is orthogonal to $\frac{\left(A^{(i_k)}\right)^T A^{(i_k)}}{\|A^{(i_k)}\|_2^2} \times (x_{k+1} - x_\star)$. Then by using the Pythagorean theorem we can obtain

$$\|z_{k+1} - x_\star\|_2^2$$

$$= \left\| \left(I - \frac{\left(A^{(i_k)}\right)^T A^{(i_k)}}{\|A^{(i_k)}\|_2^2} \right) (z_k - x_\star) \right\|_2^2 + \left\| \frac{\left(A^{(i_k)}\right)^T A^{(i_k)}}{\|A^{(i_k)}\|_2^2} (x_{k+1} - x_\star) \right\|_2^2. \quad (4.3)$$

In addition, it is found, with the use of the definition of expectation and the expansion of the norm, that

$$\begin{aligned}
&\mathbb{E}_k \left\| \frac{\left(A^{(i_k)}\right)^T A^{(i_k)}}{\|A^{(i_k)}\|_2^2} (x_{k+1} - x_\star) \right\|_2^2 \\
&= \mathbb{E}_k \left[(x_{k+1} - x_\star)^T \left(\frac{\left(A^{(i_k)}\right)^T A^{(i_k)}}{\|A^{(i_k)}\|_2^2} \right)^2 (x_{k+1} - x_\star) \right] \\
&= \mathbb{E}_k^j \left[\mathbb{E}_k^i \left[(x_{k+1} - x_\star)^T \frac{\left(A^{(i_k)}\right)^T A^{(i_k)}}{\|A^{(i_k)}\|_2^2} (x_{k+1} - x_\star) \right] \right] \\
&= \mathbb{E}_k^j \left[\mathbb{E}_k^i \left[\frac{\left\|A^{(i_k)} (x_{k+1} - x_\star)\right\|_2^2}{\|A^{(i_k)}\|_2^2} \right] \right] \\
&= \mathbb{E}_k^j \left[\sum_{i_k=1}^m \frac{\|A^{(i_k)}\|_2^2}{\|A\|_F^2} \cdot \frac{\left\|A^{(i_k)} (x_{k+1} - x_\star)\right\|_2^2}{\|A^{(i_k)}\|_2^2} \right] \\
&= \mathbb{E}_k \left[\frac{\|A (x_{k+1} - x_\star)\|_2^2}{\|A\|_F^2} \right] \\
&\le \frac{\beta}{\|A\|_F^2} \|A (x_k - x_\star)\|_2^2 \\
&\vdots \\
&\le \frac{\beta^k \delta}{\|A\|_F^2} \|A (x_0 - x_\star)\|_2^2, \qquad (4.4)
\end{aligned}$$

where the second equality can be obtained with some brief calculations, and by using equation (3.7) and equation (3.8) in Theorem 3.1 and induction, respectively, it is known that the first inequality and the second inequality hold true.

Furthermore, it holds that

$$\begin{aligned}
&\mathbb{E}_k \left\| \left(I - \frac{\left(A^{(i_k)}\right)^T A^{(i_k)}}{\|A^{(i_k)}\|_2^2} \right) (z_k - x_\star) \right\|_2^2 \\
&= \mathbb{E}_k \left[(z_k - x_\star)^T \left(I - \frac{\left(A^{(i_k)}\right)^T A^{(i_k)}}{\|A^{(i_k)}\|_2^2} \right)^2 (z_k - x_\star) \right] \\
&= \mathbb{E}_k \left[(z_k - x_\star)^T \left(I - \frac{\left(A^{(i_k)}\right)^T A^{(i_k)}}{\|A^{(i_k)}\|_2^2} \right) (z_k - x_\star) \right] \\
&= (z_k - x_\star)^T \left(I - \frac{A^T A}{\|A\|_F^2} \right) (z_k - x_\star) \\
&\le \alpha \|z_k - x_\star\|_2^2, \qquad (4.5)
\end{aligned}$$

for the inequality we use the equation (4.1) in Lemma 4.1.

Lastly, by applying the full expectation for each side of equation (4.3) simultaneously and combining equation (4.4) and equation (4.5), we can get equation (4.2), i.e.,

$$
\begin{aligned}
&\mathbb{E}\left\|z_{k+1}-x_\star\right\|_2^2 \\
&=\mathbb{E}\left\|\left(I-\frac{\left(A^{(i_k)}\right)^T A^{(i_k)}}{\|A^{(i_k)}\|_2^2}\right)(z_k-x_\star)\right\|_2^2+\mathbb{E}\left\|\frac{\left(A^{(i_k)}\right)^T A^{(i_k)}}{\|A^{(i_k)}\|_2^2}(x_{k+1}-x_\star)\right\|_2^2 \\
&\leq \alpha\mathbb{E}\left\|z_k-x_\star\right\|_2^2+\frac{\beta^k\delta}{\|A\|_F^2}\left\|A\left(x_0-x_\star\right)\right\|_2^2 \\
&\leq \alpha^2\mathbb{E}\left\|z_{k-1}-x_\star\right\|_2^2+\frac{\alpha\beta^{(k-1)}\delta}{\|A\|_F^2}\left\|A\left(x_0-x_\star\right)\right\|_2^2+\frac{\beta^k\delta}{\|A\|_F^2}\left\|A\left(x_0-x_\star\right)\right\|_2^2 \\
&\leq \alpha^3\mathbb{E}\left\|z_{k-2}-x_\star\right\|_2^2+\left(\frac{\beta^k\delta}{\|A\|_F^2}+\frac{\alpha\beta^{(k-1)}\delta}{\|A\|_F^2}+\frac{\alpha^2\beta^{(k-2)}\delta}{\|A\|_F^2}\right)\left\|A\left(x_0-x_\star\right)\right\|_2^2 \\
&\ \ \vdots \\
&\leq \alpha^{k+1}\left\|z_0-x_\star\right\|_2^2+\sum_{l=0}^{k}\frac{\alpha^l\beta^{k-l}\delta}{\|A\|_F^2}\left\|A\left(x_0-x_\star\right)\right\|_2^2 .
\end{aligned}
$$

□

Remark 4.2. In Theorem 4.1, we do not emphasize the size relationship between the number of columns and rows of matrix A, since the RECDA method is effective for both underdetermined and overdetermined linear systems. In addition, when $m > n$, $x_\star$ represents the least square solution x_{LS} and when $m < n$, $x_\star$ represents the least Euclidean norm solution x_{LN}.

5. Numerical experiments

We design some numerical experiments on methods for solving linear system (1) within this section, i.e. Examples 5.1-5.3, to show the effectiveness of our methods. For RCDA method, the RCD method and the NRCD method are used to compare with it, while for the RECDA method, the RECD method and REK method are used to compare with it. Here the RCD, NRCD, RCDA, RECD, REK, and RECD methods represent the Algorithm 1 in Section 2, the Algorithm 1 in [15], the Algorithm 4 in Section 3, the Algorithm 3 in Section 2, the Algorithm 3 in [26] and the Algorithm 5 in Section 4, respectively. All the experiments performed within this section are conducted using MATLAB R2022a on a computer with 16 GB RAM, 64-bit operating system, windows 10, and AMD Ryzen 7 4800U, 1.80 GHz.

There are two kinds of matrices applied in the experiments, one is the random matrices generated by MATLAB function *randn* randomly, which follows the standard normal distribution, as well as the other one is the real-world matrices selected from sparse matrix collection [5], which have specific application areas, for more details, please refer to Table 10. Meanwhile, for randomly generated matrices, the experiment can be specifically classified into three cases: (i) linear system (1.1) is

Table 1. IT and CPU of RCD, NRCD, RCDA, RECD and RECDA for $m \times n$ matrices A with $n = 300$ and different m when (1.1) is consistent.

m×n		1000×300	2000×300	3000×300	4000×300	5000×300
RCD	IT	7616.0	4905.4	4296.4	4043.4	3735.3
	CPU	0.1608	0.1399	0.1456	0.1467	0.2170
NRCD	IT	2520.9	1378.5	1097.3	974.9	909.7
	CPU	0.0768	0.0492	0.0455	0.0444	0.0610
RCDA	IT	2500.2	1373.6	1106.1	978.7	912.4
	CPU	0.0646	0.0424	0.0384	0.0365	0.0452
	SP_A	2.4892	3.2995	3.7917	4.0192	4.8009
RECD	IT	14650.2	8839.6	7761.4	7341.9	7115.0
	CPU	0.6320	0.4769	0.5014	0.8839	1.1800
RECDA	IT	11400.9	7403.3	6727.2	6422.8	6322.8
	CPU	0.4261	0.3609	0.4049	0.6031	0.8551
	SP_E	1.4832	1.3214	1.2383	1.4656	1.3800

Table 2. IT and CPU of RCD, NRCD, RCDA, RECD and RECDA for $m \times n$ matrices A with $n = 400$ and different m when (1.1) is consistent.

m×n		1000×400	2000×400	3000×400	4000×400	5000×400
RCD	IT	14331.2	7551.5	6313.0	5684.8	5463.8
	CPU	0.2542	0.1551	0.1487	0.1467	0.2308
NRCD	IT	4952.8	2235.1	1704.7	1477.7	1345.3
	CPU	0.1171	0.0596	0.0511	0.0495	0.0640
RCDA	IT	4973.8	2242.3	1714.2	1480.6	1348.7
	CPU	0.1088	0.0573	0.0439	0.0405	0.0519
	SP_A	2.3364	2.7068	3.3872	3.6222	4.4470
RECD	IT	28897.5	14124.0	11518.5	10555.3	9970.9
	CPU	1.4546	0.8919	0.7662	1.3058	1.7865
RECDA	IT	22398.5	11397.1	9639.7	9106.2	8777.4
	CPU	0.8831	0.5829	0.6066	0.8790	1.2199
	SP_E	1.6472	1.5301	1.2631	1.4856	1.4645

overdetermined ($m > n$) and consistent; (ii) linear system (1.1) is overdetermined ($m > n$) and inconsistent; (iii) linear system (1.1) is underdetermined ($m < n$).

In order to construct the consistent linear system, it is taken that $b = Ax_\star$, where $x_\star$ is created from the MATLAB function $randn(n, 1)$, and as for the inconsistent linear system, the scheme $b = Ax_\star + \Psi$ is used, where $\Psi \in \text{null}(A^T)$ can be obtained by the application of the MATLAB function *null*. Besides, the initial vectors for the RCD, NRCD, and RCDA methods are set to $x_0 = 0$ and the iterations of these methods terminate immediately when relative solution error (RSE) fulfills $\text{RSE} \leq 10^{-6}$ or the number of iteration steps is more than 600000, as well as for the

RECD and RECDA methods, the initial vectors are $x_0 = 0$ and $z_0 = 0$, as for the REK method, the initial vectors are $x_0 = 0$ and $z_0 = b$ as well as the iterations of these methods terminate immediately when the error satisfies ERR $\leq 10^{-6}$, where

$$\mathrm{RSE} = \frac{\|x_k - x_\star\|_2^2}{\|x_\star\|_2^2},\ \mathrm{ERR} = \|z_k - x_\star\|_2^2,$$

and for the case that the number of iteration steps is over 600000, it is indicated in the subsequent tables by the label " $--$ ".

Table 3. IT and CPU of RCD, NRCD, RCDA, RECD and RECDA for $m \times n$ matrices A with $n = 500$ and different m when (1.1) is consistent.

m×n		1000×500	2000×500	3000×500	4000×500	5000×500
RCD	IT	26930.3	10758.0	8530.5	7722.4	7333.8
	CPU	0.5229	0.2872	0.2306	0.2135	0.3332
NRCD	IT	9603.4	3461.8	2426.6	2054.2	1833.3
	CPU	0.2833	0.1010	0.0993	0.0936	0.1215
RCDA	IT	9789.0	3381.0	2443.9	2037.2	1841.7
	CPU	0.2217	0.0920	0.0723	0.0643	0.0858
	$\mathrm{SP_A}$	2.3586	3.1217	3.1895	3.3204	3.8834
RECD	IT	55782.3	21209.9	15961.6	14335.8	13360.2
	CPU	2.8411	1.1802	1.1342	1.8000	2.4266
RECDA	IT	43581.6	16753.2	13172.5	12046.8	11557.4
	CPU	1.7984	0.8961	0.8619	1.1840	1.6518
	$\mathrm{SP_E}$	1.5798	1.3170	1.3159	1.5203	1.4691

The performance of various methods is captured through the IT (the number of iteration steps) as well as CPU (computation time in seconds) time, where IT and CPU time represent the mean number of iteration steps and mean computation time for 50 operations of corresponding method, respectively. Further, we define $\mathrm{speed-up_{RCDA}}(\mathrm{SP_A})$ and $\mathrm{speed-up_{RECDA}}(\mathrm{SP_E})$ to demonstrate the advantages of our algorithms more clearly, where $\mathrm{speed-up_{RCDA}}$ represents time speed-up ratio for RCDA method relative to RCD method and $\mathrm{speed-up_{RECDA}}$ represents time speed-up ratio for RECDA method relative to RECD method, that is,

$$\mathrm{speed-up_{RCDA}} = \frac{\text{CPU time of RCD}}{\text{CPU time of RCDA}},$$
$$\mathrm{speed-up_{RECDA}} = \frac{\text{CPU time of RECD}}{\text{CPU time of RECDA}}.$$

Example 5.1. In the present example, we show numerical results for case (i) and case (ii) of randomly generated matrices $A \in \mathbb{R}^{m\times n}(m > n)$. CPU time and the number of IT and for RCD, NRCD, RCDA, RECD and RECDA methods to solve overdetermined linear system (1.1) are listed in Tables 1-6, where Tables 1-3 represent the experimental results of consistent linear system and Tables 4-6 represent the experimental results of inconsistent linear system.

Table 4. IT and CPU of RCD, NRCD, RCDA, RECD and RECDA for $m \times n$ matrices A with $n = 300$ and different m when (1.1) is inconsistent.

m×n		1000×300	2000×300	3000×300	4000×300	5000×300
RCD	IT	7597.0	4917.0	4266.3	4058.9	3801.9
	CPU	0.1525	0.0958	0.0965	0.1018	0.1372
NRCD	IT	2495.9	1368.4	1096.3	975.4	916.2
	CPU	0.0585	0.0365	0.0346	0.0328	0.0446
RCDA	IT	2484.4	1351.1	1103.2	986.8	909.7
	CPU	0.0507	0.0321	0.0286	0.0260	0.0342
	SP_A	3.0079	2.9844	3.3741	3.9154	4.0117
RECD	IT	14770.2	8888.6	7789.3	7401.2	7051.6
	CPU	0.5382	0.4329	0.4698	0.8032	1.1241
RECDA	IT	11512.0	7402.2	6710.4	6460.1	6328.1
	CPU	0.4107	0.3469	0.3912	0.5380	0.7691
	SP_E	1.3104	1.2479	1.2009	1.4929	1.4616

Based on these data, we can clearly observe that with respect to CPU time and the number of IT, the RCDA method performs better than RCD method significantly for both consistent and inconsistent linear systems. When (1.1) is consistent, the speed-up ratio for CPU time of RCDA method relative to RCD method is estimated to achieve minimum of 2.3364 and maximum of 4.8009, as well as in the case where (1.1) is inconsistent, the speed-up ratio is also considerable, at least 2.7090 and maximum up to 4.2425. In addition, the RCDA method still has an advantage in terms of time efficiency with less CPU running time compared to the NRCD method, although the difference with respect to the number of IT for RCDA method and NRCD method is not significant.

Similarly, in comparison with the RECD method, the RCDA method shows significant advantages with regard to CPU time as well as the number of IT, regardless of whether (1.1) is consistent or inconsistent. Furthermore, the CPU time speed-up ratio for RECDA method compared to RECD method is in the range of 1.2383 at least and 1.6472 at most when (1.1) is consistent, and this speed-up ratio is in the range of 1.2009 at least and 1.5947 at most when (1.1) is inconsistent.

The experimental results are shown more clearly in Figures 1 to 6, where Figures 1-3 show the relationship curves of CPU time or the number of IT relative to the number of rows for matrix $A \in \mathbb{R}^{m\times n}$ when (1.1) is consistent, while Figures 4-6 show the corresponding relationship curves when (1.1) is inconsistent. Observing the data shown in the figure, we can notice that with fixed n, the number of IT of these methods shows a decreasing trend with the gradual increase of m. In particular, the most remarkable decrease occurs when m increases from 1000 to 2000; subsequently, the decrease gradually slows down with further increase in m. However, unlike the decreasing trend of the number of IT, the relationship between CPU time and m is characterized by a U-shaped curve. Specifically, as m gradually increases, CPU time firstly decreases and then increases. Besides, it is easy to see that both the NRCD and RCDA methods require least number of IT when iterations are terminated, regardless of the values of m and n. Meanwhile, the RCDA method

Table 5. IT and CPU of RCD, NRCD, RCDA, RECD and RECDA for $m \times n$ matrices A with $n = 400$ and different m when (1.1) is inconsistent.

m×n		1000×400	2000×400	3000×400	4000×400	5000×400
RCD	IT	14289.6	7531.8	6427.6	5666.9	5528.6
	CPU	0.2735	0.1638	0.1593	0.1548	0.2257
NRCD	IT	4992.7	2240.2	1717.6	1467.4	1343.7
	CPU	0.1123	0.0613	0.0538	0.0507	0.0653
RCDA	IT	4928.0	2217.4	1712.1	1477.3	1335.2
	CPU	0.0956	0.0485	0.0424	0.0400	0.0532
	SP_A	2.8609	3.3773	3.7571	3.8700	4.2425
RECD	IT	28902.4	14060.2	11544.6	10562.5	10032.0
	CPU	1.1410	0.7114	0.7224	1.1649	1.5932
RECDA	IT	22352.6	11388.0	9702.6	9075.6	8820.8
	CPU	0.8508	0.5611	0.5943	0.7850	1.0880
	SP_E	1.3411	1.2679	1.2155	1.4839	1.4643

performs optimally with respect to CPU time, followed by the NRCD method.

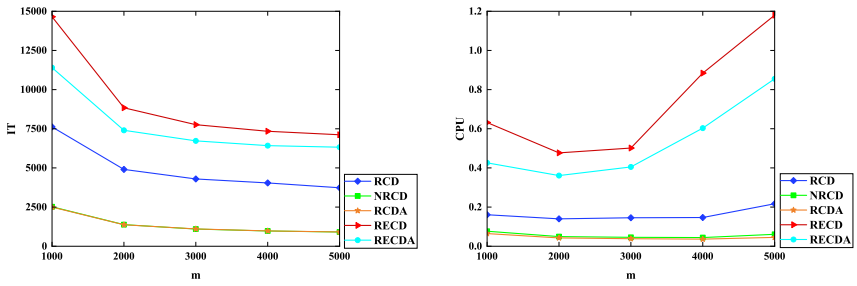

Figure 1. IT and CPU versus m for RCD, NRCD, RCDA, RECD and RECDA when (1.1) is consistent and $n = 300$.

Example 5.2. The present example shows the numerical results for case (iii) of the randomly generated matrices $A \in \mathbb{R}^{m\times n}(m < n)$. Although the RECD method and RECDA method are inferior to the randomized version of coordinate descent without extrapolation in both iteration steps and CPU time for dealing with large scale overdetermined linear systems ($m > n$), they have significant advantages when dealing with underdetermined linear systems ($m < n$). The RCD, NRCD and RCDA methods are not able to converge to least Euclidean norm solution x_{LN} when $m < n$, so Tables 7-9 only list CPU time and the number of IT for RECD, REK and RECDA methods. From these data we again verify the effectiveness of RECDA method, in both CPU time and the number of IT. For random matrices in this example, the RECDA method performed better than RECD and REK methods and has significant advantages, where the CPU time speed-up ratio for RECDA method relative to RECD method ranges from a minimum of 1.2346 to a maximum

Table 6. IT and CPU of RCD, NRCD, RCDA, RECD and RECDA for $m \times n$ matrices A with $n = 500$ and different m when (1.1) is inconsistent.

m×n		1000×500	2000×500	3000×500	4000×500	5000×500
RCD	IT	26521.0	10922.2	8539.9	7656.3	7232.0
	CPU	0.5288	0.2478	0.2214	0.2124	0.3037
NRCD	IT	9807.4	3417.7	2453.7	2054.5	1840.6
	CPU	0.2667	0.1224	0.1022	0.0946	0.1243
RCDA	IT	9597.5	3405.7	2411.1	2057.0	1842.2
	CPU	0.1952	0.0782	0.0627	0.0589	0.0798
	SP_A	2.7090	3.1688	3.5311	3.6061	3.8058
RECD	IT	55648.5	21096.0	15859.6	14208.2	13341.6
	CPU	2.2862	1.1464	1.0897	1.7078	2.2385
RECDA	IT	44001.0	16744.8	13133.5	11979.3	11544.6
	CPU	1.7832	0.8663	0.8362	1.0709	1.4569
	SP_E	1.2821	1.3233	1.3032	1.5947	1.5365

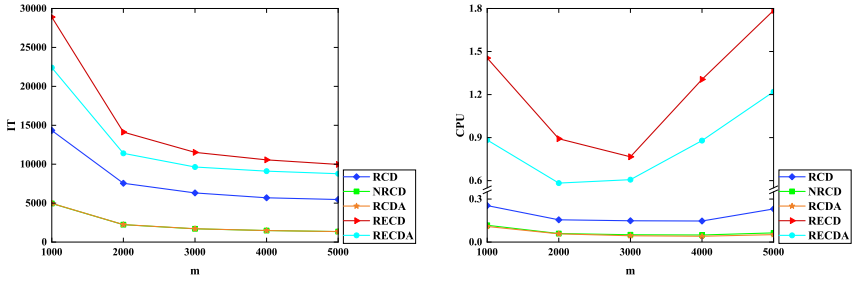

Figure 2. IT and CPU versus m for RCD, NRCD, RCDA, RECD and RECDA when (1.1) is consistent and $n = 400$.

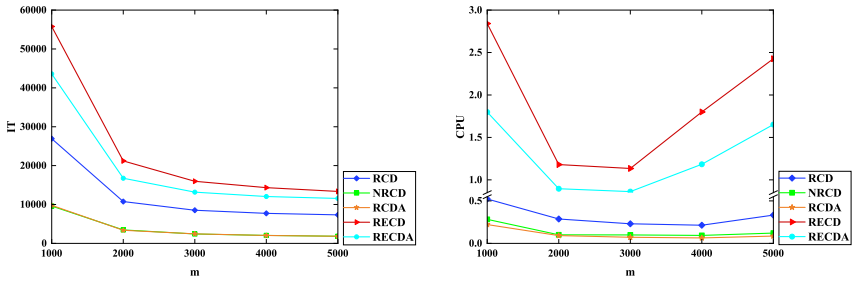

Figure 3. IT and CPU versus m for RCD, NRCD, RCDA, RECD and RECDA when (1.1) is consistent and $n = 500$.

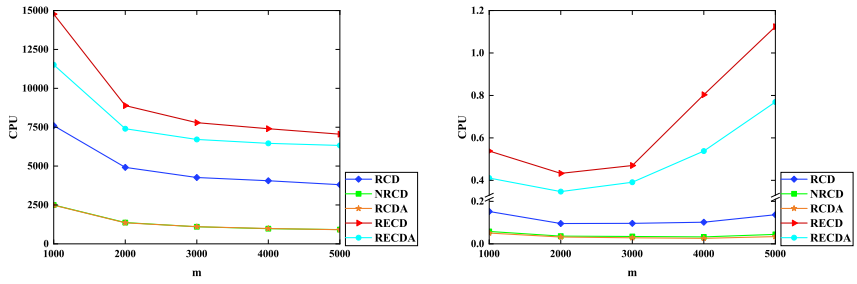

Figure 4. IT and CPU versus m for RCD, NRCD, RCDA, RECD and RECDA when (1.1) is inconsistent and $n = 300$.

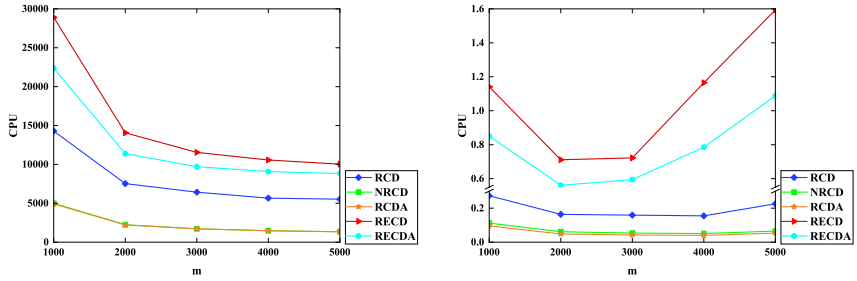

Figure 5. IT and CPU versus m for RCD, NRCD,RCDA, RECD and RECDA when (1.1) is inconsistent and $n = 400$.

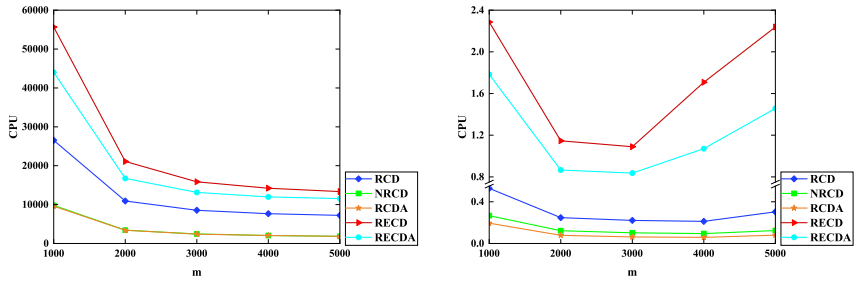

Figure 6. IT and CPU versus m for RCD, NRCD, RCDA, RECD and RECDA when (1.1) is inconsistent and $n = 500$.

of 1.5548.

Example 5.3. For this example, we consider real-world sparse matrices selected from sparse matrix collection [5] with practical applications that are either fat or thin and characteristics of these matrices such as size, density, condition number, rank, and application areas can be found in Table 10, where the density is defined by

$$\text{density} = \frac{\text{Number of nonzero elements of the } m \times n \text{ matrix}}{mn}.$$

From the data in Table 11, it can be found that both the RCD method and

Table 7. IT and CPU of RECD, REK and RECDA for $m \times n$ matrices A with $m = 300$ and different n for Example 5.2.

m×n		300×1000	300×2000	300×3000	300×4000	300×5000
RECD	IT	14597.2	8919.9	7771.6	7286.4	7121.9
	CPU	0.5905	0.5680	0.7361	1.2393	1.6444
REK	IT	14513.0	88112.8	7753.1	7336.5	7061.7
	CPU	0.5445	0.5026	0.6508	1.0878	1.5036
RECDA	IT	11406.3	7099.1	6149.1	5717.8	5492.9
	CPU	0.4783	0.4539	0.5564	0.7995	1.1094
	SP_E	1.2346	1.2514	1.3230	1.5501	1.4822

Table 8. IT and CPU of RECD, REK and RECDA for $m \times n$ matrices A with $m = 400$ and different n for Example 5.2.

m×n		400×1000	400×2000	400×3000	400×4000	400×5000
RECD	IT	28675.3	14025.2	11519.0	10553.8	10068.5
	CPU	1.3152	1.0302	1.1436	1.8145	2.4068
REK	IT	28380.4	14073.5	11474.8	10511.3	10157.0
	CPU	1.2753	0.9611	1.0521	1.7905	2.0582
RECDA	IT	22852.9	11263.5	9326.5	8421.7	8045.8
	CPU	1.0157	0.7600	0.8923	1.2258	1.6557
	SP_E	1.2949	1.3555	1.2816	1.4803	1.4536

the RCDA method have a good performance for dealing with thin matrices, but the RCDA method performs better, where the CPU time speed-up ratio for RCDA method compared to RCD method reaches a maximum value of 4.0335 in the case of matrix ash958. However, while dealing with fat matrices, the RCD and RCDA methods still failed to converge to the corresponding solutions when iteration steps up to 600000, so the data in the table are replaced by " $-$ $-$". For RECD and RECDA methods, they are effective for dealing with both fat and thin matrices, and the RECDA method outperforms the RECD method with respect to both CPU time and the number of IT, where the CPU time speed-up ratio for RECDA method relative to the RECD method reaches a maximum value of 1.5243 for the matrix abtaha1.

6. Conclusion

In this paper, we present two new randomized iterative algorithms to solve large-scale linear systems, that is, the RCDA method and the RECDA method, which are both based on the probability criterion constructed from the sine value of the angle between two successive projection hyperplanes. Our new algorithms not only overcome the shortcomings of the RCD method and RECD method but also fully utilize the various products of iterations. Additionally, we develop convergence

Table 9. IT and CPU of RECD, REK and RECDA for $m \times n$ matrices A with $m = 500$ and different n for Example 5.2.

m×n		500×1000	500×2000	500×3000	500×4000	500×5000
RECD	IT	55488.9	21088.6	16081.0	14250.8	13334.5
	CPU	2.7517	1.6219	1.7242	2.6644	3.5464
REK	IT	55347.0	20906.5	16055.4	14278.2	13284.9
	CPU	2.4438	1.4586	1.5778	2.4590	3.2000
RECDA	IT	43738.2	16831.5	13123.8	11616.3	10807.5
	CPU	2.0178	1.2013	1.3318	1.7387	2.2810
	SP_E	1.3637	1.3501	1.2946	1.5324	1.5548

Table 10. The properties of test sparse matrices in Example 5.3.

Matrix	Size	Density	Cond (A)	Rank	Application field
Cities	55×46	53.04%	207.15	46	Weighted Bipartite Graph
WorldCities	315×100	23.87%	66.00	100	Weighted Bipartite Graph
ash958	958×252	0.68%	3.20	252	Least Squares Problem
abtaha1	14596×209	1.68%	12.23	209	Combinatorial Problem
n3c5-b7	30×120	6.67%	2.23	30	Combinatorial Problem
crew1	135×6469	20.84%	18.20	135	Linear Programming Problem

Table 11. IT and CPU of RECD, REK and RECDA for matrices A for Example 5.3.

m×n		Cities	WorldCities	ash958	abtaha1	n3c5-b7	crew1
RCD	IT	192145.3	21065.3	4624.2	56425.8	−−	−−
	CPU	2.6652	0.3327	0.0843	1.6039	−−	−−
RCDA	IT	61529.7	5339.9	1078.8	15469.3	−−	−−
	CPU	1.0387	0.1030	0.0209	0.4915	−−	−−
	SPA	2.5659	3.2301	4.0335	3.2633	−−	−−
RECD	IT	553703.5	70097.8	11170.5	171823.5	1285.6	29700.5
	CPU	18.3646	2.8592	0.5301	43.6731	0.0460	11.3200
RECDA	IT	406635.6	51157.3	8858.2	130259.8	970	22278.5
	CPU	13.3632	2.1169	0.3897	28.6515	0.0328	8.3130
	SPE	1.3743	1.3507	1.3603	1.5243	1.4024	1.3617

theories for both the RCDA method and the RECDA method as well as perform a series of numerical experiments. The results of experiments have verified the superiority for these two methods with regard to the number of iteration steps and computation time.

Acknowledgements

The authors gratefully acknowledge the anonymous reviewers and editors for their valuable suggestions, which significantly improved the manuscript.

References

[1] Z.-Z. Bai, L. Wang and W.-T. Wu, *On convergence rate of the randomized gauss-seidel method*, Linear Algebra and its Applications, 2021, 611, 237–252.

[2] Z.-Z. Bai and W.-T. Wu, *On greedy randomized kaczmarz method for solving large sparse linear systems*, SIAM Journal on Scientific Computing, 2018, 40(1), A592–A606.

[3] Z.-Z. Bai and W.-T. Wu, *On greedy randomized coordinate descent methods for solving large linear least-squares problems*, Numerical Linear Algebra with Applications, 2019, 26(4), e2237.

[4] P. Breheny and J. Huang, *Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection*, The Annals of Applied Statistics, 2011, 5(1), 232.

[5] T. A. Davis and Y. Hu, *The university of florida sparse matrix collection*, ACM Transactions on Mathematical Software (TOMS), 2011, 38(1), 1–25.

[6] K. Du, *Tight upper bounds for the convergence of the randomized extended kaczmarz and gauss–seidel algorithms*, Numerical Linear Algebra with Applications, 2019, 26(3), e2233.

[7] K. Du and X.-H. Sun, *A doubly stochastic block gauss–seidel algorithm for solving linear equations*, Applied Mathematics and Computation, 2021, 408, 126373.

[8] Y.-J. Guan, W.-G. Li, L.-L. Xing and T.-T. Qiao, *A note on convergence rate of randomized kaczmarz method*, Calcolo, 2020, 57, 1–11.

[9] D. Leventhal and A. S. Lewis, *Randomized methods for linear constraints: Convergence rates and conditioning*, Mathematics of Operations Research, 2010, 35(3), 641–654.

[10] Y. Liu, X.-L. Jiang and C.-Q. Gu, *On maximum residual block and two-step gauss–seidel algorithms for linear least-squares problems*, Calcolo, 2021, 58, 1–32.

[11] Z. Lu and L. Xiao, *On the complexity analysis of randomized block-coordinate descent methods*, Mathematical Programming, 2015, 152, 615–642.

[12] A. Ma, D. Needell and A. Ramdas, *Convergence properties of the randomized extended gauss–seidel and kaczmarz methods*, SIAM Journal on Matrix Analysis and Applications, 2015, 36(4), 1590–1604.

[13] I. Necoara, *A random coordinate descent method for large-scale resource allocation problems*, in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, IEEE, 2012, 4474–4479.

[14] Y. Nesterov, *Efficiency of coordinate descent methods on huge-scale optimization problems*, SIAM Journal on Optimization, 2012, 22(2), 341–362.

[15] Y.-Q. Niu and B. Zheng, *A new randomized gauss–seidel method for solving linear least-squares problems*, Applied Mathematics Letters, 2021, 116, 107057.

[16] Y.-Q. Niu and B. Zheng, *A randomized sparse kaczmarz solver for sparse signal recovery via minimax-concave penalty*, Mathematical Methods in the Applied Sciences, 2024, 47, 6431-6445.

[17] P. Richtárik and M. Takáč, *Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function*, Mathematical Programming, 2014, 144(1), 1–38.

[18] A. Rodomanov and D. Kropotov, *A randomized coordinate descent method with volume sampling*, SIAM Journal on Optimization, 2020, 30(3), 1878–1904.

[19] J. M. Steele, *The Cauchy-Schwarz Master Class: An Introduction to the Art of Mathematical Inequalities*, Cambridge University Press, 2004.

[20] T. Strohmer and R. Vershynin, *A randomized kaczmarz algorithm with exponential convergence*, Journal of Fourier Analysis and Applications, 2009, 15(2), 262–278.

[21] L.-Z. Tan and X.-P. Guo, *On multi-step greedy randomized coordinate descent method for solving large linear least-squares problems*, Computational and Applied Mathematics, 2023, 42(1), 37.

[22] N. Wu and H. Xiang, *On the generally randomized extended gauss-seidel method*, Applied Numerical Mathematics, 2022, 172, 382–392.

[23] Q. Wu, M. Li, J. Zhu, et al., *Dp-rbadabound: A differentially private randomized block-coordinate adaptive gradient algorithm for training deep neural networks*, Expert Systems with Applications, 2023, 211, 118574.

[24] W. Wu, *Convergence of the randomized block gauss-seidel method*, SIAM Undergraduate Research Online, 2018, 11, 369–382.

[25] J. Zhang and J. Guo, *On relaxed greedy randomized coordinate descent methods for solving large linear least-squares problems*, Applied Numerical Mathematics, 2020, 157, 372–384.

[26] A. Zouzias and N. M. Freris, *Randomized extended kaczmarz for solving least squares*, SIAM Journal on Matrix Analysis and Applications, 2013, 34(2), 773–793.

Received December 2024; Accepted February 2025; Available online April 2025.