mathematics

MDPI

Article

Accelerated Randomized Coordinate Descent for Solving Linear Systems

Qin Wang , Weiguo Li *, Wendi Bao and Feiyu Zhang

College of Science, China University of Petroleum, Qingdao 266580, China
* Correspondence: liwg@s.upc.edu.cn

Abstract: The randomized coordinate descent (RCD) method is a simple but powerful approach to solving inconsistent linear systems. In order to accelerate this approach, the Nesterov accelerated randomized coordinate descent method (NARCD) is proposed. The randomized coordinate descent with the momentum method (RCDm) is proposed by Nicolas Loizou, we will provide a new convergence boundary. The global convergence rates of the two methods are established in our paper. In addition, we show that the RCDm method has an accelerated convergence rate by choosing a proper momentum parameter. Finally, in numerical experiments, both the RCDm and the NARCD are faster than the RCD for uniformly distributed data. Moreover, the NARCD has a better acceleration effect than the RCDm and the Nesterov accelerated stochastic gradient descent method. When the linear correlation of matrix A is stronger, the NARCD acceleration is better.

Keywords: Nesterov-accelerated; momentum; Kaczmarz method; large linear system

MSC: 65F10; 65F45

Citation: Wang, Q.; Li, W.; Bao, W.; Zhang, F. Accelerated Randomized Coordinate Descent for Solving Linear Systems. *Mathematics* **2022**, *10*, 4379. https://doi.org/10.3390/math10224379

Academic Editor: Maria Isabel Berenguer

Received: 20 October 2022
Accepted: 12 November 2022
Published: 21 November 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

1. Introduction

Consider a large-scale overdetermined linear system

$$Ax = b, \tag{1}$$

where $A \in R^{m\times n}$, $m \geq n$. We can solve the least-squares problem $min_x\|b - Ax\|^2$. We assume that the columns of A are normalized:

$$\|A_i\| = 1. \tag{2}$$

This assumption has no substantial impact on the implementation costs. We could just normalize each A_i the first time the algorithm encounters it. However, we do not assume (2) about the algorithms, and include factors A_i as needed. Regardless of whether normalization is performed, our randomized algorithms yield the same sequence of iterates.

The coordinate descent (CD) technique [1], which can also be produced by applying the conventional Gauss—Seidel iteration method to the following normal equation [2], is one of the iteration methods that may be used to solve the problem (1) cheaply and effectively.

$$A^TAx = A^Tb,$$

and it is also the same as the quadratic programming problem with no constraints.

$$minf(x) = \frac{1}{2}x^TA^TAx - b^TAx, x \in R^n.$$

From [1], we can obtain

$$x^{k+1} = x^k + \frac{\langle A_i, b - Ax\rangle}{\|A_i\|^2}e_i. \tag{3}$$

In solving problem (1), the coordinate descent approach has a long history of addressing optimization issues and various applications in a wide range of fields such as biological feature selection [3], machine learning [4], protein structure [5], tomography [6,7], and so on. Inspired by the randomized coordinate descent (RCD) method, a lot of related works were presented, such as greedy versions of the randomized coordinate descent [8,9] and block versions of the randomized coordinate descent [10–12]. The coordinate descent method is a column projection method and the Kaczmarz [13] method is a row projection method. The RCD method is inspired by the randomized Kaczmarz(RK) [14] method. For the Kaczmarz-type approach; a lot of relevant work has also been conducted. Readers can refer to [15–20].

In this paper, for solving large systems of linear equations, we use two methods to accelerate the RCD method. First, we obtained an accelerated RCD method by adding Nesterov's acceleration mechanism to the traditional RCD algorithm, called the Nesterov accelerated randomized coordinate descent method (NARCD). It is commonly known that by using an appropriate multi-step technique [21], the traditional gradient method may be turned into a quicker system. To solve the number of unconstrained minimization problems with strongly convex objectives, Nesterov improved this accelerated format [22]. Second, we can apply the heavy ball method (momentum method) to accelerate the RCD. Polyak invented the heavy ball method [23], which is a common approach for speeding up the convergence rate of gradient-type algorithms. Many researchers looked into variations of the heavy ball method, see [24]. By these two methods—to accelerate the RCD—the Nesterov accelerated randomized coordinate descent method (NARCD) and the randomized coordinate descent with momentum method (RCDm) were obtained.

In this paper, given a positive semidefinite matrix M, $\|x\|_M$ is defined as $\sqrt{x^T M x}$, $\langle . \rangle$and $\|.\|$ stands for the scalar product and the spectral norm, where the column vector is denoted by e_i, with 1 at the ith position and 0 elsewhere. In addition, for a given matrix A, A_i, $\|A\|_F$, $\sigma_{min}(A)$, and A^T are used to denote its ith column, Frobenius norm, the smallest nonzero singular value, and the transpose of A respectively. A^+ is the Moore–Penrose pseudoinverse of A. Note that $\lambda_1 = \frac{1}{\|(AA^T)^+\|}$. Let us denote $i(k)$ as the index randomly generated at iteration k, and let $I(k)$ denote all random indices that occurred or before iteration k, so that

$$I(k) = \{i(k), i(k-1), ..., i(0)\},$$

the sequences x_{k+1},y_{k+1},v_{k+1} are determined by $I(k)$. In the following part of the proof, we use $E_{i(k)|I(k-1)}(.)$ to denote the expectation of a random variable condition of $I(k-1)$ with respect to the index $i(k)$. So that

$$E_{I(k)}(.) = E_{I(k-1)}(E_{i(k)|I(k-1)}(.)). \tag{4}$$

The organization of this paper is as follows. In Section 2, we propose the NARCD method naturally and prove the convergence of the method. In Section 3, we propose the RCDm method and prove its convergence. In Section 4, to demonstrate the efficacy of our new methods, several numerical examples are offered. Finally, we present some brief concluding remarks in Section 5.

2. Nesterov's Accelerated Randomized Coordinate Descent

The NARCD algorithm applies the Nesterov accelerated procedure [22], which is more well-known in terms of the gradient descent algorithm. Moreover, the Nesterov acceleration scheme creates the sequences $\{x_k\}$, $\{y_k\}$, and $\{v_k\}$. When applied to $min_x f(x)$, gradient descent sets $x_{k+1} = x_k - \theta_k \nabla f(x)$, where ∇f is the objective gradient and θ_k is the step-size. We define the following iterative scheme:

$$
\begin{aligned}
y_k &= \alpha_k v_k + (1-\alpha_k)x_k, \\
x_{k+1} &= y_k - \theta_k \nabla f(y_k), \\
v_{k+1} &= \beta_k v_k + (1-\beta_k)y_k - \gamma_k \nabla f(y_k).
\end{aligned}
$$

The aforementioned scheme's key addition is that it employs acceptable values for the parameters α_k, β_k, and γ_k, resulting in improved convergence in traditional gradient descent. In [25], the Nesterov-accelerated procedure is applied to the Kaczmarz method, which is a row action method. The RCD is a column action method and the Nesterov-accelerated procedure can be applied in the same way. The relationship between parameters α_k, β_k, and γ_k is given in [22,25]. Now, using the general setup of Nesterov's scheme, we can obtain the NARCD algorithm (Algorithm 1).

The framework of the NARCD method is given as follows.

Algorithm 1 Nesterov's accelerated randomized coordinate descent method (NARCD)

Input: $A \in R^{m\times n}$, $b \in R^m$, $K \in R$, $x^{(0)} \in R^n$, $\lambda \in [0, \lambda_1]$.

1: Initialize $v_0 = x_0$, $\gamma_{-1} = 0$, $k = 0$.
2: **while** $k < K$ **do**
3: Choose γ_k to be the larger root of

$$\gamma_k^2 - \frac{\gamma_k}{n} = (1 - \frac{\gamma_k \lambda}{n})\gamma_{k-1}^2 \tag{5}$$

4: Set α_k and β_k as follows:

$$\alpha_k = \frac{n - \gamma_k \lambda}{\gamma_k (n^2 - \lambda)} \tag{6}$$

$$\beta_k = 1 - \frac{\lambda \gamma_k}{n} \tag{7}$$

5: Set $y_k = \alpha_k v_k + (1-\alpha_k)x_k$
6: Choose $i = i(k)$ from $\{1, 2, ..., n\}$ with equal probability
7: $x_{k+1} = y_k + \frac{\langle A_i, b - Ay_k \rangle}{\|A_i\|^2} e_i$
8: Set $v_{k+1} = \beta_k v_k + (1-\beta_k)y_k + \gamma_k \frac{\langle A_i, b - Ay_k \rangle}{\|A_i\|^2} e_i$
9: $k = k + 1$
10: **end while**

Output: x_K

Remark 1. *In order to avoid the calculation of the product of matrix and vector (Ay_k in steps 7 and 8), we adopt the following*

$$
\begin{aligned}
Y_k &= \alpha_k V_k + (1-\alpha_k)X_k, \\
Z_k &= b - Y_k, \\
\mu_k &= \frac{\langle A_i, Z_k \rangle}{\|A_i\|^2}, \\
X_{k+1} &= Y_k + \mu_k A_i, \\
V_{k+1} &= \beta_k V_k + (1-\beta_k)Y_k + (\gamma_k \mu_k)A_i,
\end{aligned}
$$

and $X_0 = Ax_0$, $V_0 = X_0$. At the same time, we can use $r_k = b - Y_k$ to estimate the residue.

Lemma 1. *For any solution x^* to $A^T A x^* = A^T b$, $y \in R^n$ and $P(y) = y + \frac{\langle A_i, b - Ay \rangle}{\|A_i\|^2} e_i$, $\|A_i\|^2 = 1$. We can obtain*

$$E_i(\|A(P(y)-x^*)\|^2) = \|A(y-x^*)\|^2 - \frac{1}{n}\|A^T(Ay-b)\|^2. \tag{8}$$

where a random variable i satisfies the uniform distribution of the set {1,2,..,n}.

Proof. Using E_i to donate the expectation with respect to the index i, we have

$$\begin{aligned}
&E_i(\|A(P(y)-x^*)\|^2)\\
&= E_i(\|A(y+\langle A_i, b-Ay\rangle e_i - x^*)\|^2)\\
&= E_i(\|(A(y-x^*)+\langle A_i, b-Ay\rangle A_i)\|^2)\\
&= \|A(y-x^*)\|^2 + E_i(\|\langle A_i, b-Ay\rangle A_i\|^2) + 2E_i\langle A(y-x^*), \langle A_i, b-Ay\rangle A_i\rangle\\
&= \|A(y-x^*)\|^2 + \frac{1}{n}\|A^TA(y-x^*)\|^2 + \frac{2}{n}\langle A(y-x^*), AA^T(b-Ay)\rangle\\
&= \|A(y-x^*)\|^2 - \frac{1}{n}\|A^TA(y-x^*)\|^2,
\end{aligned}$$

where the last equality uses $A^TAx^* = A^Tb$. □

Lemma 2. *For any $y \in R^n$, we have*

$$E_i\Big(\|A_i\langle A_i, b-Ay\rangle\|^2_{(AA^T)^+}\Big) \le \frac{1}{n}\|A^TA(y-x^*)\|^2, \tag{9}$$

where the random variable i satisfies the uniform distribution of the set {1,2,..,n}.

Proof. We know the compact singular value decomposition of A as $A = U\Sigma V^T$, where $U \in R^{m\times r}, V \in R^{n\times r}, \Sigma \in R^{r\times r}$, r is the rank of A, and $U^TU = I, V^TV = I$, Σ is the positive diagonal, and we can obtain $(AA^T)^+ = U\Sigma^{-2}U^T$.

$$\begin{aligned}
&E_i\Big(\|A_i\langle A_i, b-Ay\rangle\|^2_{(AA^T)^+}\Big)\\
&= \frac{1}{n}\sum_{i=1}^{n}\langle A_i\langle A_i, b-Ay\rangle, (AA^T)^+A_i\langle A_i, b-Ay\rangle\rangle\\
&= \frac{1}{n}trace[(AA^T)^+\sum_{i=1}^{n}A_i\langle A_i, b-Ay\rangle^2A_i^T]\\
&= \frac{1}{n}trace[(AA^T)^+Adiag(A^T(b-Ay))^2A^T]\\
&= \frac{1}{n}trace[U\Sigma^{-2}U^TU\Sigma V^Tdiag(A^T(b-Ay))^2V\Sigma U^T]\\
&= \frac{1}{n}trace[U\Sigma^{-1}V^Tdiag(A^T(b-Ay))^2V\Sigma U^T]\\
&= \frac{1}{n}trace[V^Tdiag(A^T(b-Ay))^2V]\\
&= \frac{1}{n}\|diag(A^T(b-Ay))V\|_F^2\\
&= \frac{1}{n}\sum_{i=1}^{n}(\langle A_i, b-Ay\rangle)^2\|v_i\|^2\\
&\le \frac{1}{n}\|A^TA(y-x^*)\|^2.
\end{aligned}$$

As the sixth equation is a consequence of trace(ABC)=trace(BCA), and $V = [v_1, v_2, ..., v_n]^T$, $\|v_i\|^2 \le 1$, so that the last inequality holds. □

Lemma 3. *As all $n^2 - \lambda > 0$, the following definition:*

$$\alpha_k = \frac{n - \gamma_k \lambda}{\gamma_k (n^2 - \lambda)}, \beta_k = 1 - \frac{\lambda \gamma_k}{n}$$

for both sequences $\{\alpha_k\}$, $\{\beta_k\}$ lie in the interval $[0, 1]$ if and only if γ_k satisfies the following property:

$$\frac{1}{n} \leq \gamma_k \leq \frac{n}{\lambda},$$

and $\gamma_{-1} = 0$, if $\gamma_{k-1} \leq \frac{1}{\sqrt{\lambda}}$, then $\gamma_k \in [\gamma_{k-1}, \frac{1}{\sqrt{\lambda}}]$.

Proof. The first part of the lemma clearly holds. For the second part, recall from (5) that γ_k is the larger root of the following convex quadratic function:

$$g(\gamma) = \gamma^2 - \frac{\gamma}{n}(1 - \lambda \gamma_{k-1}^2) - \gamma_{k-1}^2,$$

we note the following:

$$g(\gamma_{k-1}) = -(\gamma_{k-1}/n)(1 - \lambda \gamma_{k-1}^2) \leq 0,$$

$$\begin{aligned} g(\frac{1}{\sqrt{\lambda}}) &= \frac{1}{\lambda} - \frac{1}{n\sqrt{\lambda}}(1 - \lambda \gamma_{k-1}^2) - \gamma_{k-1}^2 \\ &= \frac{1}{\lambda} - \frac{1}{n\sqrt{\lambda}} + \gamma_{k-1}^2 (\frac{\sqrt{\lambda}}{m} - 1) \\ &\geq \frac{1}{\lambda} - \frac{1}{n\sqrt{\lambda}} + \frac{1}{\lambda}(\frac{\sqrt{\lambda}}{m} - 1) = 0, \end{aligned}$$

which together imply that $\gamma_k \in [\gamma_{k-1}, \frac{1}{\sqrt{\lambda}}]$. □

Lemma 4. *Let a, b, and c be any vector in R^n, then the following identity holds:*

$$2\langle a - c, c - b \rangle = \|a - b\|^2 - \|a - c\|^2 - \|c - b\|^2. \tag{10}$$

Theorem 1. *The coordinate descent method with Nesterov's acceleration for solving linear equations, $\lambda \in [0, \lambda_1]$ and x^* is the least-squares solution. Define $\sigma_1 = 1 + \frac{\sqrt{\lambda}}{2n}$ and $\sigma_2 = 1 - \frac{\sqrt{\lambda}}{2n}$, then for all $k \geq 0$, we have the following*

$$E(\|A(x_{k+1} - x^*)\|^2) \leq \frac{4\lambda \|A(x_0 - x^*)\|^2}{(\sigma_1^{k+1} - \sigma_2^{k+1})^2} \tag{11}$$

and

$$E(\|A(v_{k+1} - x^*)\|^2_{(AA^T)^+}) \leq \frac{4\|A(x_0 - x^*)\|^2_{(AA^T)^+}}{(\sigma_1^{k+1} + \sigma_2^{k+1})^2}. \tag{12}$$

Proof. We follow the standard notation and steps shown in [22,25]; by (5) and (6), the following relation holds:

$$\frac{1 - \alpha_k}{\alpha_k} = \frac{n\gamma_{k-1}^2}{\gamma_k}. \tag{13}$$

From (5) and (7), we have

$$\gamma_k^2 - \frac{\gamma_k}{n} - \beta_k \gamma_{k-1}^2 = 0. \tag{14}$$

Now, let us define $r_k^2 = \|A(v_k - x^*)\|^2_{(AA^T)^+}$. Then we have

$$
\begin{aligned}
r_{k+1}^2 &= \|A(v_{k+1} - x^*)\|_{(AA^T)^+}^2 \\
&= \|A(\beta_k v_k + (1-\beta_k)y_k + \gamma_k \langle A_i, b - Ay_k\rangle e_i - x^*)\|_{(AA^T)^+}^2 \\
&= \|A(\beta_k v_k + (1-\beta_k)y_k - x^*)\|_{(AA^T)^+}^2 + \gamma_k^2 \|A_i\langle A_i, b - Ay_k\rangle\|_{(AA^T)^+}^2 \\
&+ 2\gamma_k \langle A(\beta_k v_k + (1-\beta_k)y_k - x^*), (AA^T)^+ A_i \langle A_i, b - Ay_k\rangle\rangle \\
&= \|A(\beta_k v_k + (1-\beta_k)y_k - x^*)\|_{(AA^T)^+}^2 + \gamma_k^2 \|A_i\langle A_i, b - Ay_k\rangle\|_{(AA^T)^+}^2 \\
&+ 2\gamma_k \langle A(\beta_k(\frac{1}{\alpha_k} y_k - \frac{1-\alpha_k}{\alpha_k} x_k) + (1-\beta_k)y_k - x^*), (AA^T)^+ A_i \langle A_i, b - Ay_k\rangle\rangle \\
&= \|A(\beta_k v_k + (1-\beta_k)y_k - x^*)\|_{(AA^T)^+}^2 + \gamma_k^2 \|A_i\langle A_i, b - Ay_k\rangle\|_{(AA^T)^+}^2 \\
&+ 2\gamma_k \langle A(y_k - x^*) + \frac{1-\alpha_k}{\alpha_k}\beta_k A(y_k - x_k), (AA^T)^+ A_i \langle A_i, b - Ay_k\rangle\rangle. \qquad (15)
\end{aligned}
$$

Now, we divide (15) into three parts and simplify them separately. From the convexity of $\|.\|_{(AA^T)^+}^2$ and in Lemma 3, we know $\beta_k \in [0,1]$. So the first part of (15) is as follows:

$$
\begin{aligned}
&\|A(\beta_k v_k + (1-\beta_k)y_k - x^*)\|_{(AA^T)^+}^2 \\
&= \|A(\beta_k(v_k - x^*) + (1-\beta_k)(y_k - x^*))\|_{(AA^T)^+}^2 \\
&\leq \beta_k \|A(v_k - x^*)\|_{(AA^T)^+}^2 + (1-\beta_k)\|A(y_k - x^*)\|_{(AA^T)^+}^2 \\
&= \beta_k \|A(v_k - x^*)\|_{(AA^T)^+}^2 + \frac{\gamma_k \lambda}{n}\|A(y_k - x^*)\|_{(AA^T)^+}^2 \\
&\leq \beta_k \|A(v_k - x^*)\|_{(AA^T)^+}^2 + \frac{\gamma_k}{n}\|A(y_k - x^*)\|^2, \qquad (16)
\end{aligned}
$$

where the last inequality makes use of $\lambda \leq \lambda_1 = \frac{1}{\|(AA^T)^+\|}$. Using Lemmas 1 and 2, the second part of (15) is as follows:

$$
\begin{aligned}
&\gamma_k^2 E_{i(k)|I(k-1)}\left(\|A_i\langle A_i, b - Ay_k\rangle\|_{(AA^T)^+}^2\right) \\
&\leq \frac{\gamma_k^2}{n}\|A^T A(y_k - x^*)\|^2 \\
&= \gamma_k^2 \|A(y_k - x^*)\|^2 - \gamma_k^2 E_{i(k)|I(k-1)}(\|A(x_{k+1} - x^*)\|^2). \qquad (17)
\end{aligned}
$$

We use the identity of (8) in the last part of our proof. We take expectations in the last part of (15) and can obtain

$$
\begin{aligned}
&2\gamma_k E_{i(k)|I(k-1)}(\langle A(y_k - x^*) + \frac{1-\alpha_k}{\alpha_k}\beta_k A(y_k - x_k), (AA^T)^+ A_i\langle A_i, b - Ay_k\rangle\rangle) \\
&= 2\gamma_k \langle A(y_k - x^*) + \frac{1-\alpha_k}{\alpha_k}\beta_k A(y_k - x_k), (AA^T)^+ E_{i(k)|I(k-1)} A_i \langle A_i, b - Ay_k\rangle\rangle \\
&= \frac{2\gamma_k}{n}\langle A(y_k - x^*) + \frac{1-\alpha_k}{\alpha_k}\beta_k A(y_k - x_k), (AA^T)^+ \sum_{i=1}^{n} A_i \langle A_i, b - Ay_k\rangle\rangle \\
&= \frac{2\gamma_k}{n}\langle A(y_k - x^*) + \frac{1-\alpha_k}{\alpha_k}\beta_k A(y_k - x_k), (AA^T)^+ AA^T(b - Ay_k)\rangle \\
&= \frac{2\gamma_k}{n}\langle A(y_k - x^*) + \frac{1-\alpha_k}{\alpha_k}\beta_k A(y_k - x_k), (b - Ay_k)\rangle \\
&= \frac{2\gamma_k}{n}\langle A(y_k - x^*), b - Ay_k\rangle + \frac{2\gamma_k}{n}\frac{1-\alpha_k}{\alpha_k}\beta_k \langle A(y_k - x_k), b - Ay_k\rangle \\
&= -\frac{2\gamma_k}{n}\|A(y_k - x^*)\|^2 + \beta_k \gamma_{k-1}^2(\|A(x_k - x^*)\|^2 - \|A(y_k - x^*)\|^2 - \|A(y_k - x_k)\|^2) \\
&\leq -(\frac{2\gamma_k}{n} + \beta_k \gamma_{k-1}^2)\|A(y_k - x^*)\|^2 + \beta_k \gamma_{k-1}^2 \|A(x_k - x^*)\|^2, \qquad (18)
\end{aligned}
$$

where the fifth and sixth equalities make use of the (13), and (10), respectively. Substituting all three parts of (16)–(18) into(15), we have

$$
\begin{aligned}
&E_{i(k)|I(k-1)}(r_{k+1}^2)\\
&\leq \beta_k\|A(v_k-x^*)\|_{(AA^T)^+}^2+\frac{\gamma_k}{n}\|A(y_k-x^*)\|^2\\
&+\gamma_k^2\|A(y_k-x^*)\|^2-\gamma_k^2E_{i(k)|I(k-1)}(\|A(x_{k+1}-x^*)\|^2)\\
&-(\frac{2\gamma_k}{n}+\beta_k\gamma_{k-1}^2)\|A(y_k-x^*)\|^2+\beta_k\gamma_{k-1}^2\|A(x_k-x^*)\|^2\\
&=\beta_k\|A(v_k-x^*)\|_{(AA^T)^+}^2+(\gamma_k^2-\frac{\gamma_k}{n}-\beta_k\gamma_{k-1}^2)\|A(y_k-x^*)\|^2\\
&-\gamma_k^2E_{i(k)|I(k-1)}(\|A(x_{k+1}-x^*)\|^2)+\beta_k\gamma_{k-1}^2\|A(x_k-x^*)\|^2\\
&=\beta_k\|A(v_k-x^*)\|_{(AA^T)^+}^2-\gamma_k^2E_{i(k)|I(k-1)}(\|A(x_{k+1}-x^*)\|^2)+\beta_k\gamma_{k-1}^2\|A(x_k-x^*)\|^2,
\end{aligned}
\tag{19}
$$

where the last equality is the consequence of (14). Let us define two sequences $\{A_k\}$,$\{B_k\}$ as follows:

$$
B_{k+1}^2=\frac{B_k^2}{\beta_k},A_{k+1}^2=\gamma_k^2B_{k+1}^2,
\tag{20}
$$

we know the $\beta_k\in(0,1]$, and $B_k\geq 0$, $B_0\neq 0$. We have $B_{k+1}\geq B_k$. Because of the $\gamma_{-1}=0$, we have $A_0=0$. Moreover, $\gamma_k\in[\gamma_{k-1},\frac{1}{\sqrt{\lambda}}]$ in Lemma 3, so that we can obtain that the $\{A_k\}$ is also an increasing sequence. Now, multiplying both sides of (19) by B_{k+1}^2 and using the (20), we have

$$
\begin{aligned}
&B_{k+1}^2E_{i(k)|I(k-1)}\|A(v_{k+1}-x^*)\|_{(AA^T)^+}^2+A_{k+1}^2E_{i(k)|I(k-1)}\|A(x_{k+1}-x^*)\|^2\\
&\leq B_k^2\|A(v_k-x^*)\|_{(AA^T)^+}^2+A_k^2\|A(x_k-x^*)\|^2,
\end{aligned}
\tag{21}
$$

and then

$$
\begin{aligned}
&E_{I(k)}(B_{k+1}^2\|A(v_{k+1}-x^*)\|_{(AA^T)^+}^2+A_{k+1}^2\|A(x_{k+1}-x^*)\|^2)\\
&=E_{I(k-1)}(B_{k+1}^2E_{i(k)|I(k-1)}\|A(v_{k+1}-x^*)\|_{(AA^T)^+}^2+A_{k+1}^2E_{i(k)|I(k-1)}\|A(x_{k+1}-x^*)\|^2)\\
&\leq E_{I(k-1)}(B_k^2\|A(v_k-x^*)\|_{(AA^T)^+}^2+A_k^2\|A(x_k-x^*)\|^2)\\
&\leq E_{I(0)}(B_1^2\|A(v_1-x^*)\|_{(AA^T)^+}^2+A_1^2\|A(x_1-x^*)\|^2)\\
&\leq B_0^2\|A(v_0-x^*)\|_{(AA^T)^+}^2+A_0^2\|A(x_1-x^*)\|^2)\\
&=B_0^2\|A(v_0-x^*)\|_{(AA^T)^+}^2\\
&=B_0^2\|A(x_0-x^*)\|_{(AA^T)^+}^2.
\end{aligned}
\tag{22}
$$

So, by the (22), we can obtain

$$
\begin{aligned}
E\|A(v_{k+1}-x^*)\|_{(AA^T)^+}^2&\leq\frac{B_0^2}{B_{k+1}^2}\|A(x_0-x^*)\|_{(AA^T)^+}^2,\\
E\|A(x_{k+1}-x^*)\|^2&\leq\frac{B_0^2}{A_{k+1}^2}\|A(x_0-x^*)\|_{(AA^T)^+}^2,
\end{aligned}
\tag{23}
$$

we now need to analyze the growth of two sequences $\{A_k\}$ and $\{B_k\}$. Following the proof in [22,26] for the Nesterov accelerated scheme and the accelerated sampling Kaczmarz Motzkin algorithm [25], we have

$$
B_k^2=\beta_kB_{k+1}^2=(1-\frac{\lambda\gamma_k}{n})B_{k+1}^2=(1-\frac{\lambda A_{k+1}}{nB_{k+1}})B_{k+1}^2.
$$

It implies that

$$\begin{aligned} B_k^2 &= (1 - \frac{\lambda A_{k+1}}{nB_{k+1}})B_{k+1}^2 \\ &= B_{k+1}^2 - \frac{\lambda}{n}A_{k+1}B_{k+1}, \end{aligned}$$

then

$$\frac{\lambda}{n}A_{k+1}B_{k+1} = B_{k+1}^2 - B_k^2 = (B_{k+1} - B_k)(B_{k+1} + B_k) \leq 2B_{k+1}(B_{k+1} - B_k).$$

Moreover, because the $\{B_k\}$ and the $\{A_k\}$ are increasing sequences, we can simplify them and obtain

$$B_{k+1} \geq B_k + \frac{\lambda}{2n}A_{k+1} \geq B_k + \frac{\lambda}{2n}A_k. \tag{24}$$

Similarly, we have

$$\begin{aligned} \frac{A_{k+1}^2}{B_{k+1}^2} - \frac{A_{k+1}}{nB_{k+1}} &= \gamma_k^2 - \frac{\gamma_k}{n} \\ &= \beta_k \gamma_{k-1}^2 \\ &= \frac{A_k^2}{B_{k+1}^2}, \end{aligned}$$

where the second equality uses (14) and the third equality uses (20). Using the above relationship, we have

$$\begin{aligned} \frac{1}{n}A_{k+1}B_{k+1} &= A_{k+1}^2 - A_k^2 \\ &= (A_{k+1} + A_k)(A_{k+1} - A_k) \\ &\leq 2A_{k+1}(A_{k+1} - A_k). \end{aligned}$$

Therefore,

$$A_{k+1} \geq A_k + \frac{B_k}{2n}. \tag{25}$$

By combining the two expressions of (25) and (24), we have

$$\begin{bmatrix} A_{k+1} \\ B_{k+1} \end{bmatrix} \geq \begin{bmatrix} 1 & \frac{1}{2n} \\ \frac{\lambda}{2n} & 1 \end{bmatrix}^{k+1} \begin{bmatrix} A_0 \\ B_0 \end{bmatrix}.$$

The Jordan decomposition of the matrix in the above expression is

$$\begin{bmatrix} 1 & \frac{1}{2n} \\ \frac{\lambda}{2n} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ \sqrt{\lambda} & -\sqrt{\lambda} \end{bmatrix}^{-1} \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ \sqrt{\lambda} & -\sqrt{\lambda} \end{bmatrix}.$$

Here, $\sigma_1 = 1 + \frac{\sqrt{\lambda}}{2n}$ and $\sigma_2 = 1 - \frac{\sqrt{\lambda}}{2n}$. Because of $A_0 = 0$, we have

$$\begin{aligned} \begin{bmatrix} A_{k+1} \\ B_{k+1} \end{bmatrix} &\geq \begin{bmatrix} 1 & \frac{1}{2n} \\ \frac{\lambda}{2n} & 1 \end{bmatrix}^{k+1} \begin{bmatrix} A_0 \\ B_0 \end{bmatrix} \\ &\geq \begin{bmatrix} 1 & 1 \\ \sqrt{\lambda} & -\sqrt{\lambda} \end{bmatrix}^{-1} \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}^{k+1} \begin{bmatrix} 1 & 1 \\ \sqrt{\lambda} & -\sqrt{\lambda} \end{bmatrix} \begin{bmatrix} 0 \\ B_0 \end{bmatrix} \\ &= \frac{1}{2} \begin{bmatrix} (\frac{\sigma_1^{k+1} - \sigma_2^{k+1}}{\sqrt{\lambda}})B_0 \\ (\sigma_1^{k+1} + \sigma_2^{k+1})B_0 \end{bmatrix}. \end{aligned}$$

The above relationship gives us the growth bound for the sequences$\{A_k\}$ and $\{B_k\}$. Substituting these above bounds in (23), we have

$$E\|A(v_{k+1}-x^*)\|^2_{(AA^T)^+} \le \frac{B_0^2}{B_{k+1}^2}\|A(x_0-x^*)\|^2_{(AA^T)^+} \le \frac{4\|A(x_0-x^*)\|^2_{(AA^T)^+}}{(\sigma_1^{k+1}+\sigma_2^{k+1})^2},$$
$$E\|A(x_{k+1}-x^*)\|^2 \le \frac{B_0^2}{A_{k+1}^2}\|A(x_0-x^*)\|^2_{(AA^T)^+} \le \frac{4\lambda\|A(x_0-x^*)\|^2}{(\sigma_1^{k+1}-\sigma_2^{k+1})^2},$$

and we have completed the proof.

Remark 2. *From the relationship between y_k, x_k, and v_k, $y_k = \alpha_k v_k + (1-\alpha_k)x_k$, we know that*

$$\begin{aligned}&E\|A(y_{k+1}-x^*)\|^2_{(AA^T)^+}\\ &= E\|A(\alpha_{k+1}(v_{k+1}-x^*)+(1-\alpha_{k+1})(x_{k+1}-x^*))\|^2_{(AA^T)^+}\\ &\le \alpha_{k+1}E\|A(v_{k+1}-x^*)\|^2_{(AA^T)^+} + (1-\alpha_{k+1})E\|A(x_{k+1}-x^*)\|^2_{(AA^T)^+},\end{aligned}$$

and

$$\begin{aligned}\|A(x_{k+1}-x^*)\|^2_{(AA^T)^+} &\le \|A(x_{k+1}-x^*)\|^2\|(AA^T)^+\|^2\\ &= \frac{\|A(x_{k+1}-x^*)\|^2}{\sigma^4_{min}(A)},\end{aligned}$$

where the $\sigma^4_{min}(A)$ is the nonzero minimum singular value of A. By the above inequality and Theorem 1, we have

$$\begin{aligned}E\|A(y_{k+1}-x^*)\|^2_{(AA^T)^+} &\le \alpha_{k+1}E\|A(v_{k+1}-x^*)\|^2_{(AA^T)^+} + \frac{(1-\alpha_{k+1})}{\sigma^4_{min}(A)}E\|A(x_{k+1}-x^*)\|^2\\ &\le \Big(\frac{4\alpha_{k+1}}{(\sigma_1^{k+1}+\sigma_2^{k+1})^2} + \frac{4(1-\alpha_{k+1})\lambda}{\sigma^4_{min}(AA)(\sigma_1^{k+1}-\sigma_2^{k+1})^2}\Big)\|A(x_0-x^*)\|^2.\end{aligned}$$

□

3. Randomized Coordinate Descent with Momentum Method

The iterative formula of the gradient descent (GD) method is as follows,

$$x_{k+1} = x_k - \lambda_k \nabla f(x_k),$$

where λ_k is a positive step-size parameter. Polyak [23] proposed the gradient descent method with momentum (GDm) by introducing a momentum term $\delta(x_k - x_{k-1})$, which was also known as the heavy ball method

$$x_{k+1} = x_k - \lambda_k \nabla f(x_k) + \delta(x_k - x_{k-1}),$$

where δ is a momentum parameter. Letting $g(x_k)$ be an unbiased estimator of the true gradient $\nabla f(x_k)$, we have the stochastic gradient descent with momentum (mSGD) method.

$$x_{k+1} = x_k - \lambda_k g(x_k) + \delta(x_k - x_{k-1}).$$

The randomized coordinate descent with momentum (RCDm) method is proposed in [27]. We will give a new convergence boundary.

The RCDm method takes the explicit iterative form

$$x_{k+1} = x_k + \frac{\langle A_i, b - Ax_k\rangle}{\|A_i\|^2}e_i + \delta(x_k - x_{k-1}). \tag{26}$$

The framework of the RCDm method is given as follows (Algorithm 2).

Algorithm 2 Randomized coordinate descent with momentum method (RCDm)

Input: $A \in R^{m\times n}$, $b \in R^m$, $K \in R$, $x^{(0)} \in R^n$, δ.
1: Initialize $k = 0$.
2: **while** $k < K$ **do**
3: Choose $i = i(k)$ from $\{1, 2, ..., n\}$ with equal probability
4: $x_{k+1} = x_k + \frac{\langle A_i, b - Ax_k\rangle}{\|A_i\|^2} e_i + \delta(x_k - x_{k-1})$
5: $k = k + 1$
6: **end while**
Output: x_K

Remark 3. *In order to avoid the calculation of the product of matrix and vector (Ax_k in step 5), we adopt the following method*

$$\alpha_k = \frac{\langle A_i, r_k\rangle}{\|A_i\|^2},$$
$$r_{k+1} = (1 + \delta)r_k - \alpha_k A_i - \delta r_{k-1},$$

and $r_0 = b - Ax_0$, $r_{-1} = r_0$.

Lemma 5 ([27]). *Fix $F_1 = F_0 \geq 0$ and let $\{F_k\}_{k\geq 0}$ be a sequence of nonnegative real numbers satisfying the relation*

$$F_{k+1} \leq a_1 F_k + a_2 F_{k-1}, \forall k \geq 1,$$

where $a_2 \geq 0$, $a_1 + a_2 < 1$, and at least one of the coefficients a_1, a_2 is positive. Then the sequence satisfies the relation $F_{k+1} \leq q^k(1 + \xi)F_0$ for all $k \geq 1$, where $q = \frac{a_1+\sqrt{a_1^2+4a_2}}{2}$ and $\xi = q - a_1 \geq 0$. Moreover,

$$q \geq a_1 + a_2,$$

with equality if and only if $a_2 = 0$ (in that case, $q = a_1$ and $\xi = 0$).

Theorem 2. *Assume $\delta \geq 0$, and that the expressions $a_1 = 1 - \frac{\sigma_{min}^2(A)}{n} + 3\delta - 3\frac{\delta\sigma_{min}^2(A)}{n} + 2\delta^2$ and $a_2 = 2\delta^2 + \delta - \frac{\delta\sigma_{min}^2(A)}{n}$ satisfy $a_1 + a_2 < 1$, where $\sigma_{min}^2(A)$ is nonzero minimum singular value of A. Let $\{x_k\}_{k=0}^{\infty}$ be the iteration sequence generated by the RCDm method starting from initial guess $x_0 = 0$. Then, it holds that*

$$E(\|A(x_{k+1} - x^*)\|^2) \leq q^k(1 + \xi)\|A(x_0 - x^*)\|^2, \tag{27}$$

where $q = \frac{a_1+\sqrt{a_1^2+4a_2}}{2}$, $\xi = q - a_1 \geq 0$, x^* is the least-squares solution. Moreover, a_1, a_2, q obeys $a_1 + a_2 \leq q < 1$.

Proof. From the algorithm of RCDm, we have

$$\begin{aligned}
&E_{i(k))|I(k-1)}\|A(x_{k+1} - x^*)\|^2 \\
&= E_{i(k)|I(k-1)}\|A(x_k + \langle A_i, b - Ax_k\rangle e_i + \delta(x_k - x_{k-1}) - x^*)\|^2 \\
&= E_{i(k)|I(k-1)}\|A(x_k - x^*) + A_i\langle A_i, b - Ax_k\rangle\|^2 + \delta^2\|A(x_k - x_{k-1})\|^2 \\
&+ 2\delta E_{i(k)|I(k-1)}\langle A(x_k - x_{k-1}), A(x_k - x^*) + A_i\langle A_i, b - Ax_k\rangle\rangle.
\end{aligned} \tag{28}$$

We consider the three terms in (28) in turn. For the first term, we have

$$\begin{aligned}
&E_{i(k)|I(k-1)}\|A(x_k-x^*)+A_i\langle A_i,b-Ax_k\rangle\|^2\\
&=\|A(x_k-x^*)\|^2+E_{i(k)|I(k-1)}\|A_i\langle A_i,b-Ax_k\rangle\|^2\\
&+2E_{i(k)|I(k-1)}\langle A(x_k-x^*),A_i\langle A_i,b-Ax_k\rangle\rangle\\
&=\|A(x_k-x^*)\|^2+\frac{1}{n}\sum_{i=1}^{n}\|A_i\langle A_i,b-Ax_k\rangle\|^2\\
&+\frac{2}{n}\sum_{i=1}^{n}\langle A(x_k-x^*),A_i\langle A_i,b-Ax_k\rangle\rangle\\
&=\|A(x_k-x^*)\|^2-\frac{1}{n}\|A^TA(x_k-x^*)\|^2\\
&\le\|A(x_k-x^*)\|^2-\frac{\sigma_{min}^2(A)}{n}\|A(x_k-x^*)\|^2\\
&=(1-\frac{\sigma_{min}^2(A)}{n})\|A(x_k-x^*)\|^2,
\end{aligned}\tag{29}$$

where the last inequality is the consequence of singular value inequality ($\|Ax\|^2 \ge \sigma_{min}^2(A)\|x\|^2$), and $n=\|A\|_F^2 \ge \sigma_{min}^2(A)$. From the second term, we have

$$\begin{aligned}
&\delta^2\|A(x_k-x_{k-1})\|^2\\
&=\delta^2\|A(x_k-x^*)-A(x_{k-1}-x^*)\|^2\\
&\le 2\delta^2(\|A(x_k-x^*)\|^2+\|A(x_{k-1}-x^*)\|^2).
\end{aligned}\tag{30}$$

From the third term, we have

$$\begin{aligned}
&2\delta E_{i(k)|I(k-1)}\langle A(x_k-x_{k-1}),A(x_k-x^*)+A_i\langle A_i,b-Ax_k\rangle\rangle\\
&=2\delta\langle A(x_k-x_{k-1}),A(x_k-x^*)\rangle+2\delta E_{i(k)|I(k-1)}\langle A(x_k-x_{k-1}),A_i\langle A_i,b-Ax_k\rangle\rangle\\
&=\delta(\|A(x_k-x_{k-1})\|^2+\|A(x_k-x^*)\|^2-\|A(x_{k-1}-x^*)\|^2)\\
&+\frac{2\delta}{n}\langle A(x_k-x_{k-1}),\sum_{i=1}^{n}A_i\langle A_i,b-Ax_k\rangle\rangle\\
&=\delta(\|A(x_k-x_{k-1})\|^2+\|A(x_k-x^*)\|^2-\|A(x_{k-1}-x^*)\|^2)\\
&+\frac{2\delta}{n}\langle A(x_k-x_{k-1}),AA^T(b-Ax_k)\rangle\\
&=\delta(\|A(x_k-x_{k-1})\|^2+\|A(x_k-x^*)\|^2-\|A(x_{k-1}-x^*)\|^2)\\
&-\frac{2\delta}{n}\langle A^TA(x_k-x_{k-1}),A^TA(x_k-x^*)\rangle\\
&=\delta(\|A(x_k-x_{k-1})\|^2+\|A(x_k-x^*)\|^2-\|A(x_{k-1}-x^*)\|^2)\\
&-\frac{\delta}{n}(\|A^TA(x_k-x_{k-1})\|^2+\|A^TA(x_k-x^*)\|^2-\|A^TA(x_{k-1}-x^*)\|^2)\\
&\le\delta(\|A(x_k-x_{k-1})\|^2+\|A(x_k-x^*)\|^2-\|A(x_{k-1}-x^*)\|^2)\\
&-\frac{\delta\sigma_{min}^2(A)}{n}(\|A(x_k-x_{k-1})\|^2+\|A(x_k-x^*)\|^2-\|A(x_{k-1}-x^*)\|^2)\\
&=(\delta-\frac{\delta\sigma_{min}^2(A)}{n})(\|A(x_k-x_{k-1})\|^2+\|A(x_k-x^*)\|^2-\|A(x_{k-1}-x^*)\|^2)\\
&\le(\delta-\frac{\delta\sigma_{min}^2(A)}{n})(3\|A(x_k-x^*)\|^2+\|A(x_{k-1}-x^*)\|^2).
\end{aligned}\tag{31}$$

where the second equality uses (10), the first inequality uses a singular value inequality

and (10), and the last inequality is a consequence of $\delta\|A(x_k - x_{k-1})\|^2 \leq 2\delta\|A(x_k - x^*)\|^2 + 2\delta\|A(x_{k-1} - x^*)\|^2$. Using the (29)–(31), we obtain

$$\begin{aligned} &E_{i(k)|I(k-1)}\|A(x_{k+1} - x^*)\|^2 \\ &\leq (1 - \frac{\sigma_{min}^2(A)}{n})\|A(x_k - x^*)\|^2 + 2\delta^2(\|A(x_k - x^*)\|^2 + \|A(x_{k-1} - x^*)\|^2) \\ &+ (\delta - \frac{\delta\sigma_{min}^2(A)}{n})(3\|A(x_k - x^*)\|^2 + \|A(x_{k-1} - x^*)\|^2). \end{aligned} \tag{32}$$

Moreover, then

$$\begin{aligned} &E\|A(x_{k+1} - x^*)\|^2 \\ &\leq (1 - \frac{\sigma_{min}^2(A)}{n} + 3\delta - 3\frac{\delta\sigma_{min}^2(A)}{n} + 2\delta^2)E\|A(x_k - x^*)\|^2 \\ &+ (2\delta^2 + \delta - \frac{\delta\sigma_{min}^2(A)}{n})E\|A(x_{k-1} - x^*)\|^2. \end{aligned} \tag{33}$$

By Lemma 5, let $F_k = E\|A(x_k - x^*)\|^2$, we have the following relation

$$F_{k+1} \leq a_1 F_k + a_2 F_{k-1},$$

and we have

$$E\|A(x_k - x^*)\|^2 \leq q^{k-1}(1 + \xi)\|x_0 - x^*\|^2, \tag{34}$$

where $a_1 = 1 - \frac{\sigma_{min}^2(A)}{n} + 3\delta - 3\frac{\delta\sigma_{min}^2(A)}{n} + 2\delta^2$, $a_2 = 2\delta^2 + \delta - \frac{\delta\sigma_{min}^2(A)}{n}$, $\xi = q - a_1 \geq 0$, and we have completed the proof. □

Remark 4. *$a_1 = 1 - \frac{\sigma_{min}^2(A)}{n} + 3\delta - 3\frac{\delta\sigma_{min}^2(A)}{n} + 2\delta^2$, $a_2 = 2\delta^2 + \delta - \frac{\delta\sigma_{min}^2(A)}{n}$. When $\delta = 0$, we can obtain $a_1 = 1 - \frac{\sigma_{min}^2(A)}{n}$, $a_2 = 0$ satisfy $a_1 + a_2 < 1$. When δ takes a small value, we can obtain that this relation $a_1 + a_2$ is satisfied. In addition, the RCDm method degenerates to the RCD method when $\delta = 0$. The RCDm method converges faster than the RCD method if we choose a proper δ. Numerical experiments will show the effectiveness of the RCDm method.*

When $\delta \geq 0$, we can conclude that $a_2 \geq 0$. For the above theorem, we have to satisfy $a_1 + a_2 < 1$, so $a_1 + a_2 = 4\delta^2 + 4(1 - \frac{\sigma_{min}^2(A)}{n})\delta + 1 - \frac{\sigma_{min}^2(A)}{n}$. We set $\omega = \frac{\sigma_{min}^2(A)}{n}$. It can be concluded that $\delta \in [0, \frac{\omega - 1 + \sqrt{(1-\omega)^2 + \omega}}{2}]$. When $\delta \in [0, \frac{\omega - 1 + \sqrt{(1-\omega)^2 + \omega}}{2}]$, the RCDm method converges. However, in the later experiments, the choice of δ will exceed this range because it took a lot of scaling to reach this range.

4. Numerical Experiments

In this section, we compare the influence of different δ on the RCDm algorithm and the effectiveness of the RCD, RCDm, and NARCD methods for solving the large linear system $Ax = b$. All experiments were performed in MATLAB [28] (version R2018a), on a personal laptop with a 1.60 GHz central processing unit (Intel(R) Core(TM) i5-10210U CPU), 8.00 GB memory, and a Windows operating system (64 bits, Windows 10).

In all implementations, the starting point was chosen to be x0 = zeros(n, 1), the right vector $b + \epsilon = Ax^* + \epsilon$ where $\epsilon \in N(A^T)$ and $x^* = ones(n, 1)$. The relative residual error (RRE) at the kth iteration is defined as follows:

$$RRE = \frac{\|b - Ax_k\|^2}{\|b\|^2}.$$

The iterations are terminated once the relative solution error satisfies $RRE < 10^{-8}$ or the number of iteration steps exceeds 5,000,000. If the number of iteration steps exceeds

5,000,000, it is denoted as "-". IT and CPU denote the number of iteration steps and the CPU times (in seconds) respectively. In addition, the CPU and IT mean the arithmetical averages of the elapsed running times and the required iteration steps with respect to 50 trials of repeated runs of the corresponding method. The speed-up of the RCD method against the RCDm method is defined as follows:

$$speed - up_1 = \frac{CPUofRCD}{CPUofRCDm}$$

and the speed-up of the RCD method against the NARCD method is defined as follows:

$$speed - up_2 = \frac{CPUofRCD}{CPUofNARCD}.$$

4.1. Experiments for Different δ on the RCDm

The matrix A is randomly generated by using the MATLAB function unifrnd (0,1,m,n). We observe that RCDm, with appropriately chosen momentum parameters $0 < \delta \leq 0.4$, always converges faster than their no-momentum variants. In this subsection, we let $\delta = 0, 0.1, 0.2, 0.3, 0.4$ to compare their performances. Numerical results are reported in Tables 1–3 and Figure 1. We can conclude some observations as follows. when $\delta = 0.1, 0.2, 0.3, 0.4$, the acceleration effect is good.

Table 1. For different δ, IT, and CPU of RCDm for matrices $A \in R^{m\times n}$ with m = 8000 and different n.

δ	IT				CPU			
	$m \times 3000$	$m \times 2000$	$m \times 1000$	$m \times 800$	$m \times 3000$	$m \times 2000$	$m \times 1000$	$m \times 800$
0	416,319	209,592	59,636	43,107	21.2625	9.9795	2.3001	1.9263
0.1	338,108	155,672	52,806	38,483	18.2890	7.8586	2.0735	1.5128
0.2	352,018	146,365	46,744	37,429	19.4076	7.3941	1.7775	1.4354
0.3	326,510	135,157	47,963	32,412	17.4489	6.8153	2.2577	1.2290
0.4	279,492	123,871	41,812	31,270	15.2037	6.2775	2.0089	1.2017

Table 2. For different δ, IT, and CPU of RCDm for matrices $A \in R^{m\times n}$ with m = 800 and different n.

δ	IT				CPU			
	800×300	800×200	800×100	800×50	800×300	800×200	800×100	800×50
0	43,760	18,200	5615	2449	0.2736	0.1239	0.0442	0.0149
0.1	39,442	16,200	5696	2188	0.2413	0.0929	0.0323	0.0131
0.2	30,900	15,724	5286	2427	0.2091	0.0861	0.0303	0.0150
0.3	28,501	14,047	4645	1858	0.2408	0.0791	0.0264	0.0118
0.4	25,918	12,211	4185	1735	0.1487	0.1374	0.0244	0.0114

Table 3. For different δ, IT, and CPU of RCDm for matrices $A \in R^{m\times n}$ with m = 300 and different n.

δ	IT				CPU			
	300×200	300×150	300×100	300×50	300×200	300×150	300×100	300×50
0	90,517	32,685	13,310	3858	0.4722	0.1976	0.0641	0.0169
0.1	67,610	30,382	13,022	3121	0.4669	0.1719	0.0566	0.0137
0.2	77,748	31,879	11,382	2654	0.4529	0.1657	0.0492	0.0126
0.3	78,023	25,130	9888	2490	0.4528	0.1540	0.0489	0.0147
0.4	66,663	18,566	7965	2344	0.4037	0.1046	0.0411	0.0115

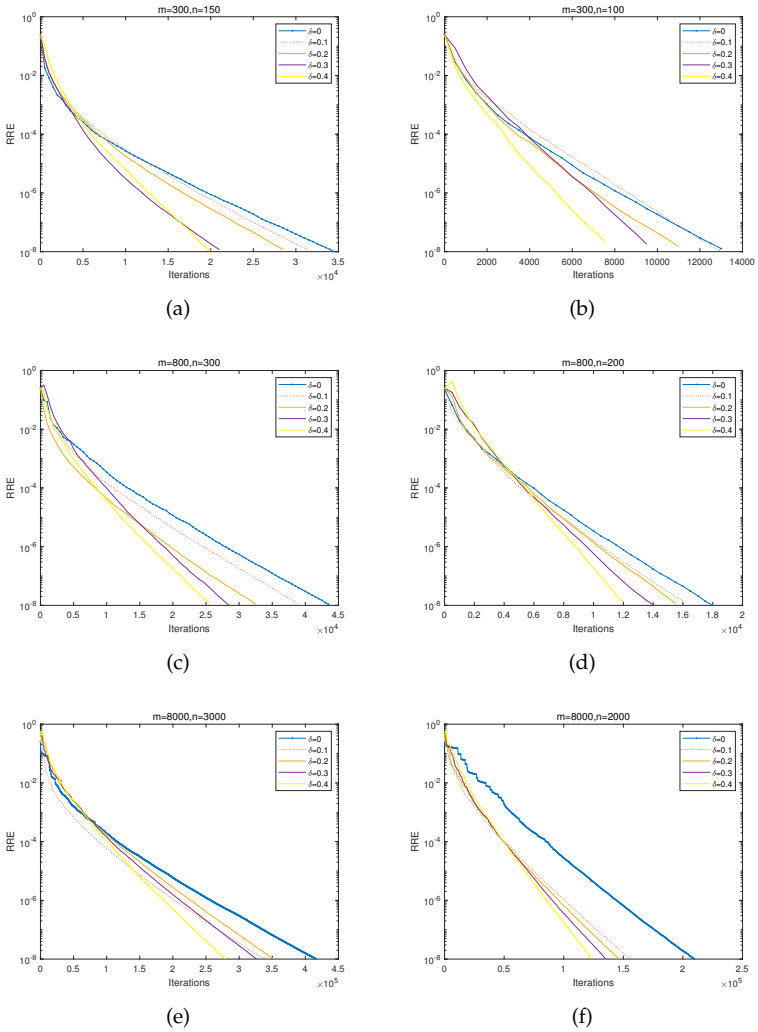


Figure 1. (**a**,**b**): m = 300 rows and n = 150, 100 columns for different δ. (**c**,**d**): m = 800 rows and n = 300, 200 columns for different δ. (**e**,**f**): m = 8000 rows and n = 3000, 2000 columns for different δ.

4.2. Experiments for NARCD, RCDm, RCD, NASGD

Matrix A is randomly generated by using the MATLAB function unifrnd (0,1,m,n). For the RCDm method, let us take the momentum parameter $\delta = 0.3$. For the NARCD method, let us take the Nesterov accelerated parameter $\lambda = 0.05$. For the Nesterov accelerated stochastic gradient descent method (NASGD), it is the step size $\alpha = 0.01$. We observe the performances of RCD, RCDm, and NARCD methods with matrices A of different sizes. From Figure 2 and Tables 4–7, we found that both the NARCD and the RCDm with appropriate momentum parameters can accelerate the RCD; the NARCD and the RCDm always converge faster than the RCD. Moreover, we found that the NARCD has a better acceleration effect than the RCDm. From Table 7, for matrix A$\in R^{8000\times3000}$, the NARCD method demonstrates the best numerical results than the other matrices in terms of the value of the speed-up, where the speed-up is 3.0206. From Figure 3, we found that the acceleration of NARCD method and RCDm method experience gentle changes as the matrix becomes larger, so we can see that these two methods still have good speed-ups when the matrix is very large. From Figure 4, we found that the NARCD converges faster than the NASGD.

Table 4. IT and CPU of RCD, RCDm, NARCD for matrices $A \in R^{m\times n}$ with m = 4000 and different n.

	IT		**CPU**	
	4000 × 800	**4000 × 1000**	**4000 × 800**	**4000 × 1000**
RCD	57,723	81,926	1.3939	1.9264
RCDm	44,962	66,425	1.1068	1.5824
NARCD	20,075	24,184	0.8657	0.9711

Table 5. IT and CPU of RCD, RCDm, NARCD for matrices $A \in R^{m\times n}$ with m = 8000 and different n.

	IT		**CPU**	
	8000 × 2000	**8000 × 3000**	**8000 × 2000**	**8000 × 3000**
RCD	194,046	414,465	9.4357	24.2728
RCDm	144,592	312,665	8.0861	19.1512
NARCD	45,700	71,216	4.5801	8.0357

Table 6. IT and CPU of RCD, RCDm, NARCD for matrices $A \in R^{m\times n}$ with m = 12,000 and different n.

	IT		**CPU**	
	12,000 ×2000	**12,000 ×4000**	**12,000 ×2000**	**12,000 ×4000**
RCD	146,007	465,484	10.8746	31.4180
RCDm	110,339	340,633	8.5894	23.4675
NARCD	43,110	89,046	5.7312	11.0535

Table 7. The speed-up of the RCD method against the NARCD and RCDm.

	4000 × 1000	**8000 × 3000**	**12,000 ×4000**
$speed - up_1$	1.2173	1.2674	1.3387
$speed - up_2$	1.9837	3.0206	2.8423

4.3. Experiment with Different Correlations of Matrix A

Matrix A is randomly generated by using the MATLAB function unifrnd (c,1,m,n), $c \in [0, 1)$. We let $c = 0, 0.2, 0.4, 0.6$. For the RCDm method, let us take the momentum parameter $\delta = 0.3$. For the NARCD method, let us take the Nesterov accelerated parameter $\lambda = 0.05$. As the value of c increases, the correlation of matrix A becomes stronger. From Tables 8–12, we know that as c increases, the condition number for the matrix increases. The larger the condition number, the more ill-conditioned the matrix. The more ill-conditioned the matrix, the more time it takes to solve. From Tables 10 and 12, we know that the acceleration effect of RCDm does not change much with the increase of the c value, but the acceleration effect of NARCD is becoming better.

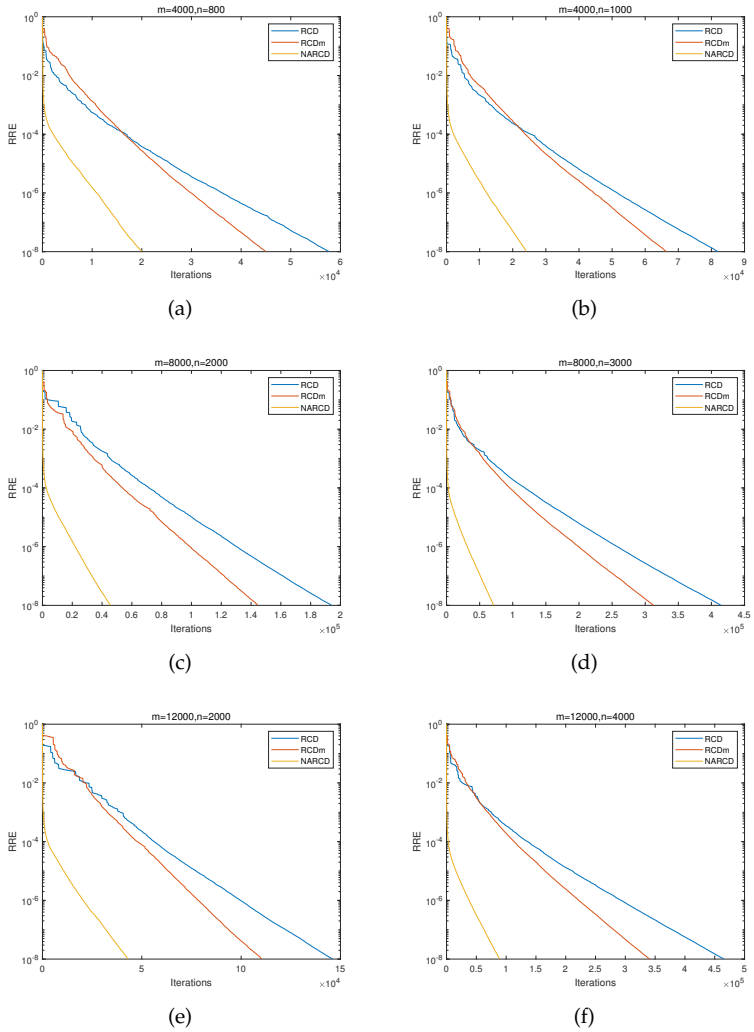


Figure 2. (**a**,**b**): m = 4000 rows and n = 800, 1000 columns for RCD, RCDm, and NARCD. (**c**,**d**): m = 8000 rows and n = 2000, 3000 columns for RCD, RCDm, and NARCD. (**e**,**f**): m = 12,000 rows and n = 2000, 4000 columns for RCD, RCDm, and NARCD.

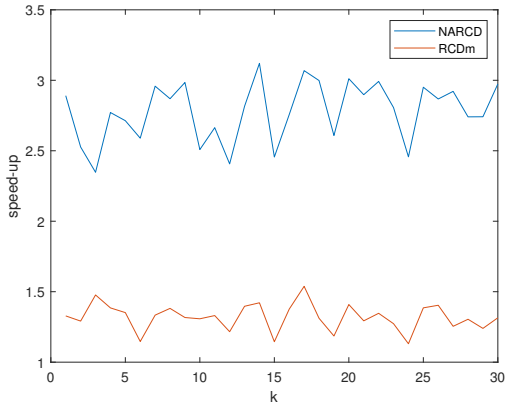


Figure 3. The speed-up of the RCD method against the NARCD and RCDm for matrices $A \in R^{m\times n}$ with $m = 300 \times k$ and $n = 100 \times k$.

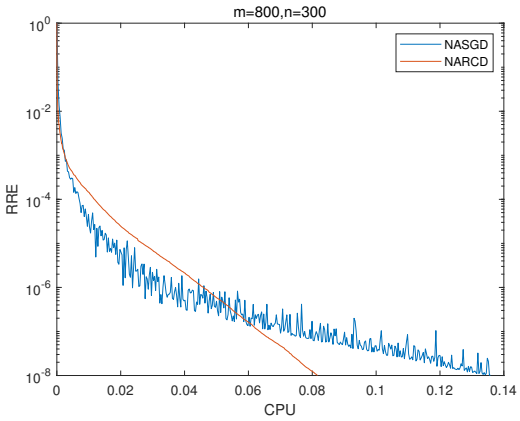


Figure 4. m = 800 and n = 300 for NARCD and NASGD.

Table 8. The condition number of different matrices.

	0	**0.2**	**0.4**	**0.9**
$cond(A_{800\times300})$	75.6431	113.8689	172.7404	1425.0834
$cond(A_{1000\times800})$	452.9536	673.9111	1104.0601	8969.5561

Table 9. IT and CPU of RCD, RCDm, NARCD for matrices $A \in R^{m\times n}$ m = 800, n = 300.

	IT				**CPU**			
	0	**0.2**	**0.4**	**0.9**	**0**	**0.2**	0.4	**0.9**
RCD	34,953	68,289	150,982	-	0.1993	0.4009	0.9189	-
RCDm	30,908	54,842	120,490	4,374,385	0.1715	0.3642	0.6787	24.3558
NARCD	8921	14,960	25,714	469,083	0.0708	0.1148	0.2134	3.5497

Table 10. The speed-up of the RCD method against the NARCD and RCDm, $A \in R^{m\times n}$ m = 800, n = 300.

	0	**0.2**	**0.4**	**0.9**
$speed - up_1$	1.1620	1.1007	1.3539	-
$speed - up_2$	2.5291	3.4921	4.3059	-

Table 11. IT and CPU of RCD, RCDm, NARCD for matrices $A \in R^{m\times n}$ m = 1000, n = 800.

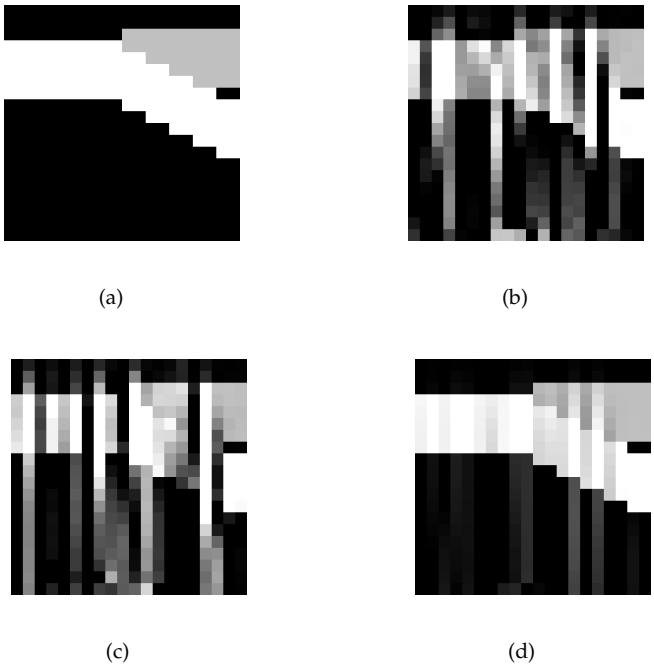
	IT				**CPU**			
	0	**0.2**	**0.4**	**0.9**	**0**	**0.2**	**0.4**	**0.9**
RCD	1,111,363	1,694,262	3,609,833	-	8.0508	11.9815	25.3595	-
RCDm	746,948	1,128,363	2,190,455	-	5.0748	7.7701	15.2623	-
NARCD	90,521	145,186	209,795	1,123,632	0.8005	1.0775	1.8670	10.4081

Table 12. The speed-up of the RCD method against the NARCD and RCDm, $A \in R^{m\times n}$ m = 1000, n = 800.

	0	**0.2**	**0.4**	**0.9**
$speed - up_1$	1.5864	1.5420	1.6615	-
$speed - up_2$	10.0572	11.1197	13.5830	-

4.4. The Two-Dimensional Tomography Test Problems

In this section, we use the previously and newly proposed methods to reconstruct 2D seismic travel time tomography. The 2D seismic travel-time tomography reconstruction is implemented in the function seismictomo (N, s, p) in the MATLAB package AIR TOOLS [29], which generated a sparse matrix A, an exact solution x_*(which is shown in Figure 4a) and the right vector $b + \epsilon = Ax^* + \epsilon$ where $\epsilon \in N(A^T)$. We set N = 20, s = 30 and p = 100 in the function seismictomo (N, s, p). We utilize the RCD, RCDm(δ=0.3) and NARCD($\lambda = 0.05$) methods to solve the linear least-squares problem (1). The experiment ran 90,000 iterations. From Figure 5, we see that the results of the NARCD methods are better than those of the RCD and RCDm methods under the same number of iteration steps.



(a) (b)

(c) (d)

Figure 5. Performance of RCD, RCDm, and NARCD methods for the seismictomo test problem. (**a**) Exact seismic. (**b**) RCD. (**c**) RCDm. (**d**) NARCD.

5. Conclusions

To solve a large system of linear equations, two new acceleration methods for the RCD method are proposed, called the NARCD method and the RCDm method. Their convergences were proved, and the estimations of the convergence rates of the NARCD method and the RCDm method are given, respectively. The two methods are shown to be equally successful in numerical experiments. In uniformly distributed data, with appropriately chosen momentum parameters, the RCDm is better than the RCD in IT and CPU. The NARCD and the RCDm are faster than the RCD, and the NARCD has a better acceleration effect than the RCDm. In the case of an overdetermined linear system, for the NARCD method, the fatter the matrix, the better the acceleration. The acceleration effect of NARCD becomes better when c in the MATLAB function unifrnd (c, 1, m,n) increases. The block coordinate descent method is a very efficient method for solving large linear equations; in future work, it would be interesting to apply the two accelerated formats to the block coordinate descent method.

Author Contributions: Software, W.B.; Validation, F.Z.; Investigation, Q.W.; Writing—original draft, Q.W.; Writing—review and editing, W.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported by the National Key Research and Development program of China (2019YFC1408400).

Data Availability Statement: The data that support the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Leventhal, D.; Lewis, A.S. Randomized methods for linear constraints: Convergence rates and conditioning. *Math. Oper. Res.* **2010**, *35*, 641–654. [CrossRef]
2. Ruhe, A. Numerical aspects of Gram-Schmidt orthogonalization of vectors. *Linear Algebra Its Appl.* **1983**, *52*, 591–601. [CrossRef]
3. Breheny, P.; Huang, J. Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. *Ann. Appl. Stat.* **2011**, *5*, 232. [CrossRef]
4. Chang, K.W.; Hsieh, C.J.; Lin, C.J. Coordinate descent method for large-scale l2-loss linear support vector machines. *J. Mach. Learn. Res.* **2008**, *9*, 1369–1398.
5. Canutescu, A.A.; Dunbrack Jr, R.L. Cyclic coordinate descent: A robotics algorithm for protein loop closure. *Protein Sci.* **2003**, *12*, 963–972. [CrossRef] [PubMed]
6. Bouman, C.A.; Sauer, K. A unified approach to statistical tomography using coordinate descent optimization. *IEEE Trans. Image Process.* **1996**, *5*, 480–492. [CrossRef] [PubMed]
7. Ye, J.C.; Webb, K.J.; Bouman, C.A.; Millane, R.P. Optical diffusion tomography by iterative-coordinate-descent optimization in a Bayesian framework. *JOSA A* **1999**, *16*, 2400–2412. [CrossRef]
8. Bai, Z.Z.; Wu, W.T. On greedy randomized coordinate descent methods for solving large linear least-squares problems. *Numer. Linear Algebra Appl.* **2019**, *26*, e2237. [CrossRef]
9. Zhang, J.; Guo, J. On relaxed greedy randomized coordinate descent methods for solving large linear least-squares problems. *Appl. Numer. Math.* **2020**, *157*, 372–384. [CrossRef]
10. Lu, Z.; Xiao, L. On the complexity analysis of randomized block-coordinate descent methods. *Math. Program.* **2015**, *152*, 615–642. [CrossRef]
11. Necoara, I.; Nesterov, Y.; Glineur, F. Random block coordinate descent methods for linearly constrained optimization over networks. *J. Optim. Theory Appl.* **2017**, *173*, 227–254. [CrossRef]
12. Richtárik, P.; Takáč, M. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Math. Program.* **2014**, *144*, 1–38. [CrossRef]
13. Karczmarz, S. Angenaherte auflosung von systemen linearer glei-chungen. *Bull. Int. Acad. Pol. Sic. Let. Cl. Sci. Math. Nat.* **1937**, *35*, 355–357.
14. Strohmer, T.; Vershynin, R. A randomized Kaczmarz algorithm with exponential convergence. *J. Fourier Anal. Appl.* **2009**, *15*, 262–278. [CrossRef]
15. Bai, Z.Z.; Wu, W.T. On greedy randomized Kaczmarz method for solving large sparse linear systems. *SIAM J. Sci. Comput.* **2018**, *40*, A592–A606. [CrossRef]
16. Bai, Z.Z.; Wu, W.T. On relaxed greedy randomized Kaczmarz methods for solving large sparse linear systems. *Appl. Math. Lett.* **2018**, *83*, 21–26. [CrossRef]
17. Liu, Y.; Gu, C.Q. Variant of greedy randomized Kaczmarz for ridge regression. *Appl. Numer. Math.* **2019**, *143*, 223–246. [CrossRef]
18. Guan, Y.J.; Li, W.G.; Xing, L.L.; Qiao, T.T. A note on convergence rate of randomized Kaczmarz method. *Calcolo* **2020**, *57*, 1–11. [CrossRef]
19. Du, K.; Gao, H. A new theoretical estimate for the convergence rate of the maximal weighted residual Kaczmarz algorithm. *Numer. Math. Theory Methods Appl* **2019**, *12*, 627–639.
20. Yang, X. A geometric probability randomized Kaczmarz method for large scale linear systems. *Appl. Numer. Math.* **2021**, *164*, 139–160. [CrossRef]
21. Nesterov, Y. A method for unconstrained convex minimization problem with the rate of convergence O (1/kˆ 2). *Dokl. Akad. Nauk Sssr* **1983**, *269*, 543–547.
22. Nesterov, Y. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM J. Optim.* **2012**, *22*, 341–362. [CrossRef]
23. Polyak, B.T. Some methods of speeding up the convergence of iteration methods. *Ussr Comput. Math. Math. Phys.* **1964**, *4*, 1–17. [CrossRef]
24. Sun, T.; Li, D.; Quan, Z.; Jiang, H.; Li, S.; Dou, Y. Heavy-ball algorithms always escape saddle points. *arXiv* **2019**, arXiv:1907.09697.
25. Sarowar Morshed, M.; Saiful Islam, M. Accelerated Sampling Kaczmarz Motzkin Algorithm for The Linear Feasibility Problem. *J. Glob. Optim.* **2019**, *77*, 361–382. [CrossRef]
26. Liu, J.; Wright, S. An accelerated randomized Kaczmarz algorithm. *Math. Comput.* **2016**, *85*, 153–178. [CrossRef]
27. Loizou, N.; Richtárik, P. Momentum and stochastic momentum for stochastic gradient, newton, proximal point and subspace descent methods. *Comput. Optim. Appl.* **2020**, *77*, 653–710. [CrossRef]

28. Higham, D.J.; Higham, N.J. *MATLAB Guide*; SIAM: Philadelphia, PA, USA, 2016.
29. Hansen, P.C.; Jørgensen, J.S. AIR Tools II: Algebraic iterative reconstruction methods, improved implementation. *Numer. Algorithms* **2018**, *79*, 107–137. [CrossRef]