mathematics

MDPI

Article

Randomized Block Kaczmarz Methods for Inner Inverses of a Matrix

Lili Xing, Wendi Bao *, Ying Lv, Zhiwei Guo and Weiguo Li

College of Science, China University of Petroleum, Qingdao 266580, China; xinglily2010@upc.edu.cn (L.X.); lyrr1017@163.com (Y.L.); gzw_13605278246@163.com (Z.G.); liwg@upc.edu.cn (W.L.)
* Correspondence: baowd@upc.edu.cn

Abstract: In this paper, two randomized block Kaczmarz methods to compute inner inverses of any rectangular matrix A are presented. These are iterative methods without matrix multiplications and their convergence is proved. The numerical results show that the proposed methods are more efficient than iterative methods involving matrix multiplications for the high-dimensional matrix.

Keywords: rectangular matrix; block Kaczmarz method; inner inverse; convergence

MSC: 65F10; 65F45; 65H10

1. Introduction

Consider the linear matrix equation

$$AXA = A, \tag{1}$$

where $A \in \mathbb{C}^{m\times n}$ and $X \in \mathbb{C}^{n\times m}$. The solution X of (1) is called the inner inverse of A. For arbitrary $X^{(0)} \in \mathbb{C}^{n\times m}$, all the inner inverses of A can be expressed as

$$X_0^- = X^{(0)} + A^\dagger - A^\dagger A X^{(0)} A A^\dagger,$$

where $A^\dagger$ is the Moore–Penrose generalized inverse of A.

Inner inverses play a role in solving systems of linear equations, finding solutions to least squares problems, and characterizing the properties of linear transformations [1]. They are also useful in various areas of engineering, such as robotics, big data analysis, network learning, sensory fusion, and so on [2–6].

To calculate the inner inverse of a matrix, various methods can be used, such as the Moore–Penrose pseudoinverse, singular value decomposition (SVD), or the method of partitioned matrices [1]. To our knowledge, few people discuss numerical methods for solving all the inner inverses of a matrix. In [7], the authors designed an iterative method based on gradient (GBMC) to solve the matrix Equation (1), which has the following iterative formula:

$$X^{(k+1)} = X^{(k)} + \mu A^*(A - AX^{(k)}A)A^*,\ k = 0, 1, 2, \ldots.$$

Here, $0 < \mu < \frac{2}{\|A\|_2^4}$ is called the convergence factor. If the initial matrix $X^{(0)} = A^*$, then the sequence $X^{(k)}$ converges to $A^\dagger$. Recently, various nonlinear and linear recurrent neural network (RNN) models have been developed for computing the pseudoinverse of any rectangular matrices (for more details, see [8–11]). The gradient-based neural network (GNN), whose derivation is based on the gradient of an nonnegative energy function, is an alternative for calculating the Moore–Penrose generalized inverses [12–15]. These methods for solving the inner inverses and for other generalized inverses of a matrix frequently use

check for updates

Citation: Xing, L.; Bao, W.; Lv, Y.; Guo, Z.; Li, W. Randomized Block Kaczmarz Methods for Inner Inverses of a Matrix. *Mathematics* **2024**, *12*, 475. https://doi.org/10.3390/math12030475

Academic Editor: Luca Gemignani

Received: 10 December 2023
Revised: 25 January 2024
Accepted: 30 January 2024
Published: 2 February 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

the matrix–matrix product operation, and consume a lot of computing time. In this paper, we aimed to explore two block Kaczmarz methods to solve (1) by the product of the matrix and vector.

In this paper, we denote A^*, $A^\dagger$, A^-, $\|A\|_2$, $\|A\|_F$ and $\langle A, B\rangle_F = \text{trace}(A^*B)$ as the conjugate transpose, the Moore–Penrose pseudoinverse, the inner inverse, the 2-norm, the Frobenius norm of A and the inner product of two matrices A and B, respectively. In addition, for a given matrix $A = (a_{ij}) \in \mathbb{C}^{m\times n}$, $A_{i,:}$, $A_{:,j}$, $R(A)$, $\sigma_{\max}(A)$ and $\sigma_{\min}(A)$, are used to denote its ith row, jth column, the column space of A, the maximum singular value and the smallest nonzero singular value of A, respectively. Recall that $\sigma_{\max}(A) = \|A\|_2$ and $\sigma_{\min}(A^\dagger) = \frac{1}{\|A\|_2}$. For an integer $m > 1$, let $[m] = \{1, 2, \ldots, m\}$. Let $\mathbb{E}_k$ denote the expected value conditional on the first k iterations; that is,

$$\mathbb{E}_k[\cdot] = \mathbb{E}[\cdot | j_0, j_1, \ldots, j_{k-1}],$$

where $j_s (s = 0, 1, ..., k-1)$ is the column chosen at the sth iteration.

The rest of this paper is organized as follows. In Section 2, we derive the projected randomized block Kaczmarz method (MII-PRBK) for finding the inner inverses of a matrix and give its theoretical analysis. In Section 3, we discuss the randomized average block Kaczmarz method (MII-RABK) and its convergence results. In Section 4, some numerical examples are provided to illustrate the effectiveness of the proposed methods. Finally, some brief concluding remarks are described in Section 5.

2. Projected Randomized Block Kaczmarz Method for Inner Inverses of a Matrix

The classical Kaczmarz method is a row iterative scheme for solving the linear consistent system $Ax = b$ that requires only $O(n)$ cost per iteration and storage and has a linear rate of convergence [16]. At each step, the method projects the current iteration onto the solution space of a single constraint. In this section, we are concerned with the randomized Kaczmarz method to solve the matrix Equation (1). At the kth iteration, we find the next iterate $X^{(k+1)}$ that is closest to the current iteration $X^{(k)}$ in the Frobenius norm under the ith condition $A_{i,:}XA = A_{i,:}$. Hence, $X^{(k+1)}$ is the optimal solution of the following constrained optimization problem:

$$\min_{X\in\mathbb{C}^{n\times m}} \frac{1}{2}\|X - X^{(k)}\|_F^2 \ \ s.t. \ \ A_{i,:}XA = A_{i,:}, i \in [m]. \tag{2}$$

Using the Lagrange multiplier method, turn (2) into the unconstrained optimization problem

$$\min_{X\in\mathbb{C}^{n\times m}, Y\in\mathbb{C}^{n\times 1}} L(X, Y) = \min_{X\in\mathbb{C}^{n\times m}, Y\in\mathbb{C}^{n\times 1}} \left\{ \frac{1}{2}\|X - X^{(k)}\|_F^2 + \langle Y, (A_{i,:}XA - A_{i,:})^* \rangle \right\}. \tag{3}$$

Differentiating Lagrangian function $L(X, Y)$ with respect to X and setting to zero gives $X^{(k+1)} = X^{(k)} - A_{i,:}^* Y^* A^*$. Substituting into (3) and differentiating $L(X, Y)$ with respect to Y, we can obtain $Y^* = -\frac{1}{\|A_{i,:}\|_2^2}(A_{i,:} - A_{i,:}X^{(k)}A)(A^*A)^\dagger$. So, the projected randomized block Kaczmarz for solving $AXA = A$ iterates as

$$X^{(k+1)} = X^{(k)} + \frac{A_{i,:}^*}{\|A_{i,:}\|_2^2}(A_{i,:} - A_{i,:}X^{(k)}A)A^\dagger, k = 0, 1, 2, \ldots, \tag{4}$$

where $i \in [m]$ is selected with probability $p_i = \frac{\|A_{i,:}\|_2^2}{\|A\|_F^2}$. We describe this method as Algorithm 1, which is called the MII-PRBK algorithm.

Algorithm 1 Projected randomized block Kaczmarz method for matrix inner inverses (MII-PRBK)

Input: $A \in \mathbb{C}^{m\times n}$, $X^{(0)} \in \mathbb{C}^{n\times m}$

1: **for** $k = 0, 1, 2, \ldots,$ **do**

2: Pick i with probability $p_i(A) = \frac{\|A_{i,:}\|_2^2}{\|A\|_F^2}$

3: Compute $X^{(k+1)} = X^{(k)} + \frac{A_{i,:}^*}{\|A_{i,:}\|_2^2}\left((A_{i,:} - (A_{i,:}X^{(k)})A)A^\dagger\right)$

4: **end for**

The following lemmas will be used extensively in this paper. Their proofs are straightforward.

Lemma 1. *Let $A \in \mathbb{C}^{m\times n}$ be given. For any vector $u \in R(A^*)$, it holds that*

$$\|Au\|_2^2 \geq \sigma_{\min}^2(A)\|u\|_2^2. \tag{5}$$

Lemma 2 ([17])**.** *Let $A \in \mathbb{C}^{m\times n}$ be given. For any matrix $B \in \mathbb{C}^{n\times m}$, if $B_{:,j} \in R(A^*), j = 1, 2, \ldots, m$, it holds that*

$$\|AB\|_F^2 \geq \sigma_{\min}^2(A)\|B\|_F^2. \tag{6}$$

Using Lemma 2 twice, and the fact $\|AB\|_F^2 = \|BA\|_F^2$, we can obtain Lemma 3.

Lemma 3. *Let $A \in \mathbb{C}^{m\times n}$ be given. For any matrix $B \in \mathbb{C}^{n\times m}$, if $B_{:,j} \in R(A^*), j = 1, 2, \ldots, m$ and $(B_{i,:})^* \in R(A), i = 1, 2, \ldots, n$, it holds that*

$$\|ABA\|_F^2 \geq \sigma_{\min}^4(A)\|B\|_F^2. \tag{7}$$

Remark 1. *Lemma 3 can be seen as a special case of Lemma 1 in [18]. That is, let $\mathcal{B} = \{B \in \mathbb{C}^{n\times m} \mid \exists Y \in \mathbb{C}^{m\times n}\ s.t.\ B = A^*YA^*\}$. For any matrix $B \in \mathcal{B}$, it holds $\|ABA\|_F^2 \geq \sigma_{\min}^4(A)\|B\|_F^2$. Notice that $\mathcal{B}$ is well defined because $0 \in \mathcal{B}$ and $A^\dagger \in \mathcal{B}$.*

Theorem 1. *The sequence $\{X^{(k)}\}$ generated by the MII-PRBK method starting from any initial matrix $X^{(0)} \in \mathbb{C}^{n\times m}$ converges linearly to X_0^- in the mean square form. Moreover, the solution error in expectation for the iteration sequence $X^{(k)}$ obeys*

$$\mathbb{E}\left[\left\|X^{(k)} - X_0^-\right\|_F^2\right] \leq \rho^k\left\|X^{(0)} - X_0^-\right\|_F^2, k = 1, 2, \ldots, \tag{8}$$

where $\rho = 1 - \frac{\sigma_{\min}^4(A)}{\|A\|_F^2\|A\|_2^2}$.

Proof. For $k = 0, 1, 2, \ldots$, by (4) and $AA^\dagger A = A$, we can obtain

$$AX_0^-A = A(X^{(0)} + A^\dagger - A^\dagger AX^{(0)}AA^\dagger)A = A \tag{9}$$

and

$$\begin{aligned} A_{i,:}X^{(k+1)}A &= A_{i,:}\left(X^{(k)} + \frac{A_{i,:}^*}{\|A_{i,:}\|_2^2}\left(A_{i,:} - A_{i,:}X^{(k)}A\right)A^\dagger\right)A \\ &= A_{i,:}X^{(k)}A + (A_{i,:} - A_{i,:}X^{(k)}A)A^\dagger A \\ &= A_{i,:}. \end{aligned} \tag{10}$$

Combining (9), (10) and $(A^\dagger)^* = A(A^*A)^\dagger$, it follows from

$$\begin{aligned}\langle X^{(k+1)} - X^{(k)}, X^{(k+1)} - X_0^- \rangle_F &= \frac{1}{\|A_{i,:}\|_2^2} \langle A_{i,:}^*(A_{i,:} - A_{i,:}X^{(k)}A)A^\dagger, X^{(k+1)} - X_0^- \rangle_F \\ &= \frac{1}{\|A_{i,:}\|_2^2} \text{trace}\Big((A^\dagger)^*(A_{i,:} - A_{i,:}X^{(k)}A)^* A_{i,:}(X^{(k+1)} - X_0^-)\Big) \\ &= \frac{1}{\|A_{i,:}\|_2^2} \text{trace}\Big((A_{i,:} - A_{i,:}X^{(k)}A)^* A_{i,:}(X^{(k+1)} - X_0^-)(A^\dagger)^*\Big) \\ &= 0 \text{ (by } A_{i,:}X^{(k+1)}A = A_{i,:}X_0^-A = A_{i,:}\text{)}\end{aligned}$$

and

$$\begin{aligned}\left\|X^{(k+1)} - X^{(k)}\right\|_F^2 &= \frac{1}{\|A_{i,:}\|_2^4} \left\|A_{i,:}^*(A_{i,:} - A_{i,:}X^{(k)}A)A^\dagger\right\|_F^2 \\ &= \frac{1}{\|A_{i,:}\|_2^4} \text{trace}\Big((A^\dagger)^*(A_{i,:} - A_{i,:}X^{(k)}A)^* A_{i,:}A_{i,:}^*(A_{i,:} - A_{i,:}X^{(k)}A)A^\dagger\Big) \\ &= \frac{1}{\|A_{i,:}\|_2^2} \left\|(A^\dagger)^*(A_{i,:} - A_{i,:}X^{(k)}A)^*\right\|_2^2 \\ &\geq \frac{\sigma_{\min}^2(A^\dagger)}{\|A_{i,:}\|_2^2} \left\|A_{i,:} - A_{i,:}X^{(k)}A\right\|_2^2 \text{ (by Lemma 1)}\end{aligned}$$

that

$$\begin{aligned}\|X^{(k+1)} - X_0^-\|_F^2 &= \left\|X^{(k)} - X_0^-\right\|_F^2 - \left\|X^{(k+1)} - X^{(k)}\right\|_F^2 \\ &\leq \left\|X^{(k)} - X_0^-\right\|_F^2 - \frac{\sigma_{\min}^2(A^\dagger)}{\|A_{i,:}\|_2^2} \|A_{i,:} - A_{i,:}X^{(k)}A\|_2^2.\end{aligned}$$

By taking the conditional expectation, we have

$$\begin{aligned}\mathbb{E}_k\left[\left\|X^{(k+1)} - X_0^-\right\|_F^2\right] &\leq \left\|X^{(k)} - X_0^-\right\|_F^2 - \sigma_{\min}^2(A^\dagger)\mathbb{E}_k\left[\frac{\|A_{i,:} - A_{i,:}X^{(k)}A\|_2^2}{\|A_{i,:}\|_2^2}\right] \\ &= \left\|X^{(k)} - X_0^-\right\|_F^2 - \sigma_{\min}^2(A^\dagger) \sum_{i=1}^m \frac{\|A_{i,:}\|_2^2}{\|A\|_F^2} \frac{\|A_{i,:} - A_{i,:}X^{(k)}A\|_2^2}{\|A_{i,:}\|_2^2} \\ &= \left\|X^{(k)} - X_0^-\right\|_F^2 - \frac{\sigma_{\min}^2(A^\dagger)}{\|A\|_F^2} \left\|A - AX^{(k)}A\right\|_F^2 \\ &= \left\|X^{(k)} - X_0^-\right\|_F^2 - \frac{\sigma_{\min}^2(A^\dagger)}{\|A\|_F^2} \left\|A(X^{(k)} - X_0^-)A\right\|_F^2 \\ &\leq \left\|X^{(k)} - X_0^-\right\|_F^2 - \frac{\sigma_{\min}^4(A)\sigma_{\min}^2(A^\dagger)}{\|A\|_F^2} \left\|X^{(k)} - X_0^-\right\|_F^2 \\ &= \left(1 - \frac{\sigma_{\min}^4(A)}{\|A\|_F^2\|A\|_2^2}\right) \left\|X^{(k)} - X_0^-\right\|_F^2.\end{aligned}$$

The second inequality is obtained by Lemma 3 because $X^{(0)} - X_0^- = A^\dagger - A^\dagger AX^{(0)}AA^\dagger \in \mathcal{B}$ and $X^{(k)} - X_0^- \in \mathcal{B}$ on induction.

By the law of total expectation, we have

$$\mathbb{E}\left[\left\|X^{(k)} - X_0^-\right\|_F^2\right] \leq \rho\mathbb{E}\left[\left\|X^{(k-1)} - X_0^-\right\|_F^2\right] \leq \cdots \leq \rho^k \left\|X^{(0)} - X_0^-\right\|_F^2, k = 1, 2, \ldots.$$

This completes the proof. □

3. Randomized Average Block Kaczmarz Method for Inner Inverses of a Matrix

In practice, the main drawback of (4) is that each iteration is expensive and difficult to parallelize, since the Moore–Penrose inverse $A^\dagger$ is needed to compute. In addition, $A^\dagger$ is unknown or too large to store in some practical problem. It is necessary to develop the pseudoinverse-free methods to compute the inner inverses of large-scale matrices. In this section, we exploit the average block Kaczmarz method [16,19] for solving linear equations to matrix equations.

At each iteration, the PRBK method (4) does an orthogonal projection of the current estimate matrix X^k onto the corresponding hyperplane $H_i = \{X \in \mathbb{C}^{n\times m} | A_{i,:}XA = A_{i,:}\}$. Next, instead of projecting on the hyperplane H_i, we consider the approximate solution $X^{(k+1)}$ by projecting the current estimate $X^{(k)}$ onto the hyperplane $H_{i,j} = \{X \in \mathbb{C}^{n\times m} | A_{i,:}X^{(k)}A_{:,j} = A_{i,j}\}$. That is,

$$X^{(k+1)} = X^{(k)} + \frac{A_{i,:}^*(A_{i,:} - A_{i,:}X^{(k)}A_{:,j})A_{:,j}^*}{\|A_{i,:}\|_2^2\|A_{:,j}\|_2^2}.$$

Then, we take a convex combination of all directions $A_{:,j}$ (the weight is $\frac{\|A_{:,j}\|_2^2}{\|A\|_F^2}$) with some stepsize $\lambda > 0$, and obtain the following average block Kaczmarz method:

$$X^{(k+1)} = X^{(k)} + \frac{\lambda}{\|A\|_F^2}\frac{A_{i,:}^*}{\|A_{i,:}\|_2^2}(A_{i,:} - A_{i,:}X^{(k)}A)A^*.$$

Setting $\alpha = \frac{\lambda}{\|A\|_F^2} > 0$, we obtain the following randomized block Kaczmarz iteration

$$X^{(k+1)} = X^{(k)} + \frac{\alpha}{\|A_{i,:}\|_2^2}A_{i,:}^*(A_{i,:} - A_{i,:}X^{(k)}A)A^*, k = 0, 1, 2, \ldots, \tag{11}$$

where $i \in [m]$ is selected with probability $p_i = \frac{\|A_{i,:}\|_2^2}{\|A\|_F^2}$. The cost of each iteration of this method is $8mn + n - 2m$ if the square of the row norm of A has been calculated in advance. We describe this method as Algorithm 2, which is called the MII-RABK algorithm.

Algorithm 2 Randomized average block Kaczmarz method for matrix inner inverses (MII-RABK)

Input: $A \in \mathbb{C}^{m\times n}$, $X^{(0)} \in \mathbb{C}^{n\times m}$ and $\alpha \in \mathbb{R}$

1: **for** $k = 0, 1, 2, \ldots,$ **do**
2: Pick i with probability $p_i(A) = \frac{\|A_{i,:}\|_2^2}{\|A\|_F^2}$
3: Compute $X^{(k+1)} = X^{(k)} + \frac{\alpha}{\|A_{i,:}\|_2^2}A_{i,:}^*\left((A_{i,:} - (A_{i,:}X^{(k)})A)A^*\right)$
4: **end for**

In the following theorem, with the idea of the RK method [20], we show that the iteration (11) converges linearly to the matrix $X_0^- = X^{(0)} + A^\dagger - A^\dagger AX^{(0)}AA^\dagger$ for any initial matrix $X^{(0)}$.

Theorem 2. *Assume* $0 < \alpha < \frac{2}{\|A\|_2^2}$. *The sequence* $\{X^{(k)}\}$ *generated by the MII-RABK method starting from any initial matrix* $X^{(0)} \in \mathbb{C}^{n\times m}$ *converges linearly to* X_0^- *in mean square form. Moreover, the solution error in expectation for the iteration sequence* $X^{(k)}$ *obeys*

$$\mathbb{E}\left[\left\|X^{(k)} - X_0^-\right\|_F^2\right] \le \hat{\rho}^k\left\|X^{(0)} - X_0^-\right\|_F^2, k = 1, 2, \ldots, . \tag{12}$$

where $\hat{\rho} = 1 - \frac{2\alpha - \alpha^2\|A\|_2^2}{\|A\|_F^2}\sigma_{\min}^4(A)$.

Proof. For $k = 0, 1, 2, \ldots$, by (11) and $AX_0^-A = A$, we have

$$\begin{aligned} X^{(k+1)} - X_0^- &= X^{(k)} + \frac{\alpha}{\|A_{i,:}\|_2^2} A_{i,:}^*(A_{i,:} - A_{i,:}X^{(k)}A)A^* - X_0^- \\ &= (X^{(k)} - X_0^-) - \frac{\alpha}{\|A_{i,:}\|_2^2} A_{i,:}^* A_{i,:}(X^{(k)} - X_0^-)AA^*, \end{aligned}$$

then

$$\begin{aligned} \|X^{(k+1)} - X_0^-\|_F^2 = \left\|X^{(k)} - X_0^-\right\|_F^2 &+ \frac{\alpha^2}{\|A_{i,:}\|_2^4} \left\|A_{i,:}^* A_{i,:}(X^{(k)} - X_0^-)AA^*\right\|_F^2 \\ &- \frac{2\alpha}{\|A_{i,:}\|_2^2} \langle X^{(k)} - X_0^-, A_{i,:}^* A_{i,:}(X^{(k)} - X_0^-)AA^* \rangle_F. \end{aligned}$$

It follows from

$$\begin{aligned} &\frac{\alpha^2}{\|A_{i,:}\|_2^4} \left\|A_{i,:}^* A_{i,:}(X^{(k)} - X_0^-)AA^*\right\|_F^2 \\ &= \frac{\alpha^2}{\|A_{i,:}\|_2^2} \left\|A_{i,:}(X^{(k)} - X_0^-)AA^*\right\|_2^2 \text{ (by trace}(uu^*) = \|u\|_2^2 \text{ for any vector } u) \\ &\leq \frac{\alpha^2 \|A\|_2^2}{\|A_{i,:}\|_2^2} \left\|A_{i,:}(X^{(k)} - X_0^-)A\right\|_2^2 \text{ (by } \|u^*A^*\|_2 = \|Au\|_2 \leq \|A\|_2 \|u\|_2), \end{aligned}$$

and

$$\begin{aligned} &\frac{2\alpha}{\|A_{i,:}\|_2^2} \langle X^{(k)} - X_0^-, A_{i,:}^* A_{i,:}(X^{(k)} - X_0^-)AA^* \rangle_F \\ &= \frac{2\alpha}{\|A_{i,:}\|_2^2} \text{trace}(A^*(X^{(k)} - X_0^-)^* A_{i,:}^* A_{i,:}(X^{(k)} - X_0^-)A) (\text{ by trace}(MN) = \text{trace}(NM)) \\ &= \frac{2\alpha}{\|A_{i,:}\|_2^2} \left\|A_{i,:}(X^{(k)} - X_0^-)A\right\|_2^2 \end{aligned}$$

that

$$\|X^{(k+1)} - X_0^-\|_F^2 \leq \left\|X^{(k)} - X_0^-\right\|_F^2 - \frac{2\alpha - \alpha^2\|A\|_2^2}{\|A_{i,:}\|_2^2} \left\|A_{i,:}(X^{(k)} - X_0^-)A\right\|_2^2.$$

By taking the conditional expectation, we have

$$\begin{aligned} \mathbb{E}_k\left[\left\|X^{(k+1)} - X_0^-\right\|_F^2\right] &\leq \left\|X^{(k)} - X_0^-\right\|_F^2 - \mathbb{E}_k\left[\frac{2\alpha - \alpha^2\|A\|_2^2}{\|A_{i,:}\|_2^2} \left\|A_{i,:}(X^{(k)} - X_0^-)A\right\|_2^2\right] \\ &= \left\|X^{(k)} - X_0^-\right\|_F^2 - \sum_{i=1}^m \frac{\|A_{i,:}\|_2^2}{\|A\|_F^2} \frac{2\alpha - \alpha^2\|A\|_2^2}{\|A_{i,:}\|_2^2} \left\|A_{i,:}(X^{(k)} - X_0^-)A\right\|_2^2 \\ &= \left\|X^{(k)} - X_0^-\right\|_F^2 - \frac{2\alpha - \alpha^2\|A\|_2^2}{\|A\|_F^2} \left\|A(X^{(k)} - X_0^-)A\right\|_F^2. \end{aligned}$$

Noting that $X^{(0)} - X_0^- = A^\dagger AX^{(0)}AA^\dagger - A^\dagger \in \mathcal{B}$ and $\frac{\alpha}{\|A_{i,:}\|_2^2} A_{i,:}^* A_{i,:}(X^{(k)} - X_0^-)AA^* \in \mathcal{B}$, we have $X^{(k+1)} - X_0^- \in \mathcal{B}$ by induction. Then, by Lemma 3 and $0 < \alpha < \frac{2}{\|A\|_2^2}$, we can obtain

$$\begin{aligned} \mathbb{E}_k\left[\left\|X^{(k+1)} - X_0^-\right\|_F^2\right] &\leq \left\|X^{(k)} - X_0^-\right\|_F^2 - \frac{2\alpha - \alpha^2\|A\|_2^2}{\|A\|_F^2} \sigma_{\min}^4(A) \left\|X^{(k)} - X_0^-\right\|_F^2 \\ &= \left(1 - \frac{2\alpha - \alpha^2\|A\|_2^2}{\|A\|_F^2} \sigma_{\min}^4(A)\right) \left\|X^{(k)} - X_0^-\right\|_F^2. \end{aligned} \tag{13}$$

Finally, by (13) and induction on the iteration index k, we obtain the estimate (12). This completes the proof. □

Remark 2. *If $X^{(0)} = 0$ or A^*, then $X_0^- = A^\dagger$, which is the unique minimum norm least squares solution of (2). Theorems 1 and 2 imply that the sequence $X^{(k)}$ generated by the MII-PRBK or MII-RABK method with $X^{(0)} = 0$ or A^* converges linearly to $A^\dagger$.*

Remark 3. *Noting that*

$$1 - \frac{1}{\|A\|_F^2\|A\|_2^2}\sigma_{\min}^4(A) \le 1 - \frac{2\alpha - \alpha^2\|A\|_2^2}{\|A\|_F^2}\sigma_{\min}^4(A),\ \alpha \in (0, \frac{2}{\|A\|_2^2})$$

this means that the convergence factor of the MII-PRBK method is smaller than that of MII-RABK method. However, in practice, it is very expensive to calculate the pseudoinverse of large-scale matrices.

Remark 4. *Replacing A^* in (11) with $A^\dagger$, we obtain the following relaxed projected randomized block Kaczmarz method (MII-PRBKr)*

$$X^{(k+1)} = X^{(k)} + \frac{\alpha}{\|A_{i,:}\|_2^2}A_{i,:}^*(A_{i,:} - A_{i,:}X^{(k)}A)A^\dagger, k = 0, 1, 2, \ldots, \tag{14}$$

where $0 < \alpha < 2$ is the step size, and i is selected with probability $p_i = \frac{\|A_{i,:}\|_2^2}{\|A\|_F^2}$. By the similar approach as used in the proof of Theorem 2, we can prove that the iteration $X^{(k)}$ satisfies the following estimate

$$E\left[\left\|X^{(k)} - X_0^-\right\|_F^2\right] \le \tilde{\rho}^k\left\|X^{(0)} - X_0^-\right\|_F^2, \tag{15}$$

where $\tilde{\rho} = 1 - \frac{(2\alpha-\alpha^2)\sigma_{\min}^4(A)}{\|A\|_2^2\|A\|_F^2}$. It is obvious that when $\alpha = 1$, the MII-PRBKr iteration (14) is actually the MII-PRBK iteration (4).

4. Numerical Experiments

In this section, we will present some experiment results of the proposed algorithms for solving the inner inverse, and compare them with GBMC [7] for rectangular matrices. All experiments are carried out by using MATLAB (version R2020a) in a personal computer with Intel(R) Core(TM) i7-4712MQ CPU @2.30 GHz, RAM 8 GB and Windows 10.

All computations are started with the random matrices $X^{(0)}$, and terminated once the relative error (RE) of the solution, defined by

$$RE = \frac{\left\|X^{(k)} - X_0^-\right\|_F}{\|X_0^-\|_F}$$

at the current iteration $X^{(k)}$, satisfies $RE < 10^{-6}$ or exceeds the maximum iteration $K = 10^6$. We report the average number of iterations (denoted as "IT") and the average computing time in seconds (denoted as"CPU") for 10 trials repeated runs of the MII-PRBK and MII-RABK methods. For clarity, we restate three methods as follows.

- GBMC ([7])

$$X^{(k+1)} = X^{(k)} + \mu A^*(A - AX^{(k)}A)A^*, 0 < \mu < \frac{2}{\|A\|_2^4}.$$

- MII-PRBK (Algorithm 1)

$$X^{(k+1)} = X^{(k)} + \frac{A^*_{i,:}}{\|A_{i,:}\|_2^2}(A_{i,:} - A_{i,:}X^{(k)}A)A^\dagger, p_i = \frac{\|A_{i,:}\|_2^2}{\|A\|_F^2}.$$

- MII-RABK (Algorithm 2)

$$X^{(k+1)} = X^{(k)} + \frac{\alpha A^*_{i,:}}{\|A_{i,:}\|_2^2}(A_{i,:} - A_{i,:}X^{(k)}A)A^*, 0 < \alpha < \frac{2}{\|A\|_2^2}, p_i = \frac{\|A_{i,:}\|_2^2}{\|A\|_F^2}.$$

We underscore once again the difference between the two algorithms; that is, Algorithm 1 needs the Moore–Penrose generalized inverse $A^\dagger$, whereas Algorithm 2 replaces $A^\dagger$ with A^* (which is easier to implement) and adds a stepsize parameter α.

Example 1. *For given m, n, the entries of A are generated from a standard normal distribution by a Matlab built-in function, i.e., $A = randn(m, n)$ or $A1 = randn(m/2, n/2), A = [A1, A1; A1, A1]$.*

Firstly, we test the impact of α in the MII-RABK method on the experimental results. To do this, we vary λ from 0.1 to 1.9 by step 0.1, where $\alpha = \frac{\lambda}{\|A\|_2^2}$ satisfies $0 < \alpha < \frac{2}{\|A\|_2^2}$ in Theorem 2. Figure 1 plots the IT and CPU versus different λ with different matrices in Table 1. From Figure 1, it can be seen that the number of iteration steps and the running time first decrease and then increase with the increase in λ, and almost achieve the minimum value when $\lambda \in [1.5, 1.7]$ for all matrices. Therefore, we set $\alpha = \frac{1.6}{\|A\|_2^2}$ in this example.

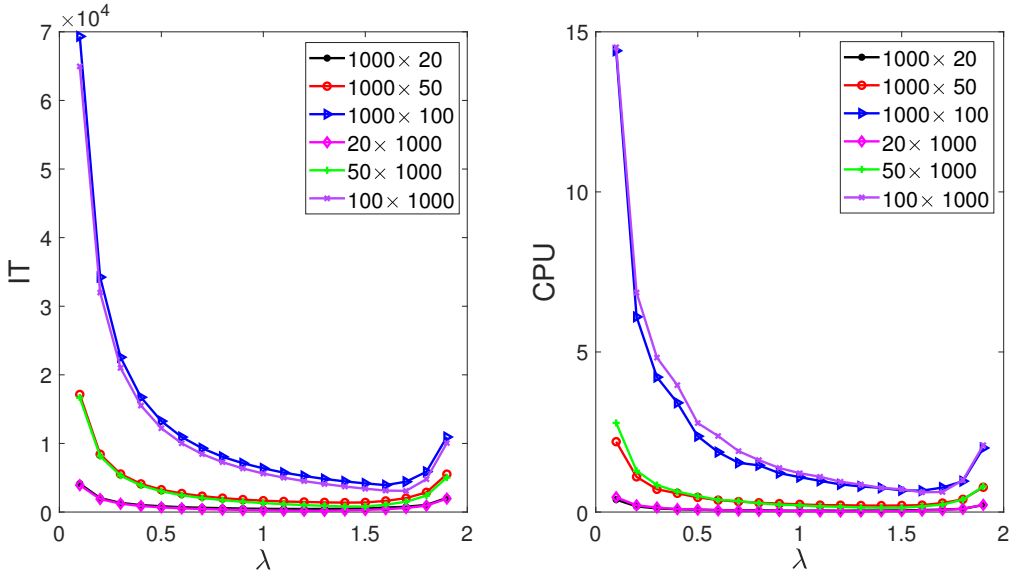

Figure 1. IT (**left**) and CPU (**right**) of different α of the RABK method for $A = randn(m, n)$ from Example 1.

The results of numerical experiments are listed in Tables 1 and 2. From these tables, it can be seen that the MII-GMBC method has the least number of iteration steps, whereas the MII-PRBK method has the least running time. Figure 2 plots the iteration steps and running time of different methods with the matrices $A = randn(m, 25)$ (top) and $A = randn(25, n)$ (bottom). It is pointed out that the initial points on the left plots (i.e., $A = randn(100, 25)$ and $A = randn(25, 100)$) indicate that the MII-RABK method requires a very large number of iteration steps, which is related to Kaczmarz's anomaly [21]. That is, the MII-RABK method enjoys a faster rate of convergence in the case where m is considerably smaller or larger than n. However, the closer m and n are, the slower the convergence is. As the

number of rows or columns increases, the iteration steps and runtime of the MII-PRBK and MII-RABK methods are increasing relatively slowly, while the runtime of the GMBC method grows dramatically (see the right plots in Figure 2). Therefore, our proposed methods are more suitable for large matrices. The curves of relative error $log_{10}(RE)$ versus the number of iterations "IT" and running time "CPU", given by the GMBC, MII-PRBK, MII-RABK methods for $A1 = randn(500, 50)$, $A = [A1, A1; A1, A1]$, are shown in Figure 3. From Figure 3, it can be seen that the relative error of GBMC method decays the fastest when the number of iterations increases and the relative error of MII-PRBK decays the fastest when the running time grows. This is because at each iteration, the GMBC method requires matrix multiplication which involves $4mn^2 + 4m^2n - m^2 - n^2$ flopping operations, whereas the MII-PRBK and MII-RABK methods only cost $8mn + n - 2m$ flops.

Table 1. The numerical results of IT and CPU for $A = randn(m, n)$ from Example 1.

m	n		GBMC	MII-PRBK	MII-RABK
50	1000	IT	29	321.0	812.3
		CPU	0.22	0.05	0.13
50	5000	IT	80	198.4	734.6
		CPU	8.47	0.41	1.40
100	10,000	IT	55	407.5	1398.2
		CPU	27.80	4.35	13.71
1000	50	IT	26	774.7	1092.1
		CPU	0.20	0.15	0.16
5000	50	IT	60	1173.6	1341.4
		CPU	6.36	2.16	2.50
10,000	100	IT	68	2276.4	2637.6
		CPU	34.36	21.13	24.64

Table 2. The numerical results of IT and CPU for $A = sprandn(m/2, n/2)$, $A = [A1, A1; A1, A1]$ from Example 1.

m	n		GBMC	MII-PRBK	MII-RABK
50	1000	IT	29	158.3	410.5
		CPU	0.19	0.02	0.05
50	5000	IT	61	92.3	353.4
		CPU	0.55	0.17	0.17
100	10,000	IT	64	175.7	898.0
		CPU	32.07	1.89	6.64
1000	50	IT	26	554.8	612.4
		CPU	0.19	0.08	0.08
5000	50	IT	62	491.5	593.4
		CPU	6.62	0.94	1.07
10,000	100	IT	56	1040.6	1266.2
		CPU	27.99	9.79	11.23

Example 2. *For given m, n, d, rc, the sparse matrix A is generated by a Matlab built-in function $sprandn(m, n, d, rc)$, with approximately dmn normally distributed nonzero entries. The input parameters d and rc are the percentage of nonzeros and the reciprocal of condition number, respectively. In this example, $A = sprandn(m, n, d, rc)$ or $A1 = sprandn(m/2, n/2, d, rc)$, $A = [A1, A1; A1, A1]$.*

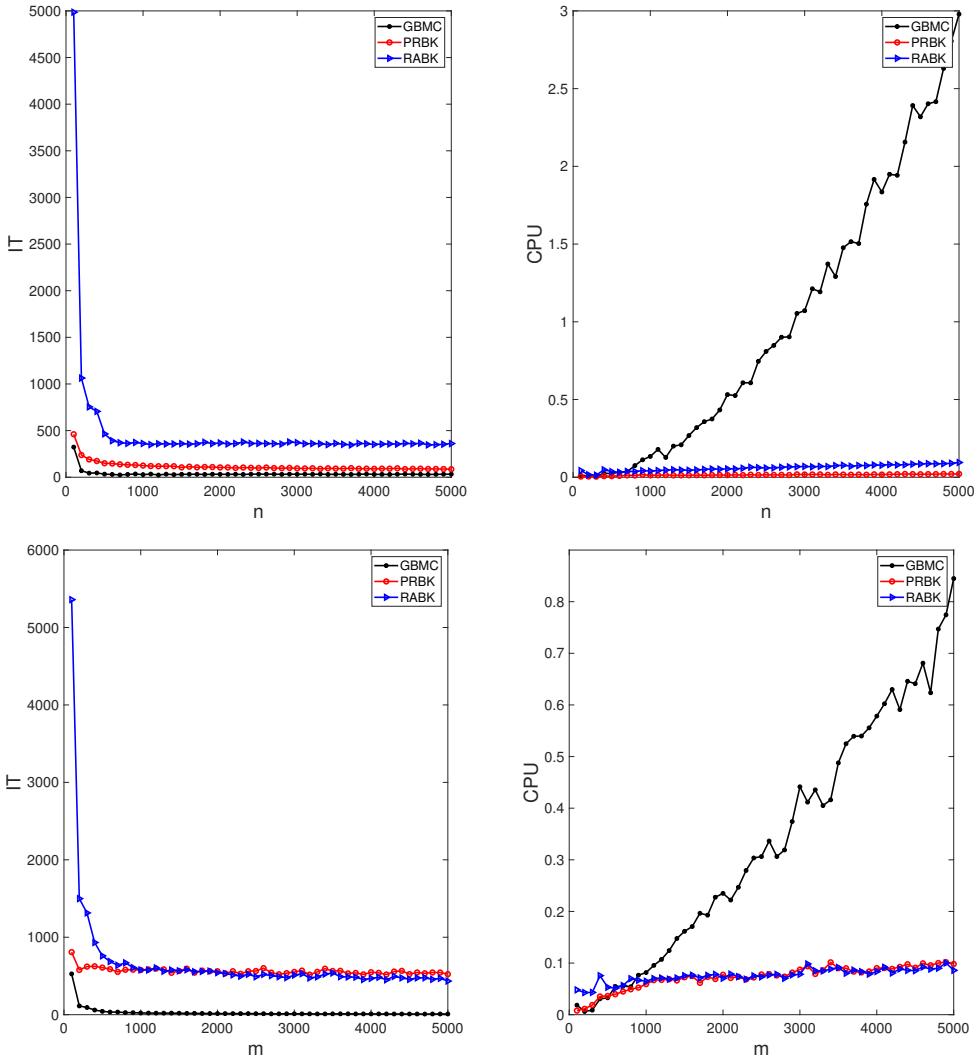

Figure 2. IT (**left**) and CPU (**right**) of different methods with the matrices $A = randn(m, 25)$ (**top**) and $A = randn(25, n)$ (**bottom**) from Example 1.

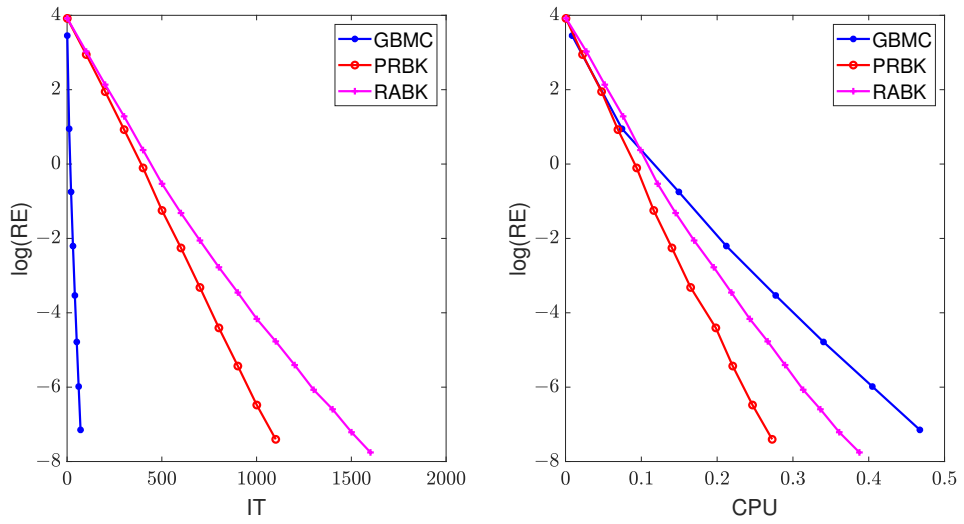

Figure 3. The relative error of the different methods for $A1 = randn(500, 50)$, $A = [A1, A1; A1, A1]$.

In this example, we set $\alpha = \frac{1.9}{\|A\|_2^2}$. The numerical results of IT and CPU are listed in Tables 3 and 4, and the curves of relative error versus IT and CPU are drawn in Figure 4. From Tables 3 and 4, we can observe that the MII-PRBK method has the least iteration steps and running time. The computational efficiency of the MII-PRBK and MII-RABK methods has been improved by at least 33 and 7 times compared to the GBMC method. Moreover, the advantages of the proposed methods are more pronounced when the matrix size increases.

Table 3. The numerical results of IT and CPU for $A = sprandn(m, n, 10\%, 0.1)$ from Example 2.

m	*n*		GBMC	MII-PRBK	MII-RABK
50	1000	IT	3285	605.4	17,184.3
		CPU	17.00	0.09	3.56
100	1000	IT	3294	1617.6	39,508.0
		CPU	22.56	0.39	9.18
50	5000	IT	2984	1337.5	21,342.2
		CPU	321.43	3.12	41.16
1000	50	IT	4271	300.7	6822.5
		CPU	21.47	0.05	1.12
1000	100	IT	4261	741.3	10,456.3
		CPU	26.91	0.79	3.05
5000	50	IT	3079	240.7	8018.8
		CPU	358.35	0.50	15.68

Table 4. The numerical results of IT and CPU for $A = sprandn(m/2, n/2, 10\%, 0.1), A = [A1, A1; A1, A1]$ from Example 2.

m	*n*		GBMC	MII-PRBK	MII-RABK
50	1000	IT	3038	240.2	5321.5
		CPU	15.56	0.04	0.87
100	1000	IT	4010	582.5	15,165.4
		CPU	26.65	0.12	3.54
50	5000	IT	2848	517.9	8767.0
		CPU	334.17	1.17	20.19
1000	50	IT	3780	99.8	3846.0
		CPU	18.96	0.02	0.59
1000	100	IT	3603	218.4	7623.5
		CPU	28.41	0.10	2.55
5000	50	IT	2768	113.0	3930.4
		CPU	339.72	0.24	7.51

Example 3. *Consider dense Toeplitz matrices. For given* m, n, $c = randn(m, 1), r = randn(n, 1)$, $A = toeplitz(c, r)$ *or* $c = randn(m, 1), r = randn(n, 1), A1 = toeplitz(c, r), A = [A1, A1; A1, A1]$.

In this example, we set $\alpha = \frac{1.5}{\|A\|_2^2}$. The numerical results of IT and CPU are listed in Table 5, and the curves of the relative error versus IT and CPU are drawn in Figure 5. We can observe the same phenomenon as that in Example 1.

Example 4. *Consider sparse Toeplitz matrices. For given* m, n, d, $c = sprandn(m, 1, d), r = sprandn(n, 1, d), A = toeplitz(c, r)$ *or* $c = sprandn(m/2, 1, d), r = sprandn(n/2, 1, d), A1 = toeplitz(c, r), A = [A1, A1; A1, A1]$.

In this example, we set $\alpha = \frac{1.5}{\|A\|_2^2}$. The numerical results of IT and CPU are listed in Table 6, and the curves of the relative error versus IT and CPU are drawn in Figure 6. Again, we can draw the same conclusion as that in Example 1.

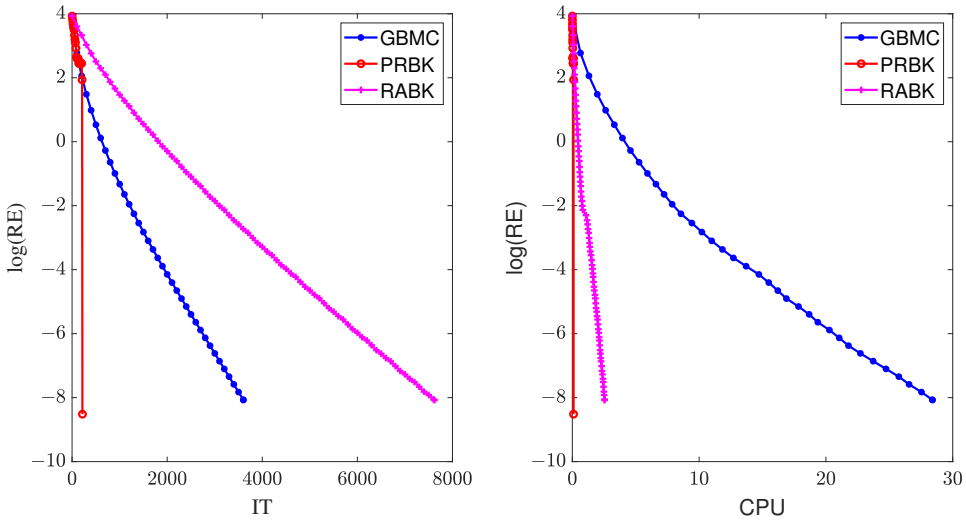

Figure 4. The relative error of the different methods for $A1 = sprandn(50, 500, 10\%, 0.1), A = [A1, A1; A1, A1]$ from Example 2.

Table 5. The numerical results of IT and CPU for $c = randn(m, 1), r = randn(n, 1), A = toeplitz(c, r)$ from Example 3.

m	n		GBMC	MII-PRBK	MII-RABK
50	1000	IT	46	179.5	416.0
		CPU	0.23	0.02	0.06
50	5000	IT	46	125.6	386.3
		CPU	4.87	0.24	0.72
100	10,000	IT	45	276.8	808.3
		CPU	22.34	2.74	8.07
1000	50	IT	45	289.0	719.2
		CPU	0.19	0.04	0.09
5000	50	IT	48	237.6	540.7
		CPU	5.12	0.44	1.11
10,000	100	IT	46	460.5	1286.4
		CPU	22.98	4.36	12.06

Table 6. The numerical results of IT and CPU for $c = sprandn(m, 1, 0.1), r = sprandn(n, 1, 0.1), A = toeplitz(c, r)$ from Example 4.

m	n		GBMC	MII-PRBK	MII-RABK
50	1000	IT	47	180.2	438.6
		CPU	0.19	0.02	0.06
50	5000	IT	47	134.5	405.3
		CPU	4.87	0.23	0.74
100	10,000	IT	45	275.4	805.8
		CPU	21.46	2.73	7.84
1000	50	IT	46	350.5	581.8
		CPU	0.19	0.05	0.07

Table 6. *Cont.*

m	*n*		GBMC	MII-PRBK	MII-RABK
5000	50	IT	49	287.4	580.5
		CPU	5.16	0.55	0.98
10,000	100	IT	43	596.5	1301.4
		CPU	21.46	6.59	12.11

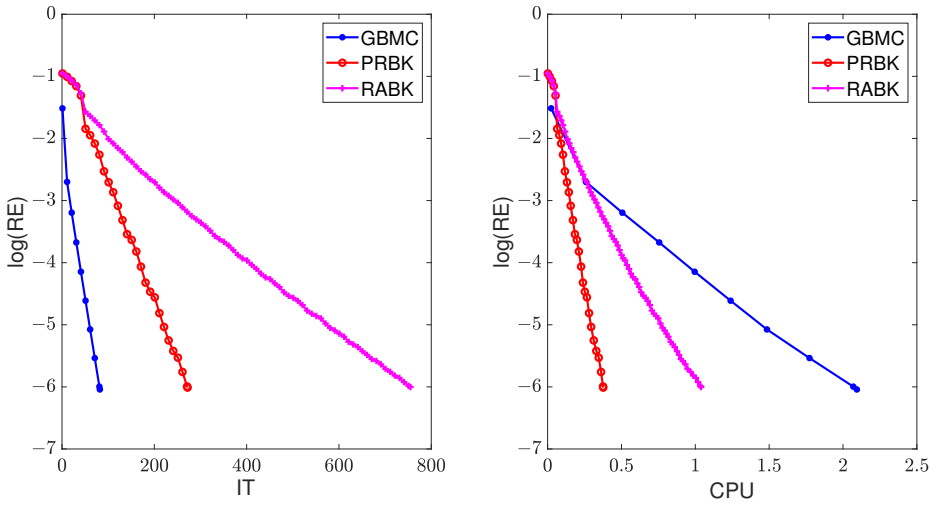

Figure 5. The relative error of the different methods for $c = randn(50,1), r = randn(1000,1), A1 = toeplitz(c,r), A = [A1, A1; A1, A1]$.

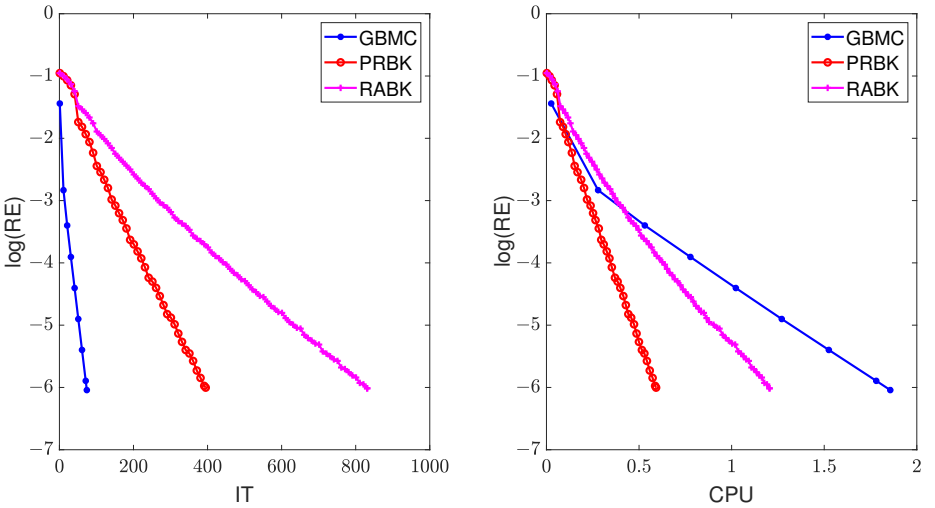

Figure 6. The relative error of the different methods for $c = sprandn(50,1), r = sprandn(1000,1), A1 = toeplitz(c,r), A = [A1, A1; A1, A1]$.

5. Conclusions

In this paper, we have proposed the randomized block Kaczmarz algorithms to compute inner inverses of any rectangle matrix, where A is full rank or rank deficient. Convergence results are provided to guarantee the convergence of the proposed methods theoretically. Numerical examples are given to illustrate the effectiveness. Since the proposed algorithms only require one row of A at each iteration without matrix–matrix product, they are suitable for the scenarios where the matrix A is too large to fit in memory or the matrix multiplication is considerably expensive. In addition, if the MII-RABK method is implemented in parallel, the running time will be greatly reduced. Therefore, in some practical applications, the MII-RABK method is more feasible when the Moore–Penrose

generalized inverse is unknown or the calculation is too expensive. Due to limitations in hardware and personal research areas, we did not perform numerical experiments on very large-scale practical problems, which will become one of our future works. In addition, we will extend the Kaczmarz method to deal with the other generalized inverses of any rectangle matrix. Moreover, providing the feasible principles for selecting parameters and designing a randomized block Kaczmarz algorithm with an adaptive stepsize will be one of our future works.

Author Contributions: Conceptualization, L.X. and W.L.; methodology, L.X. and W.B.; validation, L.X. and W.B.; writing—original draft preparation, L.X.; writing—review and editing, L.X. and W.L.; software, L.X. and Y.L.; visualization, L.X. and Z.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

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

Acknowledgments: The authors are thankful to the referees for their constructive comments and valuable suggestions, which have greatly improved the original manuscript of this paper.

Conflicts of Interest: The authors declare no conflicts of interest. The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Ben-Lsrael, A.; Greville, T.N.E. *Generalized Inverses: Theory and Applications*, 2nd ed.; Canadian Mathematical Society; Springer: NewYork, NY, USA, 2002.
2. Cichocki, A.; Unbehauen, R. *Neural Networks for Optimization and Signal Processing*; John Wiley: West Sussex, UK, 1993.
3. Guo, D.; Xu, F.; Yan, L. New pseudoinverse-based path-planning scheme with PID characteristic for redundant robot manipulators in the presence of noise. *IEEE Trans. Control Syst. Technol.* **2018**, *26*, 2008–2019. [CrossRef]
4. Mihelj, M.; Munih, M.; Podobnik, J. Sensory fusion of magneto-inertial data sensory fusion based on Kinematic model with Jacobian weighted-left-pseudoinverse and Kalman-adaptive gains. *IEEE Trans. Instrum. Meas.* **2019**, *68*, 2610–2620. [CrossRef]
5. Zhang, W.; Wu, J.; Yang, Y.; Akilan, T. Multimodel feature reinforcement framework using Moore-Penrose inverse for big data analysis. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 5008–5021. [CrossRef] [PubMed]
6. Zhuang, H.; Lin, Z.; Toh, K.A. Blockwise recursive Moore-Penrose inverse for network learning. *IEEE Trans. Syst. Man, Cybern. Syst.* **2022**, *52*, 3237–3250. [CrossRef]
7. Sheng, X.; Wang, T. An iterative method to compute Moore-Penrose inverse based on gradient maximal convergence rate. *Filomat* **2013**, *27*, 1269–1276. [CrossRef]
8. Wang, J. Recurrent neural networks for computing pseudoinverses of rank-deficientmatrices. *SIAM J. Sci. Comput.* **1997**, *18*, 1479–1493. [CrossRef]
9. Wei, Y. Recurrent neural networks for computing weighted Moore-Penrose inverse. *Appl. Math. Comput.* **2000**, *116*, 279–287. [CrossRef]
10. Wang, X.Z.; Ma, H.F.; Predrag, S.S. Nonlinearly activated recurrent neural network for computing the drazin inverse. *Neural Process. Lett.* **2017**, *46*, 195–217. [CrossRef]
11. Zhang, N.M.; Zhang, T. Recurrent neural networks for computing the moore-penrose inverse with momentum learning. *Chin. J. Electron.* **2020**, *28*, 1039–1045. [CrossRef]
12. Zhang, Y.; Guo, D.; Li, Z. Common nature of learning between backpropagation and Hopfield-type neural networks for generalized matrix inversion with simplified models. *IEEE Trans. Neural Netw. Learn. Syst.* **2013**. *24*, 579–592. [CrossRef]
13. Stanimirovic, P.; Petkovic, M.; Gerontitis, D. Gradient neural network with nonlinear activation for computing inner inverses and the Drazin inverse. *Neural Process Lett.* **2018**, *48*, 109–133. [CrossRef]
14. Lv, X.; Xiao, L.; Tan, Z.; Yang, Z.; Yuan, J. Improved gradient neural networks for solving Moore-Penrose inverse of full-rank matrix. *Neural Process. Lett.* **2019**, *50*, 1993–2005. [CrossRef]
15. Zhang, Y.; Zhang, J.; Weng, J. Dynamic Moore-Penrose inversion with unknown derivatives: Gradient neural network approach. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *34*, 10919–10929. [CrossRef] [PubMed]
16. Ion, N. Faster randomized block kaczmarz algorithms. *SIAM J. Matrix Anal. Appl.* **2019**, *40*, 1425–1452.
17. Xing, L.L.; Li, W.G.; Bao, W.D. Some results for Kaczmarz method to solve Sylvester matrix equations. *J. Frankl. Inst.* **2023**, *360*, 7457–7461.
18. Du, K.; Ruan, C.C.; Sun, X.H. On the convergence of a randomized block coordinate descent algorithm for a matrix least squaress problem. *Appl. Math. Lett.* **2022**, *124*, 107689. [CrossRef]

19. Du, K.; Si, W.T.; Sun, X.H. Randomized extended average block Kaczmarz for solving least squares. *Siam. J. Sci. Comput.* **2020**, *42*, A3541–A3559. [CrossRef]
20. Strohmer, T.; Vershynin, R. A randomized Kaczmarz algorithm with exponential convergence. *J. Fourier Anal. Appl.* **2009**, *15*, 262–278. [CrossRef]
21. Dax, A. Kaczmarz's anomaly: A surprising feature of Kaczmarz's method. *Linear Algebra Appl.* **2023**, *662*, 136–162. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.