On Augmented Kaczmarz Algorithm for the Solution of Inconsistent Linear Systems

Jingya Wang, Bin Wen, Yong Liu

***Abstract*—The augmented Kaczmarz algorithm is devised to solve an inconsistent linear system by employing the classical Kaczmarz algorithm on a parameterized augmented linear system derived from the original inconsistent linear system. The convergence of the resulting algorithm is proved. Numerical results show that our algorithm exhibits superior performance compared to the greedy randomized augmented Kaczmarz algorithm in relation to the number of iterations and computational durations for ill-conditioned linear inconsistent systems.**

***Index Terms*—Kaczmarz algorithm, inconsistent linear systems, augmented linear system, iteration steps, ill-conditioned.**

I. Introduction

We explore the iterative solution of a large inconsistent linear equations as follows

$$Ax = b, \quad \text{with} \quad A \in \mathbb{R}^{m\times n} \quad \text{and} \quad b \in \mathbb{R}^m. \tag{1}$$

Here, we assume that A possesses full column rank, and $x \in \mathbb{R}^n$ represents the variable to be solved. In the assumption that A is full rank, the linear system (1) always has a unique least squares solution $x_\star = A^\dagger b$, where $A^\dagger$ represents the Moore-Penrose pseudoinverse and this solution also possesses the minimum norm according to [1]. As a well-know fact, it holds true that $Ax_\star = b - r$, where $r \in \mathcal{N}(A^T)$ is a non-zero vector and $\mathcal{N}(\cdot)$ denotes the null space of a matrix.

The well-known iterative Kaczmarz approach [2] is widely used for solving extensive-scale linear equation systems that are either consistent or inconsistent. This method was first proposed by Stefan Kaczmarz, and re-discovered by Gordon et al. [3] using the name of algebraic reconstruction technique. Ever since its inception, the Kaczmarz method has found extensive applications across various domains, including but not limited to statistical analysis [4, 5], machine learning [6], image reconstruction [7–15] and computerized tomograph [16–18]. It is commonly referred to as the row-wise approach due to its focus on processing a solitary row of the matrix during each iteration [19, 20].

The REK, i.e., randomized extended Kaczmarz, algorithm [21] is a highly representative and practical Kaczmarz-type algorithm, initially introduced in 2013 by Zouzias and Freris for linear inconsistent systems; see also [22, 23]. Specifically, the REK algorithm consists of a pair of sub-processes at each iteration: the initial process involves iteratively solving $A^T z = 0$ by means of the randomized Kaczmarz (RK) algorithm [24] to generate a sequence of vectors $\{z_k\}_{k=0}^{\infty}$ that approximates $b_{\mathcal{R}(A)^\perp}$, which represents the orthogonal projection of b onto $\mathcal{N}(A^T)$; the second process involves directly applying the RK algorithm to $Ax = b - z_k$ to determine an approximate solution for $x_\star$. The primary characteristic of REK is to attenuate the influence of the noise in b, ensuring that, $b - z_k$ asymptotically lies within $\mathcal{R}(A)$ as required by the RK algorithm. Experimental simulations conducted in [21] demonstrate that the REK algorithm proves effective for inconsistent linear equations especially for large-scale situations; see for example [21, 25]. Overall, the REK algorithm represents an important randomized numerical iterative algorithm to the field of numerical linear algebra and provides a powerful tool for solving linear inconsistent systems efficiently and accurately. For more comprehensive information on the generalizations of REK, we suggest referring to [26–33] and the pertinent sources cited within.

Recently, a new algorithm called the greedy randomized augmented Kaczmarz (GRAK) algorithm was developed by Bai and Wu [34]. This algorithm is based on the two sub-processes of the REK algorithm and utilizes the greedy randomized Kaczmarz (GRK) algorithm [1] to iterative a consistent augmented linear equations as follows

$$\begin{pmatrix} I & A \\ A^T & O \end{pmatrix} \begin{pmatrix} z \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}, \tag{2}$$

in which $I \in \mathbb{R}^{m\times m}$ is the identity matrix of order m. This system can be derived from (1). The GRAK algorithm exhibits superior stability and efficiency compared to the REK algorithm when solving (1).

Inspired by the idea of Bai and Wu [34], Liu [35] constructed an accelerated randomized augmented Kaczmarz (AGRAK) algorithm for solving (2) by incorporating a positive parameter α into (2) as

$$\begin{pmatrix} \alpha I & A \\ A^T & O \end{pmatrix} \begin{pmatrix} \frac{1}{\alpha} z \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}, \tag{3}$$

However, calculating the residual of (2) at every step is impractical for such large problems in both accelerated and non-accelerated iterative schemes. Hence, we are going to replace the current greedy iterative strategy with a circular iterative approach by the utilization of the Kaczmarz algorithm on the augmented linear system (3). This modification aims to minimize the computational load and reduce the computation time for each iteration in the GRAK algorithm. A maximum limit on the speed at which the resulting algorithm (denoted by AKac, i.e., augmented Kaczmarz) converges is established. With numerical experiments we show that with appropriate choice of the α, the AKac algorithm demonstrates superior

Manuscript received July 17, 2024; revised November 22, 2024.

The work is supported by Natural Science Foundation of the Jiangsu Higher Education Institutions of China (23KJB110001).

Jingya Wang is an undergraduate student of the School of Mathematics and Statistics, Changshu Institute of Technology, Suzhou, Jiangsu, 215500, China (e-mail: wangjingya@cslg.edu.cn).

Bin Wen is an associate professor of the School of Mathematics and Statistics, Changshu Institute of Technology, Suzhou, Jiangsu, 215500, China (e-mail: wenbin@cslg.edu.cn).

Yong Liu is a lecturer of the School of Mathematics and Statistics, Changshu Institute of Technology, Suzhou, Jiangsu, 215500, China (corresponding author, e-mail: liuyong@cslg.edu.cn).

performance compared to the GRAK algorithm as for the total number of iterations and the computational durations.

The layout of this paper is as follows. Section II presents the AKac algorithm and its convergence theory. Section III discusses the numerical results obtained. Finally, in the concluding Section IV, we provide some final remarks.

II. Preliminaries

We will first introduce some mathematical symbols used in this article.

- $\lambda_{\min}(\cdot)$—the minimum nonzero eigenvalue of the corresponding matrix.
- e_j—the jth coordinate basis column vector.
- $\mathcal{R}(A)$—the column space of matrix A.
- $u_{\mathcal{R}(A)}$—the orthogonal projections of the vector u onto $\mathcal{R}(A)$.
- $u_{\mathcal{R}(A)^\perp}$—the orthogonal projections of $u \in \mathbb{R}^n$ onto $\mathcal{R}(A)^\perp$.

Besides, the ith row, jth column, determinant, Euclidean norm and Frobenius norm of a matrix $A \in \mathbb{R}^{m\times n}$ are denoted by $A^{(i)}$, $A_{(j)}$ $\det(A)$, $\|A\|_2$ and $\|A\|_F$, respectively. Similarly, for any $u \in \mathbb{R}^n$, $\|u\|_2$ and $u^{(i)}$ are used to represent its Euclidean norm and ith entry, respectively.

The AKac algorithm can be obtained by using the Kaczmarz method to (3), where α is an arbitrary positive parameter. Moreover, note that the special structures of the coefficient matrix and the right hand side of (3), if the row index i_k at kth iteration satisfies $m+1 \le i_k \le m+n$, then it holds that $x_{k+1} = x_k$. Hence, it is possible to rephrase the aforementioned statement in order to create a more cost-effective version of the AKac algorithm that maintains mathematical equivalence as follows.

Algorithm 1 The AKac Algorithm

Require: $A \in \mathbb{R}^{m\times n}$, $x_0 \in \mathcal{R}(A^T)$, ℓ, $b \in \mathbb{R}^m$, $z_0 \in \mathbb{R}^m$ and $\alpha > 0$.
Ensure: x_ℓ.

1: **for** $k = 0, 1, 2, \cdots, \ell-1$ **do**
2: Select $i_k = (k \bmod (m+n)) + 1$
3: If $1 \le i_k \le m$, set

$$z_{k+1} = z_k + \alpha^2 \frac{b^{(i_k)} - z_k^{(i_k)} - A^{(i_k)}x_k}{\alpha^2 + \|A^{(i_k)}\|_2^2} e_{i_k}$$

and

$$x_{k+1} = x_k + \frac{b^{(i_k)} - z_k^{(i_k)} - A^{(i_k)}x_k}{\alpha^2 + \|A^{(i_k)}\|_2^2} (A^{(i_k)})^T$$

else, set $j_k = i_k - m$,

$$z_{k+1} = z_k - \frac{A_{(j_k)}^T z_k}{\|A_{(j_k)}\|_2^2} A_{(j_k)} \quad \text{and} \quad x_{k+1} = x_k$$

4: **end for**

Regarding the convergence of the AKac algorithm, we present the subsequent theorem.

Theorem 1. *The iterate sequences $\{x_k\}_{k=0}^\infty$ and $\{z_k\}_{k=0}^\infty$ generated by the Algorithm 1 using initial guesses $x_0 \in \mathbb{R}^n$, $z_0 \in \mathbb{R}^m$ and $\alpha > 0$ converge linearly to $x_\star = A^\dagger b$ and $z_\star = b_{\mathcal{R}(A)^\perp}$ in expectation, respectively. Moreover, the solution error for $\{x_k\}_{k=0}^\infty$ and $\{z_k\}_{k=0}^\infty$ fulfills*

$$\left(\|x_k - x_\star\|_2^2 + \frac{1}{\alpha^2}\|z_k - z_\star\|_2^2\right) < \varrho^{\lfloor \frac{k}{m+n} \rfloor} \left(\|x_0 - x_\star\|_2^2 + \frac{1}{\alpha^2}\|z_0 - z_\star\|_2^2\right), \quad (4)$$

where

$$\varrho = 1 - \frac{\alpha^{2(m-n)}(\det(A^TA))^2}{\prod_{i=1}^m (\alpha^2 + \|A^{(i)}\|_2^2) \cdot \prod_{j=1}^n \|A_{(j)}\|_2^2}.$$

Proof: Notice that the vector $(\frac{1}{\alpha}z_\star^T, x_\star^T)^T$ mentioned in Theorem 1 is the solution to (3), as it satisfies the equations

$$\begin{pmatrix} \alpha I & A \\ A^T & O \end{pmatrix} \begin{pmatrix} \frac{1}{\alpha}z_\star \\ x_\star \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}.$$

Next, we will show that $(\frac{1}{\alpha}z_\star^T, x_\star^T)^T$ is also the minimum norm solution in the least squares sense. Assume that $(\frac{1}{\alpha}z_*^T, x_*^T)^T$ is a solution to (3), then, we can obtain

$$z_* + Ax_* = b \quad \text{and} \quad A^T z_* = 0,$$

which shows that $b - z_* \in \mathcal{R}(A)$ and $z_* \in \mathcal{R}(A)^\perp$. Thus,

$$b_{\mathcal{R}(A)^\perp} - z_* = b - b_{\mathcal{R}(A)} - z_* \in \mathcal{R}(A).$$

On the other hand, $b_{\mathcal{R}(A)^\perp} - z_* \in \mathcal{R}(A)^\perp$, which indicates that $b_{\mathcal{R}(A)^\perp} = z_*$, i.e., $z_* = z_\star$. So, we can obtain the equalities

$$Ax_* = b - z_* = b - z_\star = b_{\mathcal{R}(A)}.$$

This shows that there exists a vector $\bar{x} \in \mathcal{R}(A^T)^\perp$ fulfills $x_* = A^\dagger b + \bar{x}$. Hence, it holds that

$$\begin{aligned} \|z_*\|_2 + \|x_*\|_2 &= \|z_\star\|_2 + \|A^\dagger b + \bar{x}\|_2 \\ &= \|z_\star\|_2 + \|A^\dagger b\|_2 + \|\bar{x}\|_2 \\ &= \|z_\star\|_2 + \|x_\star\|_2 + \|\bar{x}\|_2 \\ &\ge \|z_\star\|_2 + \|x_\star\|_2, \end{aligned}$$

which implies that $(\frac{1}{\alpha}z_\star^T, x_\star^T)^T$ is the solution with minimum norm and least-squares error for (3).

Let

$$\tilde{A} = \begin{pmatrix} \alpha I & A \\ A^T & O \end{pmatrix},$$

then following a similar reasoning as in [34, Theorem 3.1], we can concluded that $(\frac{1}{\alpha}z_0^T, x_0^T)^T$ of the AKac algorithm lies within $\mathcal{R}(\tilde{A}^T)$, i.e.,

$$\left(\frac{1}{\alpha}z_0^T, x_0^T\right)^T \in \mathcal{R}(\tilde{A}^T).$$

In fact, setting

$$u = (A^T)^\dagger x_0 + \frac{1}{\alpha}(I - AA^\dagger)z_0$$

and

$$v = \frac{1}{\alpha}A^\dagger z_0 + (A^TA)^\dagger x_0,$$

we can get

$$\tilde{A}^T \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \alpha I & A \\ A^T & O \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{\alpha}z_0 \\ x_0 \end{pmatrix}.$$

As a result, considering Theorem 3.1 mentioned in [26], for $k = 0, 1, 2, \cdots$, we have

$$\left(\|x_k - x_\star\|_2^2 + \frac{1}{\alpha^2}\|z_k - z_\star\|_2^2\right) < \tilde{\varrho}^{\lfloor \frac{k}{m+n} \rfloor}\left(\|x_0 - x_\star\|_2^2 + \frac{1}{\alpha^2}\|z_0 - z_\star\|_2^2\right), \quad (5)$$

where

$$\tilde{\varrho} = 1 - \frac{\alpha^{2(m-n)}\det(\tilde{A}^T\tilde{A})}{\prod_{i=1}^{m+n}\|\tilde{A}^{(i)}\|_2^2}.$$

Based on the following matrix transformation

$$\begin{pmatrix} I & O \\ -\frac{1}{\alpha}A^T & I \end{pmatrix}\begin{pmatrix} \alpha I & A \\ A^T & O \end{pmatrix} = \begin{pmatrix} \alpha I & A \\ O & -\frac{1}{\alpha}A^TA \end{pmatrix},$$

we have

$$\begin{aligned}\det(\tilde{A}) &= \det\begin{pmatrix} \alpha I & A \\ A^T & O \end{pmatrix} \\ &= \det\begin{pmatrix} \alpha I & A \\ O & -\frac{1}{\alpha}A^TA \end{pmatrix} \\ &= (-1)^n\alpha^{m-n}\det(A^TA).\end{aligned}$$

As a result, with the fact that $\tilde{A}^T = \tilde{A}$, we can obtain

$$\det(\tilde{A}^T\tilde{A}) = (\det(\tilde{A}))^2 = \alpha^{2(m-n)}(\det(A^TA))^2. \quad (6)$$

On the other hand, it follows from straightforward computations that

$$\prod_{i=1}^{m+n}\|\tilde{A}^{(i)}\|_2^2 = \prod_{i=1}^{m}\left(\alpha^2 + \|A^{(i)}\|_2^2\right)\cdot\prod_{j=1}^{n}\|A_{(j)}\|_2^2. \quad (7)$$

By substituting (6) and (7) into (5), we can obtain (4), thereby concluding the proof.

III. Numerical Experiments

In this section, we choose two distinct types of matrices to assess the effectiveness of the AKac algorithm when compared to the GRAK algorithm. These two types of matrices are randomly produced by the MATLAB command `randn(m,n)` with varying values for m and n, and the sparse matrices *ash958*, *cities*, *abb313*, *Sandi_authors* and flower_5_1 taken from the University of Florida sparse matrix collection [36]. The vector $b = Ax_* + r$ with x_* being a solution of (1). The non-zero vector r is randomly generated using the MATLAB embedded command `null`. We initialize the vectors x_0 and z_0 to zero, and stop the iterations once the RSE, i.e., the relative solution error, at x_k becomes smaller than 10^{-4}. Here, the RSE is described as

$$\text{RSE} = \frac{\|x_k - x_\star\|_2^2}{\|x_\star\|_2^2} \quad \text{with} \quad x_\star = A^\dagger b.$$

We evaluate the convergence performance of the AKac algorithm and the GRAK algorithm by analyzing the quantity of iteration steps (IT thereafter) and measuring the duration in seconds for computation (CPU thereafter). For GRAK algorithm, the IT and CPU are calculated as the medians of 50 independent executions for the number of iteration steps needed and the CPU durations elapsed. In addition, as referenced in [35], we set $\alpha = \sqrt{\sigma_{min}(A)/2}$ in our numerical experiments. All experiments are conducted on a personal computer with an AMD R7 2.67GHz central processing unit and 16.00 GB memory, utilizing MATLAB (R2022b) as the software platform.

The IT and CPU required for both the AKac and GRAK algorithms to meet the termination criterion while solving Gaussian linear systems are listed in Tables I-III. Additionally, we also offer the comparison of the GRAK algorithm's acceleration compared to that of the AKac algorithm by using the quantity speed-up in these three tables. Here, the speed-up is described as

$$\text{speed-up} = \frac{\text{CPU of GRAK}}{\text{CPU of AKac}}.$$

The purpose of this quantity is to provide insights into the comparative efficiency between the GRAK and AKac algorithms, where a higher speed-up value signifies that the AKac algorithm consumes significantly less CPU time compared to the GRAK algorithm. From Tables I-III, it can be observed that the AKac algorithm demonstrates superior performance in CPU and IT compared to the GRAK algorithm, especially when m is closer to n. In such cases, a higher speed-up indicates that the AKac algorithm reaches the termination criterion faster than its counterpart, such as for $m = 600$ and $n = 500$, the speed-up even attains 382.13. Furthermore, to enhance the visual comprehension of the convergence behavior exhibited by these two algorithms, we plot the curves of the RSE in base-10 logarithm versus IT and CPU for both the GRAK and AKac algorithms with m values of $100, 600$, and 1000 in Figures 1-3. These curves reveal that the AKac algorithm exhibits a significantly faster decrease rate in RSE compared to the GRAK algorithm. This phenomenon becomes even more pronounced as m approaches n. Thus, it can be confirmed that satisfactory outcomes can still be attained without using some greedy strategies.

TABLE I: CPU and IT of AKac and GRAK for matrices $A \in \mathbb{R}^{100\times n}$ with varying n.

$m \times n$		100×70	100×80	100×90
GRAK	IT	50830.0	291338.0	1533303.0
	CPU	1.12	6.58	35.69
AKac	IT	12486.0	32241.0	434.0
	CPU	0.03	0.08	0.15
speed-up		37.33	82.25	237.93

TABLE II: CPU and IT of AKac and GRAK for matrices $A \in \mathbb{R}^{600\times n}$ with varying n.

$m \times n$		600×300	600×400	600×500
GRAK	IT	33616.0	235133.0	3780342.0
	CPU	3.5	29.6	573.2
AKac	IT	19920.0	54434.0	279640.0
	CPU	0.09	0.30	1.50
speed-up		38.89	98.67	382.13

For the five sparse matrices, we tabulated the IT and CPU required for AKac and GRAK in Table IV. However, it is worth mentioning that the GRAK algorithm failed to meet the stopping criterion after 1,000,000 iterations for matrix *cities*. From Table IV, we see that the AKac algorithm exhibits superior performance compared to the GRAK

TABLE III: CPU and IT of AKac and GRAK for matrices $A \in \mathbb{R}^{1000\times n}$ with varying n.

$m \times n$		1000×500	1000×600	1000×700
GRAK	IT	52175.0	153335.0	494853.0
	CPU	12.1	43.8	168.5
AKac	IT	35206.0	54631.0	111018.0
	CPU	0.24	0.44	0.89
speed-up		50.42	99.55	189.33

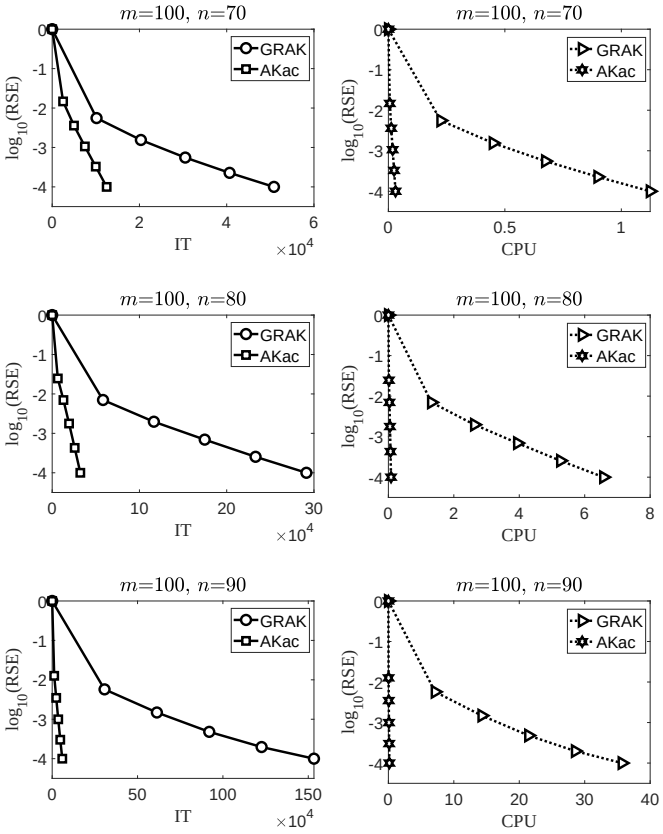

Fig. 1. Picture of the trend of $\log_{10}$(RSE) with changes in CPU and IT for AKac and GRAK for random matrices $A \in \mathbb{R}^{100\times n}$ with and $n = 70, 80, 90$.

algorithm in terms of both CPU and IT, and the speed-up even attains 708.13. To view the convergence behavior of these two algorithms visually, we additionally generate graphs illustrating the relationship between the RSE (in base-10 logarithm) and IT as well as CPU in Figure 4. As the curves in Figure 4 show that, the relative solution error curve in base-10 logarithm of the AKac algorithm in each subplot decrease more fiercely than that of the GRAK algorithm.

IV. Conclusion

By applying the Kaczmarz algorithm to the augmented linear system that can be deduced from the REK iterative process, we have obtained an augmented Kaczmarz algorithm to address inconsistent linear equation systems and established the convergence theory for the resulting algorithm. Numerical experiments show that our algorithm exhibits superior performance compared to the GRAK algorithm in both CPU times and iteration steps if an appropriate positive parameter α is conveniently available.

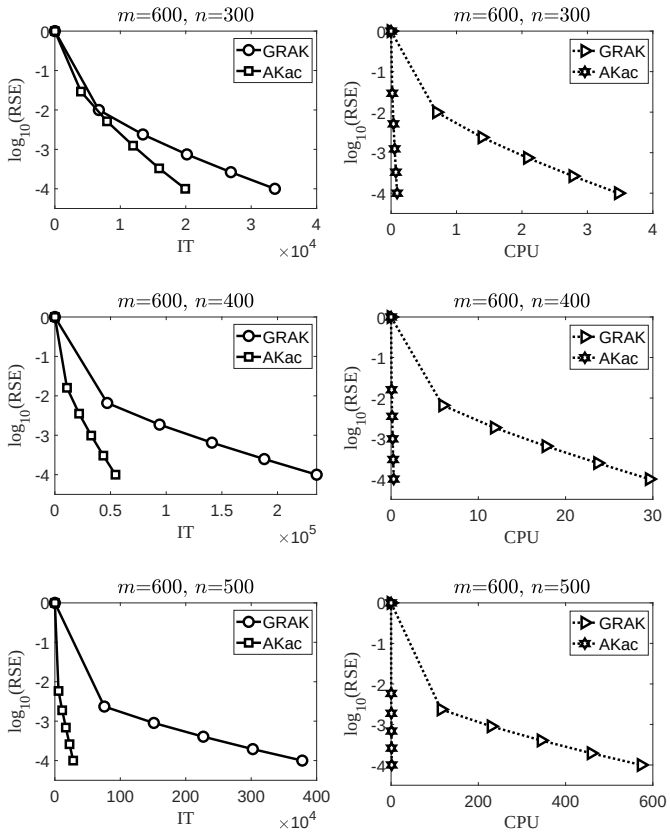

Fig. 2. Picture of the trend of $\log_{10}$(RSE) with changes in CPU and IT for AKac and GRAK for random matrices $A \in \mathbb{R}^{600\times n}$ with $n = 300, 400, 500$.

References

[1] Z. Z. Bai, W. T. Wu, "On greedy randomized Kaczmarz method for solving large sparse linear systems," *SIAM J. Sci. Comput.,* vol. 40, no. 1, pp. A592-A606, 2018.

[2] S. Kaczmarz, "Angenäherte auösung von aystemen linearer gleichungen," *Bull. Int. Acad. Polon. Sci. Lett. A,* vol. 35, pp. 355-357, 1937.

[3] R. Gordon, R. Bender, and G. T. Herman, "Algebraic reconstruction techniques for 3 dimensional electron microscopy and X-ray photograph," *J. Theoret. Biol.,* vol. 29, no. 3, pp. 471-481, 1970.

[4] Z. S. Ye, J. G. Li, and M. Zhang, "Application of ridge regression and factor analysis in design and production of alloy wheels," *J. Appl. Stat.,* vol. 41, no. 7-8, pp. 1436-1452, 2014.

[5] A. Hadgu, "An application of ridge regression analysis in the study of syphilis data," *Stat. Med.,* vol. 3, no. 3, pp. 293-299, 2010.

[6] T. Zhang, "On the dual formulation of regularized linear systems with convex risks," *Mach. Learn.,* vol. 46, no. 1, pp. 91-129, 2002.

[7] J. J. Cui, G. H. Peng, Q. Lu, and Z. G. Huang, "A class of nonstationary upper and lower triangular splitting iteration methods for ill-posed inverse problems," *IAENG, Int. J. Comput. Sci.*, vol. 47, no. 1, pp. 118-129, 2020.

[8] Y. Wang, J. C. Wang, F. J. Li, Y. C. Zhou, M. Xiong, and M. Li, "Face image super-resolution reconstruction via mapping matrix and multilayer model," *IAENG, Int. J. Comput. Sci.*, vol. 47, no. 1, pp. 68-74, 2020.

[9] P. C. Hansen, "Regularization tools version 4.0 for MATLAB 7.3," *Numer. Algorithms,* vol. 46, no. 2, pp. 189-194, 2007.

[10] P. C. Hansen, "Discrete Inverse Problems: Insight and Algorithms," Philadelphia, PA: SIAM, 2010.

[11] P. C. Hansen, J. S. Jørgensen, "AIR Tools II: algebraic iterative reconstruction methods, improved implementation," *Numer. Algorithms,* vol. 79, no. 1, pp. 107-137, 2018.

[12] Y. Liu, C. Q. Gu, "Two greedy subspace Kaczmarz algorithm for image reconstruction," *IAENG, Int. J. Appl. Math.,* vol. 50, no. 4, pp. 853-859, 2020.

[13] K. S. Sim, F. F. Ting, J. W. Leong, and C. P. Tso, "Signal-to-noise ratio estimation for SEM single image using cubic spline interpolation with linear least square regression," *Eng. Lett.*, vol. 27, no. 1, pp. 151-165, 2019.

TABLE IV: CPU and IT of AKac and GRAK for sparse matrices $A \in \mathbb{R}^{m\times n}$.

name		ash958	cities	abb313	Sandi_ authors	flower_5_1
$m \times n$		958×292	55×46	313×176	86×86	211×201
density		0.68%	53.04%	2.83%	3.35%	1.42%
cond(A)		3.20	207.15	1.82×10^{18}	1.78×10^{18}	2.0×10^{16}
GRAK	IT	3867.0	1000000.0	682345.5	332625.0	143506.0
	CPU	0.58	152.3	432.3	6.58	56.65
AKac	IT	12889.0	247777.0	23146.0	71966.0	27629.0
	CPU	0.08	0.49	1.45	0.17	0.08
speed-up		7.25	310.81	298.13	38.71	708.13

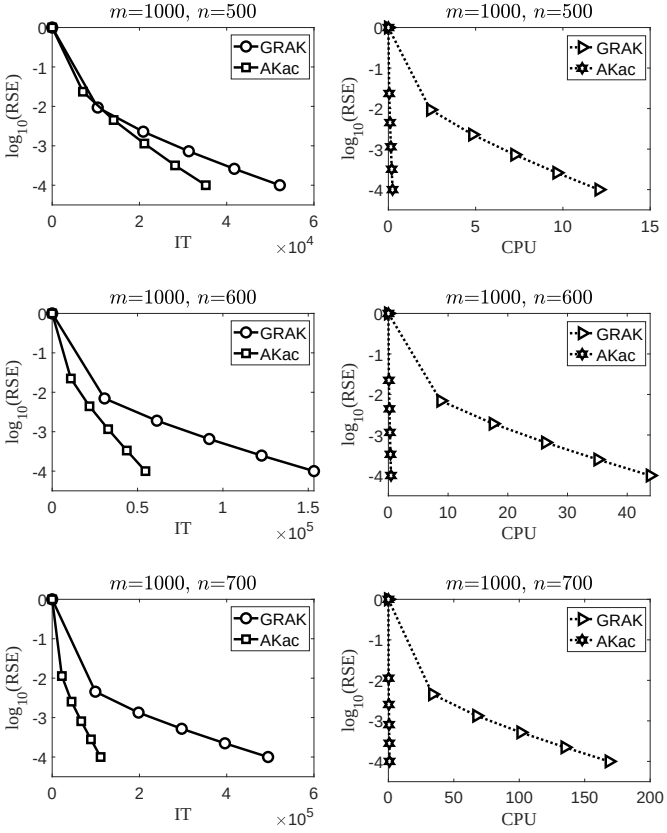

Fig. 3. Picture of the trend of $\log_{10}$(RSE) with changes in CPU and IT for AKac and GRAK for random matrices $A \in \mathbb{R}^{1000\times n}$ with $n = 500, 600, 700$.

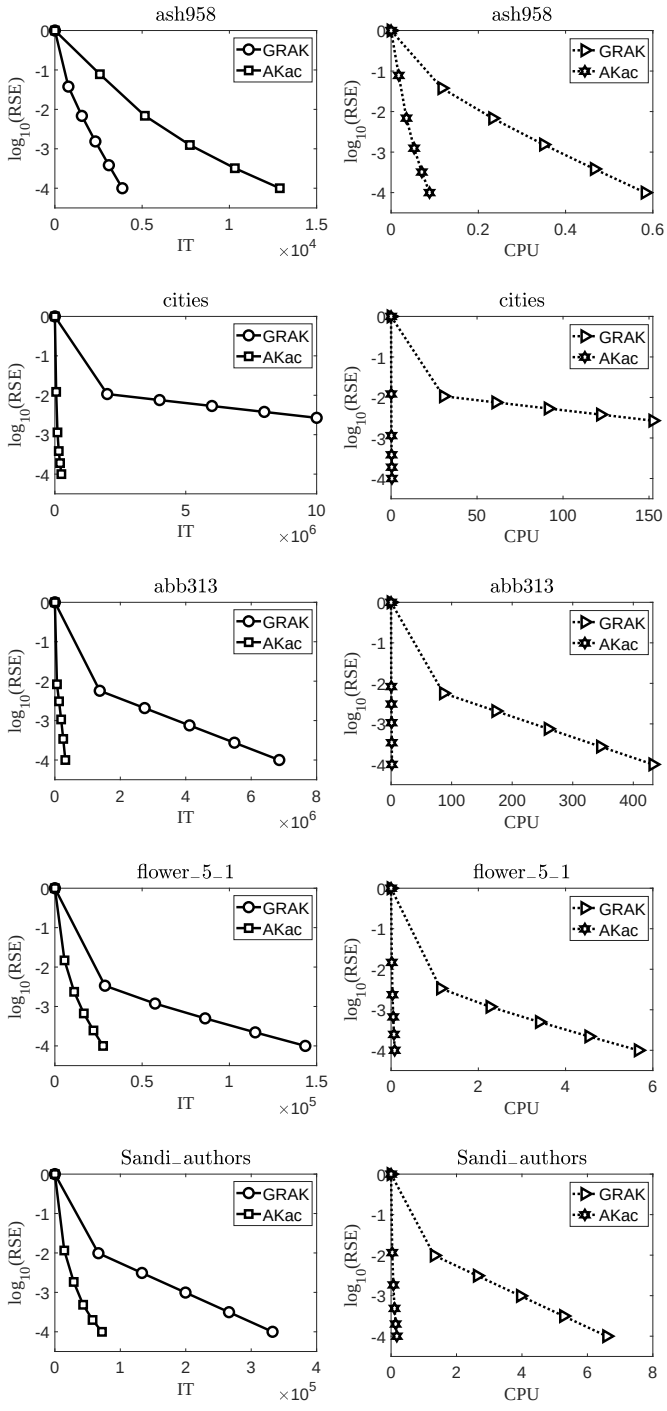

Fig. 4. Picture of the trend of $\log_{10}$(RSE) with changes in CPU and IT for AKac and GRAK on sparse matrices.

[14] M. X. Chen, B. X. Liu, D. S. Zeng, and W. Gao, "A framework for ontology-driven similarity measuring using vector learning tricks," *Eng. Lett.*, vol. 27, no. 3, pp. 549-558, 2019.

[15] G. T. Herman, L. B. Meyer, "Algebraic reconstruction techniques can be made computationally efficient," *IEEE Trans. Med. Imaging,* vol. 12, no. 3, pp. 600-609, 1993.

[16] G. T. Herman, "Fundamentals of Computerized Tomography: Image Reconstruction from Projections, 2nd. ed," New York: Springer, 2009.

[17] A. C. Kak, M. Slaney, "Principles of Computerized Tomographic Imaging," Philadelphia, PA: SIAM, 2001.

[18] F. Natterer, "The Mathematics of Computerized Tomography," Philadelphia, PA: SIAM, 2001.

[19] Y. Censor, "Row-action methods for huge and sparse systems and their applications," *SIAM Rev.,* vol. 23, no. 4, pp. 444-466, 1981.

[20] P. C. Hansen, M. Saxild-Hansen, "AIR Tools-A MATLAB package of algebraic iterative reconstruction methods," *J. Comput. Appl. Math.,* vol. 236, no.8, pp. 2167-2178, 2012.

[21] A. Zouzias, N. M. Freris, "Randomized extended Kaczmarz for solving least squares," *SIAM J. Matrix Anal. Appl.* vol. 34, no. 2, pp. 773-793, 2013.

[22] C. Popa, "Least-squares solution of overdetermined inconsistent linear systems using Kaczmarz's relaxation," *Intern. J. Comput. Math.,*

vol. 55, no. 1-2, pp. 79-89, 1995.
[23] C. Popa, "Extensions of block-projections methods with relaxation parameters to inconsistent and rank-deficient least-squares problems," *BIT,* vol. 38, no. 1, pp. 151-176, 1998.
[24] T. Strohmer, R. Vershynin, "A randomized Kaczmarz algorithm with exponential convergence," *J. Fourier Anal. Appl.,* vol. 15, no. 2, pp. 262-278, 2009.
[25] 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 ,* vol. 58, no. 2, pp. 1-32, 2021.
[26] Z. Z. Bai, W. T. Wu, "On partially randomized extended Kaczmarz method for solving large sparse overdetermined inconsistent linear systems," *Linear Algebra Appl.,* vol. 578, pp. 225-250, 2019.
[27] B. Dumitrescu, "On the relation between the randomized extended Kaczmarz algorithm and coordinate descent," *BIT,* vol. 55, no.4, pp. 1005-1015, 2015.
[28] K. Du, "Tight upper bounds for the convergence of the randomized extended Kaczmarz and Gauss-Seidel algorithms," *Numer. Linear Algebra Appl.,* vol. 26, no. 3, pp. e2233, 2019.
[29] K. Du, W. T. Si, and X. H. Sun, "Randomized extended average block Kaczmarz for solving least squares,"*SIAM J. Sci. Comput.,* vol. 42, no. 6, pp. A3541-A3559, 2020.
[30] W. T. Wu, "On two-subspace randomized extended Kaczmarz method for solving large linear least-squares problems,"*Numerical Algorithms,* vol. 89, no. 1, pp. 1-31, 2021.
[31] W. D. Bao, Z. L. Lv, F. Y. Zhang, and W. G. Li, "A class of residual-based extended Kaczmarz methods for solving inconsistent linear systems," *J. Comput. Appl. Math.,* vol. 416 pp. 114529, 2022.
[32] F. Schöpfera, D. A. Lorenz, L. Tondji, and M. Winkler, "Extended randomized Kaczmarz method for sparse least squares and impulsive noise problems," *Linear Algebra Appl.* vol. 652, pp. 132-154, 2022.
[33] Z. L. Lv, W. D. Bao, W. G. Li, F. Wang, and G. L. Wu, "On extended Kaczmarz methods with random sampling and maximum-distance for solving large inconsistent linear systems," *Results Appl. Math.,* vol. 13, pp. 100240, 2022.
[34] Z. Z. Bai, W. T. Wu, "On greedy randomized augmented Kaczmarz method for solving large sparse inconsistent linear systems," *SIAM J. Sci. Comput.,* vol. 43, no. 6, pp. A3892-A3911, 2021.
[35] Y. Liu, "Accelerated greedy randomized augmented Kaczmarz algorithm for inconsistent linear systems," *Appl. Numer. Math.* vol. 195, pp. 142-156, 2024.
[36] T. A. Davis, Y. Hu, "The University of Florida sparse matrix collection," *ACM Trans. Math. Software,* vol. 38, no. 1, pp. 1-25, 2011.