Journal of Applied Mathematics and Physics, 2021, 9, 2954-2965
https://www.scirp.org/journal/jamp
ISSN Online: 2327-4379
ISSN Print: 2327-4352

A Geometric Gaussian Kaczmarz Method for Large Scaled Consistent Linear Equations

Li Wen[1], Feng Yin[1,2*], Yimou Liao[1], Guangxin Huang[2,3]

[1]College of Mathematics and Statistics, Sichuan University of Science and Engineering, Zigong, China
[2]Sichuan Province University Key Laboratory of Bridge Non-Destruction Detecting and Engineering Computing, Zigong, China
[3]College of Mathematics and Physics, Chengdu University of Technology, Chengdu, China
Email: *fyinsuse@163.com, zeliwen@163.com, liaoyimou596@163.com, huangx@cdut.edu.cn

How to cite this paper: Wen, L., Yin, F., Liao, Y.M. and Huang, G.X. (2021) A Geometric Gaussian Kaczmarz Method for Large Scaled Consistent Linear Equations. *Journal of Applied Mathematics and Physics*, **9**, 2954-2965.
https://doi.org/10.4236/jamp.2021.911189

Received: October 29, 2021
Accepted: November 23, 2021
Published: November 26, 2021

Abstract

This paper presents a geometric Gaussian Kaczmarz (GGK) method for solving the large-scaled consistent linear systems of equation. The GGK method improves the geometric probability randomized Kaczmarz method in [1] by introducing a new block set strategy and the iteration process. The GGK is proved to be of linear convergence. Several numerical examples show the efficiency and effectiveness of the GGK method.

Keywords

Gaussian Kaczmarz, Convergence Rate, Linear Systems, Consistent

1. Introduction

We are concerned with the approximation solution of large-scaled linear systems of equation of the form

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

where $A \in \mathbb{R}^{m\times n}$ is a real matrix, $b \in \mathbb{R}^m$ is a real vector and $x \in \mathbb{R}^n$ is an unknown vector to be determined.

The Kaczmarz [2] method is a good candidate for solving large problems (1) due to its simplicity and good performance, which is applied in numerous fields, such as image reconstruction [3] [4] and digital signal processing [5]. The Kaczmarz method can be formulated as

$$x_{k+1} = x_k + \frac{b^{(i)} - A^{(i)}x_k}{\left\|A^{(i)}\right\|_2^2}\left(A^{(i)}\right)^{\mathrm{T}} \tag{2}$$

where $i = (k \bmod m) + 1$, *i.e.* all m equations in the linear systems (1) are swept through after m iterations.

There are many extended Kaczmarz methods are derived in recent years. Strohmer and Vershynin in [6] proposed a randomized Kaczmarz (RK) method with the expected exponential rate of convergence, which is also known as "linear convergence". In 2018, Bai and Wu [7] proposed a greedy randomized Kaczmarz (GRK) method. GRK introduces an effective probability criterion to grasp larger entries of the residual vector at each iteration and was proved to be of the linear convergence rate. The index set is determined by

$$u_k = \left\{ i \,|\, \left| b^{(i)} - A^{(i)} x_k \right|^2 \geq \varepsilon_k \left\| b - A x_k \right\|_2^2 \left\| A^{(i)} \right\|_2^2 \right\}, \tag{3}$$

where

$$\varepsilon_k = \frac{1}{2}\left(\frac{1}{\left\| b - A x_k \right\|_2^2} \max_{1 \leq i \leq m} \left\{ \frac{\left| b^{(i)} - A^{(i)} x_k \right|^2}{\left\| A^{(i)} \right\|_2^2} \right\} + \frac{1}{\left\| A \right\|_F^2} \right). \tag{4}$$

In order to improve the influence on the convergence rate by diagonal scalings of the coefficient matrix in (1). Yang in [1] presented a geometric probability randomized Kaczmarz algorithm (GPRK) to determine a subset J_k of $[m] = \{1, 2, \cdots, m\}$ by a given rule such that the magnitude of the value of $\left(d_i^k\right)^2$ should be larger than a prescribed threshold if the row index $i \in J_k$, where

$$d_i^k = \left\| \frac{b^{(i)} - A^{(i)} x_k}{\left\| A^{(i)} \right\|_2^2} \left(A^{(i)} \right)^{\mathrm{T}} \right\|. \tag{5}$$

Generally, the block Kaczmarz methods [8] [9] select some rows from the coefficient matrix A to construct the block matrix and compute the Moore-Penrose pseudoinverse of the block at each iteration. However, the cost of computing the Moore-Penrose pseudoinverse of a matrix is so expensive, which impacts gravely the CPU time of the algorithm.

Gower and Richtrik proposed a Gaussian Kaczmarz (GK) method [10] whose iteration process is defined by

$$x_{k+1} = x_k + \frac{\zeta_k^{\mathrm{T}} \left(b - A x_k \right)}{\left\| A^{\mathrm{T}} \zeta_k \right\|_2^2} A^{\mathrm{T}} \zeta_k, \tag{6}$$

where ζ_k is a Gaussian vector with mean $0 \in \mathbb{R}^m$ and the covariance matrix $I \in \mathbb{R}^{m \times m}$, *i.e.*, $\zeta_k \sim N(0, I)$. Here I denotes the identity matrix. The expected linear convergence rate was analyzed in the case that A is of full column rank. The idea of the GK method is also used in [11] to avoid computing the pseudoinverses of submatrices. We will adapt this iteration process in our method.

In this paper, we improve the GPRK method in [1] and present a geometric Gaussian Kaczmarz (GGK) method. The GGK introduces a new block strategy and uses the iteration process of the GK method in (6). The block set strategy of GGK is more efficient than that in [1] because it can grasp as many as possible rows of the system matrix to participate in projection. This is illustrated in Section 3.

The rest of this paper is arranged as follows. A geometric Gaussian Kaczmarz

algorithm is presented in Section 2. Its convergence is also proved. Section 3 shows several numerical examples for the proposed method and Section 4 draws some conclusions.

2. A Geometric Gaussian Kaczmarz Algorithm

This section describes a geometric Gaussian Kaczmarz (GGK) algorithm to compute the solution of (1). **Algorithm 1** summarizes the GGK algorithm. Steps 2, 3 and 4 determine the block control sequence $\{\tau_k\}_{k\geq 0}$, which is simpler than and different from that in [1], which determines the block control sequence by probability

$$Pr\left(row = ik\right) = \frac{\max\left(d_i^k\right)^2}{\sum_{i=1}^{m}\left(d_i^k\right)^2}.$$

Steps 5 and 6 give the iteration process of the GGK method.

The following results show the convergence of **Algorithm 1**.

Theorem 1. Assume the linear system (1) is consistent, and then the iterative sequence $\{x_k\}_{k\geq 0}$ generated by **Algorithm 1** converges to the least-norm solution $x^* = A^{\dagger}b$ of linear systems (1). Moreover, the solution error of linear systems (1) satisfies

$$\left\|x_{k+1} - x^*\right\|_2^2 \leq \left(1 - \eta\frac{\left\|A_{\tau_k}\right\|_F^2}{\lambda_{\max}\left(A_{\tau_k}A_{\tau_k}^{\mathrm{T}}\right)}\frac{\lambda_{\min}\left(A^{\mathrm{T}}A\right)}{\|A\|_F^2}\right)\left\|x_k - x^*\right\|_2^2. \tag{7}$$

Proof. According to **Algorithm 1**, we have

$$\begin{aligned} x_{k+1} - x^* &= x_k - x^* + \frac{\zeta_k^{\mathrm{T}}\left(b - Ax_k\right)}{\left\|A^{\mathrm{T}}\zeta_k\right\|_2^2}A^{\mathrm{T}}\zeta_k \\ &= x_k - x^* - \frac{\left(A^{\mathrm{T}}\zeta_k\right)^{\mathrm{T}}\left(x_k - x^*\right)}{\left\|A^{\mathrm{T}}\zeta_k\right\|_2^2}A^{\mathrm{T}}\zeta_k \\ &= \left(I - \frac{\left(A^{\mathrm{T}}\zeta_k\right)^{\mathrm{T}}A^{\mathrm{T}}\zeta_k}{\left\|A^{\mathrm{T}}\zeta_k\right\|_2^2}\right)\left(x_k - x^*\right). \end{aligned}$$

Algorithm 1. A geometric Gaussian Kaczmarz algorithm (GGK).

Data: $A, b, x_0, \eta \in (0,1]$, maximum iterations IterMax

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

2 Determine $d_i^k = \|\frac{b^{(i)} - A^{(i)}x_k}{\|A^{(i)}\|_2^2}(A^{(i)})^T\|$.

3 Compute $\epsilon_k = \eta\frac{\max\limits_{1\leq i\leq m}(d_i^k)^2}{\sum_{i=1}^{m}(d_i^k)^2}$.

4 Determine the control index set $\tau_k = \{i_k \mid (d_{i_k}^k)^2 \geq \epsilon_k \sum_{i=1}^{m}(d_i^k)^2\}$.

5 Compute $\zeta_k = \sum\limits_{i\in\tau_k}(b^{(i)} - A^{(i)}x_k)e_i$.

6 Update $x_{k+1} = x_k + \frac{\zeta_k^T(b - Ax_k)}{\|A^T\zeta_k\|_2^2}A^T\zeta_k$.

7 **end**

Denote the projector $P_k = \dfrac{\left(A^{\mathrm{T}}\zeta_k\right)^{\mathrm{T}} A^{\mathrm{T}}\zeta_k}{\left\|A^{\mathrm{T}}\zeta_k\right\|_2^2}$, then P_k is orthogonal because $P_k^{\mathrm{T}} = P_k$ and $P_k^2 = P_k$. Thus we have

$$\begin{aligned}\left\|x_{k+1} - x^*\right\|_2^2 &= \left\|\left(I - \frac{\left(A^{\mathrm{T}}\zeta_k\right)^{\mathrm{T}} A^{\mathrm{T}}\zeta_k}{\left\|A^{\mathrm{T}}\zeta_k\right\|_2^2}\right)\left(x_k - x^*\right)\right\|_2^2 \\ &= \left\|x_k - x^*\right\|_2^2 - \left\|\frac{\zeta_k^{\mathrm{T}}\left(b - Ax_k\right)}{\left\|A^{\mathrm{T}}\zeta_k\right\|_2^2} A^{\mathrm{T}}\zeta_k\right\|_2^2 \\ &= \left\|x_k - x^*\right\|_2^2 - \frac{\left|\zeta_k^{\mathrm{T}}\left(b - Ax_k\right)\right|^2}{\left\|A^{\mathrm{T}}\zeta_k\right\|_2^2} \\ &= \left\|x_k - x^*\right\|_2^2 - \frac{\left\|\zeta_k\right\|_2^4}{\left\|A^{\mathrm{T}}\zeta_k\right\|_2^2}.\end{aligned}$$

The last equality holds because

$$\zeta_k^{\mathrm{T}}\left(b - Ax_k\right) = \sum_{i\in\tau_k}\left(b^{(i)} - A^{(i)}x_k\right)e_i^{\mathrm{T}}\left(b - Ax_k\right) = \sum_{i\in\tau_k}\left|b^{(i)} - A^{(i)}x_k\right|^2 = \left\|\zeta_k\right\|_2^2.$$

Let $E_k \in \mathbb{R}^{m\times|\tau_k|}$ denote the matrix whose columns orderly are constituted of all the vector $e_i \in \mathbb{R}^m$ with $i \in \tau_k$, then $A_{\tau_k} = E_k^{\mathrm{T}} A$. Denoted by $\hat{\zeta}_k = E_k^{\mathrm{T}}\zeta_k$, we have

$$\left\|A^{\mathrm{T}}\zeta_k\right\|_2^2 = \zeta_k^{\mathrm{T}} AA^{\mathrm{T}}\zeta_k = \hat{\zeta}_k^{\mathrm{T}} E_k^{\mathrm{T}} AA^{\mathrm{T}} E_k\hat{\zeta}_k = \hat{\zeta}_k^{\mathrm{T}} A_{\tau_k} A_{\tau_k}^{\mathrm{T}}\hat{\zeta}_k = \left\|A_{\tau_k}^{\mathrm{T}}\hat{\zeta}_k\right\|_2^2,$$

moreover,

$$\left\|A^{\mathrm{T}}\zeta_k\right\|_2^2 \le \lambda_{\max}\left(A_{\tau_k}A_{\tau_k}^{\mathrm{T}}\right)\left\|\hat{\zeta}_k\right\|_2^2 = \lambda_{\max}\left(A_{\tau_k}A_{\tau_k}^{\mathrm{T}}\right)\left\|\zeta_k\right\|_2^2.$$

It then holds that

$$\begin{aligned}\left\|x_{k+1} - x^*\right\|_2^2 &\le \left\|x_k - x^*\right\|_2^2 - \frac{\left\|\zeta_k\right\|_2^2}{\lambda_{\max}\left(A_{\tau_k}A_{\tau_k}^{\mathrm{T}}\right)} \\ &= \left\|x_k - x^*\right\|_2^2 - \frac{\sum_{i\in\tau_k}\left|b^{(i)} - A^{(i)}x_k\right|^2}{\lambda_{\max}\left(A_{\tau_k}A_{\tau_k}^{\mathrm{T}}\right)} \\ &= \left\|x_k - x^*\right\|_2^2 - \frac{\sum_{i\in\tau_k}\left(d_i^k\right)^2\left\|A^{(i)}\right\|_2^2}{\lambda_{\max}\left(A_{\tau_k}A_{\tau_k}^{\mathrm{T}}\right)} \\ &\le \left\|x_k - x^*\right\|_2^2 - \frac{\eta\max_{i\in\tau_k}\left(d_i^k\right)^2\sum_{i\in\tau_k}\left\|A^{(i)}\right\|_2^2}{\lambda_{\max}\left(A_{\tau_k}A_{\tau_k}^{\mathrm{T}}\right)} \\ &= \left\|x_k - x^*\right\|_2^2 - \frac{\eta\max_{i\in\tau_k}\left(d_i^k\right)^2\left\|A_{\tau_k}\right\|_F^2}{\lambda_{\max}\left(A_{\tau_k}A_{\tau_k}^{\mathrm{T}}\right)}.\end{aligned}$$

For each $k \geq 0$, since

$$\max_{i\in\tau_k}\left(d_i^k\right)^2 = \max_{i\in\tau_k}\frac{\left|b^{(i)}-A^{(i)}x_k\right|^2}{\left\|A^{(i)}\right\|_2^2} \geq \sum_{i=1}^{m}\frac{\left\|A^{(i)}\right\|_2^2}{\|A\|_F^2}\frac{\left|b^{(i)}-A^{(i)}x_k\right|^2}{\left\|A^{(i)}\right\|_2^2} = \frac{\|b-Ax_k\|_2^2}{\|A\|_F^2},$$

we have

$$\begin{aligned}\left\|x_{k+1}-x^*\right\|_2^2 &\leq \left\|x_k-x^*\right\|_2^2 - \frac{\eta\frac{\|b-Ax_k\|_2^2}{\|A\|_F^2}\left\|A_{\tau_k}\right\|_F^2}{\lambda_{\max}\left(A_{\tau_k}A_{\tau_k}^{\mathrm{T}}\right)}\\ &\leq \left\|x_k-x^*\right\|_2^2 - \eta\frac{\lambda_{\min}\left(A^{\mathrm{T}}A\right)}{\lambda_{\max}\left(A_{\tau_k}A_{\tau_k}^{\mathrm{T}}\right)}\frac{\left\|A_{\tau_k}\right\|_F^2}{\|A\|_F^2}\left\|x_k-x^*\right\|_2^2\\ &= \left(1-\eta\frac{\lambda_{\min}\left(A^{\mathrm{T}}A\right)}{\lambda_{\max}\left(A_{\tau_k}A_{\tau_k}^{\mathrm{T}}\right)}\frac{\left\|A_{\tau_k}\right\|_F^2}{\|A\|_F^2}\right)\left\|x_k-x^*\right\|_2^2.\end{aligned}$$

This completes the proof. □

We remark that

$$0 \leq 1-\eta\frac{\lambda_{\min}\left(A^{\mathrm{T}}A\right)}{\lambda_{\max}\left(A_{\tau_k}A_{\tau_k}^{\mathrm{T}}\right)}\frac{\left\|A_{\tau_k}\right\|_F^2}{\|A\|_F^2} \leq 1,$$

which means that **Algorithm 1** is convergent. In fact,

$$\frac{\left\|A_{\tau_k}\right\|_F^2}{\lambda_{\max}\left(A_{\tau_k}A_{\tau_k}^{\mathrm{T}}\right)} \geq 1,$$

and

$$0 < \frac{\lambda_{\min}\left(A^{\mathrm{T}}A\right)}{\|A\|_F^2} \leq 1,$$

it follows that

$$1-\eta\frac{\left\|A_{\tau_k}\right\|_F^2}{\lambda_{\max}\left(A_{\tau_k}A_{\tau_k}^{\mathrm{T}}\right)}\frac{\lambda_{\min}\left(A^{\mathrm{T}}A\right)}{\|A\|_F^2} \leq 1-\eta\frac{\lambda_{\min}\left(A^{\mathrm{T}}A\right)}{\|A\|_F^2} < 1.$$

3. Numerical Experiments

In this section, we use **Algorithm 1** for solving different types of consistent linear systems (1) and compare it with GPRK in [1]. All experiments are carried out using MATLAB (version R2019b) on a personal computer with 2.0 GHz central processing unit (Inter(R) Core(TM) i7-8565U CPU) and 8 GB memory, and 64 bit Windows operation system.

The coefficient matrix $A \in \mathbb{R}^{m\times n}$ is either generated by the MATLAB function $randn(m,n)$ or taken from the University of Florida sparse matrix collection [12]. To ensure each linear system (1) with a selected matrix A is consistent, we let $b \in \mathbb{R}^m$ in (1) be generated by Ax^*, where the exact solution $x^* \in \mathbb{R}^n$

is produced randomly by the MATLAB function **randn** and is to be determined. The *efficiency* of each method is evaluated by the number of iterations (*IT*) and the CPU time (*CPU*). It can be measured by the *speed-up of GGK against GPRK* (*SU*) is defined by

$$SU = \frac{CPU\ of\ GPRK}{CPU\ of\ GGK}.$$

The *effectiveness* of both methods is measured by the *relative residual* (*RR*) defined by

$$RR = \frac{\|b - Ax_k\|_2}{\|b\|_2}.$$

We set the initial solution x^0 be **0** in all experiments, and the iteration does not terminate until $RR < 10^{-6}$. The numerical results of each method shown in this section are arithmetical average quantities with respect to 50 repeated trials. We set $\eta = 0.3$ in GGK for each example.

3.1. Experiments with Sparse Matrix

This subsection considers the linear systems (1) with sparse matrices. These matrices include some flat ones in Table 1 and the tall ones (their transposes) in Table 2 from the University of Florida sparse matrix collection [12].

Table 1. Comparison of IT, CPU and SU of the GGK method with that of the GPRK method for different flat spare systems.

Matrix Name	Matrix Size	Density	GPRK		GGK		SU
			IT	CPU	IT	CPU	
bibd_16_8	120 × 12,870	23.33%	2142	0.9867	566.5	0.3650	**2.70**
crew1	135 × 6469	5.38%	6346.9	1.3223	804	0.1815	7.29
df2177	630 × 10,358	0.34%	3186.4	0.4443	40	0.0078	56.96
us04	163 × 28,016	6.52%	4134.1	5.9783	1635	1.8062	3.31
GL7d25	2789 × 21,074	0.14%	15,160	11.2236	556.1	0.3721	30.16
stat96v5	2307 × 75,779	0.13%	17,897	21.0132	149	0.1499	140.18
bibd_81_3	3240 × 85,320	0.09%	14,044	23.0241	38	0.0371	620.60
abtaha1^T	209 × 14,596	1.68%	30,081	6.6795	1118.7	0.2533	26.37
abtaha2^T	331 × 37,932	1.09%	62,803	48.0985	967.7	0.6389	75.28
mk12-b2^T	1485 × 13,860	0.20%	6477.2	1.2226	33.1	0.0115	106.31
ch7-9-b2^T	1512 × 17,640	0.20%	6387.7	2.2341	31.7	0.0097	230.32
relat7^T	1045 × 21,924	0.36%	53,325	27.6583	1020.2	0.4063	68.07
Franz9^T	4164 × 19,588	0.12%	19,264	8.5559	171.8	0.0700	122.23
Franz10^T	4164 × 19,588	0.12%	19,322	10.7741	172.6	0.0673	160.09
ch7-6-b3^T	4200 × 12,600	0.10%	21,260	9.4338	48.4	0.0150	**628.92**
relat7b^T	1045 × 21,924	0.36%	65,284	27.8051	1005.9	0.3955	70.30

Table 2. Comparison of IT, CPU and SU of the GGK method with that of the GPRK method for different thin spare systems.

Matrix Name	Matrix Size	Density	GPRK		GGK		SU
			IT	CPU	IT	CPU	
bibd_16_8^T	12,870 × 120	23.33%	1040.3	1.0199	299.2	0.2809	3.63
crew1^T	6469 × 135	5.38%	2613	0.4065	522.4	0.1632	**2.49**
df2177^T	10,358 × 630	0.34%	2023.7	0.7251	43.3	0.0102	71.09
us04^T	28,016 × 163	6.52%	1898.3	2.6322	548.7	0.5170	5.09
GL7d25^T	21,074 × 2789	0.14%	12911	14.6954	443.7	0.1799	81.69
stat96v5^T	75,779 × 2307	0.13%	8190.1	19.1244	153.0	0.1722	111.06
bibd_81_3^T	85,320 × 3240	0.09%	9241.2	20.9510	61.6	0.0738	283.89
abtaha1	14,596 × 209	1.68%	1819.6	0.6833	235.8	0.0498	13.72
abtaha2	37,932 × 331	1.09%	1855.5	1.5998	140.9	0.0814	19.65
mk12-b2	13,860 × 1485	0.20%	4828.3	2.4008	43.8	0.0131	183.27
ch7-9-b2	17,640 × 1512	0.20%	4734.2	2.3630	44.1	0.0128	184.61
relat7	21,924 × 1045	0.36%	46,674	35.1415	887.4	0.3432	102.39
Franz9	19,588 × 4164	0.12%	19,363	14.9236	203.9	0.0861	173.33
Franz10	19,588 × 4164	0.12%	19,354	16.4306	200.8	0.0842	195.14
ch7-6-b3	12,600 × 4200	0.10%	16,394	8.9110	51.9	0.0139	**641.08**
relat7b	21,924 × 1045	0.36%	46,708	35.0971	920.3	0.3826	91.73

Table 1 and Table 2 list the average number of iterations (IT), computing time (CPU), and the related speed-up (SU) of GGK against GPRK. Figure 1 and Figure 2 plot RR versus IT (left) and CPU (right) of different methods for solving (1) with the matrix *bibd_16_8* and *crew1*^{T}, respectively. From Table 1 and Table 2 we see that the GGK method needs smaller IT and CPU than the GPRK method does in all cases. We can see whether system 1 is overdetermined or underdetermined. When the matrix is flat, the value of SU locates in the interval (2.70, 628.92), while for the case of the thin matrix, the value of SU ranges from 2.49 to 641.08. It is observed from Figure 1 and Figure 2 that the GGK method converges faster than the GPRK method.

3.2. Experiments with Dense Matrix

In this subsection, the test matrices are dense normally distributed random matrices including thin and flat matrices. The sizes of rows and columns of the selected matrices vary from 2000 to 30,000.

Table 3 lists the test results of *IT*, *CPU* and *SU* for flat dense matrix with different size. Table 4 displays *IT*, *CPU* and *SU* for a different thin dense matrix with different size. Figure 3 and Figure 4 plot RR versus IT (left) and CPU (right) of GGK and GPRK methods for solving (1) with the matrix $A = randn(2200, 30000)$ and $A = randn(30000, 2000)$, respectively. From Table 1 and Table 2, we can

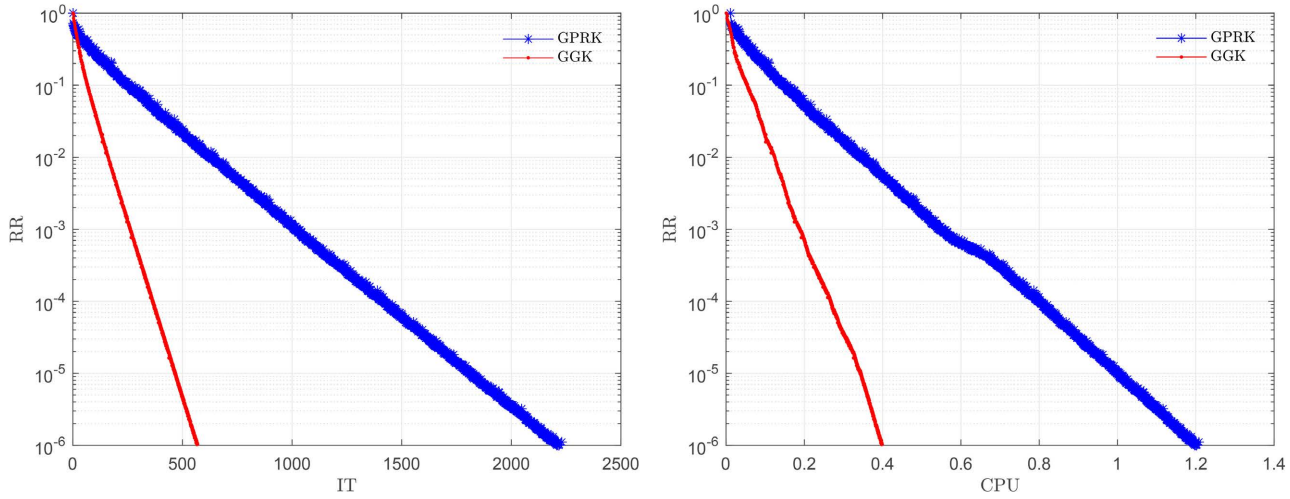

Figure 1. RR versus IT (left) and CPU (right) of the GGK method compared with the GPRK method for solving (1) with the matrix *bibd*_16_8.

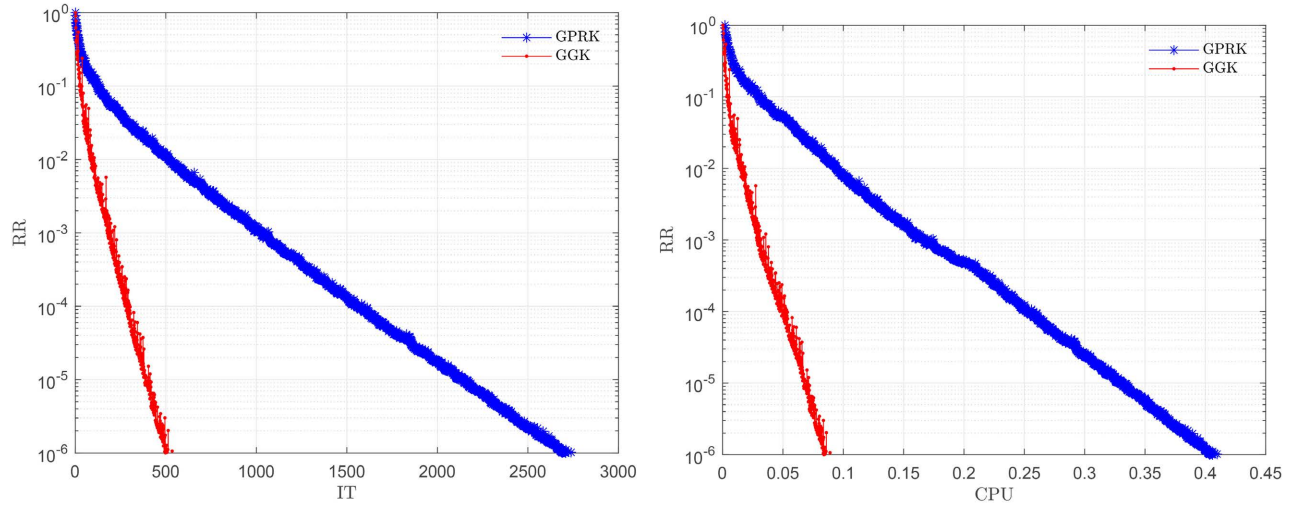

Figure 2. RR versus IT (left) and CPU (right) of the GGK method compared with the GPRK method for solving (1) with the matrix $crew1^T$.

Table 3. Comparison of IT, CPU and SU of the GGK method with that of the GPRK method for different flat dense systems.

Matrix Size	GPRK		GGK		SU
	IT	CPU	IT	CPU	
2000 × 10,000	21,569	297.5576	79.3	1.9810	98.85
2000 × 20,000	14,052	331.0652	50	2.5747	73.73
2000 × 30,000	11,762	495.7764	42.2	3.2314	**73.30**
2200 × 10,000	25,696	496.4009	86.9	6.0598	111.93
2200 × 20,000	16,259	537.1076	53.5	5.7038	135.89
2200 × 30,000	13,282	359.6292	43.9	7.1961	112.97
2400 × 10,000	29,720	535.0830	96.1	3.6249	205.64

Continued

2400 × 20,000	18,336	351.9513	55.6	3.9913	203.36
2400 × 30,000	15,271	546.2991	45.6	6.2161	182.75
2600 × 10,000	35,591	337.0178	105.7	7.0060	162.16
2600 × 20,000	20,914	390.1169	59.5	5.9782	194.22
2600 × 30,000	17,055	494.0944	47.6	6.3896	168.46
2800 × 10,000	41,358	527.2221	116.6	6.6916	171.33
2800 × 20,000	23,539	497.0007	61.6	6.0664	196.34
2800 × 30,000	19,092	599.8468	49.3	6.0754	204.63
3000 × 10,000	46,862	544.3403	127.2	2.5496	213.50
3000 × 20,000	26,704	603.3008	65.6	2.6693	226.02
3000 × 30,000	21,106	718.8279	51.5	3.1646	**227.15**

Table 4. Comparison of IT, CPU and SU of the GGK method with that of the GPRK method for different thin dense systems.

Matrix Size	GPRK		GGK		SU
	IT	CPU	IT	CPU	
10,000 × 2000	12,234	171.3824	61.5	0.8017	213.77
20,000 × 2000	7567.4	140.4876	34.7	1.3755	102.14
30,000 × 2000	6470	176.8242	27.6	1.3860	127.58
10,000 × 2200	14,776	441.3869	69.8	1.0381	**425.19**
20,000 × 2200	8633.6	392.7304	37.8	1.0948	358.72
30,000 × 2200	7323.6	405.4806	29.3	1.3893	291.86
10,000 × 2400	17,784	556.8869	76.9	1.8008	309.24
20,000 × 2400	9879.4	490.8223	40.8	1.6075	305.33
30,000 × 2400	8213.8	521.1035	31.2	2.4817	209.98
10,000 × 2600	21,949	633.4774	86.1	3.6521	173.46
20,000 × 2600	11,227	660.0913	43.2	2.3399	282.10
30,000 × 2600	9188.2	684.8802	33.3	3.3046	207.25
10,000 × 2800	25,698	758.3984	96.7	4.6805	162.00
20,000 × 2800	12,680	712.5364	46.2	3.1025	229.67
30,000 × 2800	10,136	737.2275	35.3	2.1491	343.04
10,000 × 3000	30,249	662.1158	106.8	6.3898	103.62
20,000 × 3000	14,236	595.5270	50	5.3462	**11.39**
30,000 × 3000	11,201	639.9166	36.6	4.5954	139.25

see in all cases, GGK needs less IT and CPU time than GPRK does, and the speed-up of GGK against GPRK ranges from tens of times to hundreds of times, *i.e.*, the speed-up of GGK against GPRK varies from 73.30 to 227.15 in the case of flat and from 11.39 to 425.19 in the case of thin. Similar results to **Figure 1**

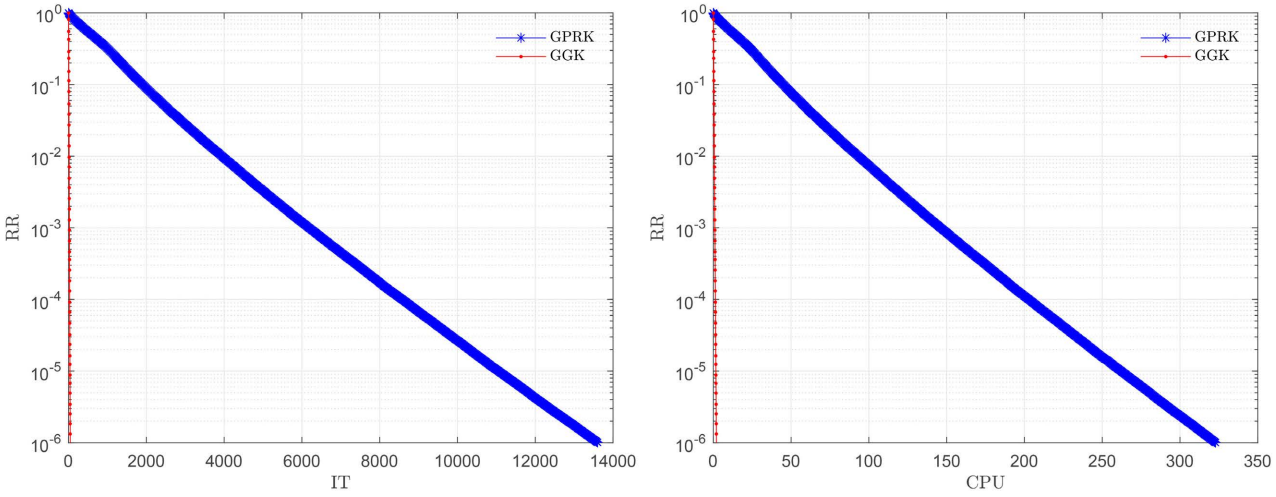

Figure 3. RR versus IT (left) and CPU (right) of the GGK method compared with the GPRK method for solving (1) with the matrix $A = randn(2200,30000)$.

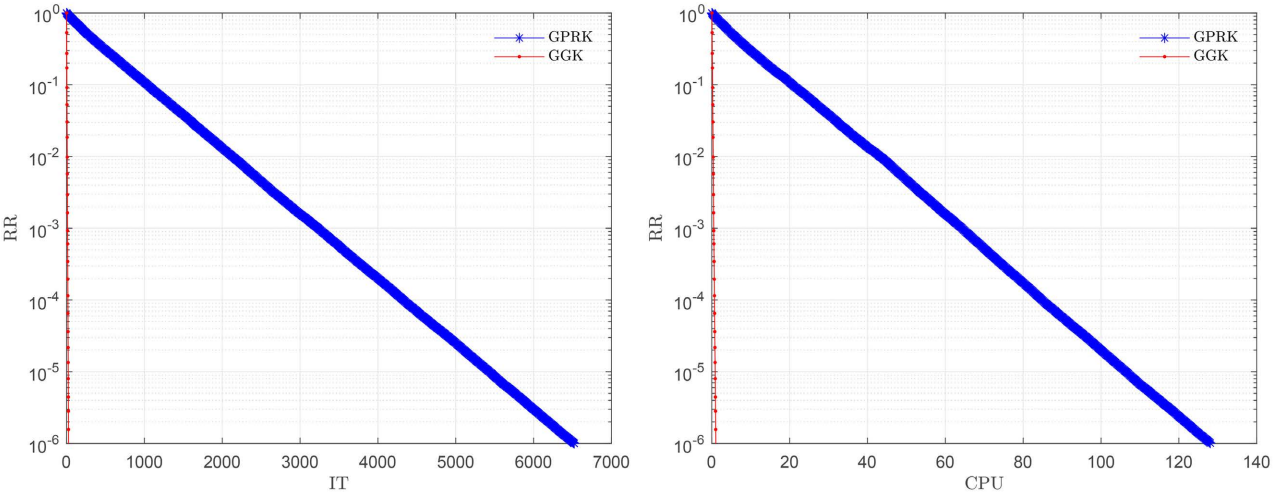

Figure 4. RR versus IT (left) and CPU (right) of the GGK method compared with the GPRK method for solving (1) with the matrix $A = randn(30000,2000)$.

and Figure 2 are drawn from Figure 3 and Figure 4 that the GGK method converges much faster than the GPRK method.

4. Conclusion

We develop a geometric Gauss-Kaczmarz (GGK) algorithm for solving large-scale consistent linear systems and the convergence is proved for this algorithm. Numerical experiments show that the GGK algorithm has better efficiency and effectiveness than the GPRK algorithm. In our future work, we will focus on block Kaczmarz methods to solve ill-posed problems.

Acknowledgements

The authors thank the reviewers for providing some helpful comments. Research

by G.H. was supported in part by Application Fundamentals Foundation of STD of Sichuan (grant 2020YJ0366) and the Opening Project of Sichuan Province University Key Laboratory of Bridge Non-destructionDetecting and Engineering Computing (grant 2020QZJ03), and research by F.Y. was partially supported by NNSF (grant 11501392) and SUSE (grant 2019RC09, 2020RC25), and research by Y.M. Liao was partially supported by the Innovation Fund of Postgraduate of SUSE (grant y2021101).

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Yang, X. (2020) A Geometric Probability Randomized Kaczmarz Method for Large Scale Linear Systems. *Applied Numerical Mathematics*, **164**, 139-160. https://doi.org/10.1016/j.apnum.2020.10.016

[2] Kaczmarz, S. (1937) Angenäherte Auflösung von Systemen Linearer Gleichungen. *Bulletin International de l' Académie Polonaise des Sciences et des Lettres*, **35**, 355-357.

[3] Ihrig, A. and Schmitz, G. (2018) Accelerating Nonlinear Speed of Sound Reconstructions Using a Randomized Block Kaczmarz Algorithm. 2018 *IEEE International Ultrasonics Symposium* (*IUS*), Kobe, Japan, 22-25 October 2018, 1-9. https://doi.org/10.1109/ULTSYM.2018.8580199

[4] Popa, C. and Zdunek, R. (2004) Kaczmarz Extended Algorithm for Tomographic Image Reconstruction from Limited-Data. *Mathematics and Computers in Simulation*, **65**, 579-598. https://doi.org/10.1016/j.matcom.2004.01.021

[5] Lorenz, D.A., Wenger, S., Schöpfer, F. and Magnor, M. (2014) A Sparse Kaczmarz Solver and A Linearized Bergman Method for Onlinear Compressed Sensing. 2014 *IEEE International Conference on Image Processing* (*ICIP*), Parise, France, 27-30 October 2014, 1347-1351. https://doi.org/10.1109/ICIP.2014.7025269

[6] Strohmer, T. and Vershynin, R. (2009) A Randomized Kaczmarz Algorithm with Exponential Convergence. *Journal of Fourier Analysis and Applications*, **15**, Article No. 262. https://doi.org/10.1007/s00041-008-9030-4

[7] Bai, Z.Z. and Wu, W.T. (2018) On Greedy Randomized Kaczmarz Method for Solving Large Sparse Linear Systems. *SIAM Journal on Scientific Computing*, **40**, A592-A606. https://doi.org/10.1137/17M1137747

[8] Needell, D., Tropp, J. A. (2014) Paved with Good Intentions: Analysis of A Randomized Block Kaczmarz Method. *Linear Algebra and Its Applications*, **441**, 199-221. https://doi.org/10.1016/j.laa.2012.12.022

[9] Niu, Y.Q. and Zheng, B. (2020) A Greedy Block Kaczmarz Algorithm for Solving Large-Scale Linear Systems. Applied Mathematics Letters, **104**, Article No. 106294. https://doi.org/10.1016/j.aml.2020.106294

[10] Gower, R.M. and Richtárik, R. (2015) A Randomized Kaczmarz Algorithm with Exponential Convergence. *SIAM Journal on Matrix Analysis and Applications*, **36**, 1660-1690. https://doi.org/10.1137/15M1025487

[11] Chen, J.Q. and Huang, Z.D. (2021) On a Fast Deterministic Block Kaczmarz Method for Solving Large-scale Linear Systems. *Numerical Algorithms*. https://doi.org/10.1007/s11075-021-01143-4

[12] Davis, T.A. and Hu, Y. (2011) The University of Florida Sparse Matrix Collection. *ACM Transactions on Mathematical Software*, **38**, 1-25. https://doi.org/10.1145/2049662.2049663