Advances in Applied Mathematics 应用数学进展, 2024, 13(11), 5063-5072
Published Online November 2024 in Hans. https://www.hanspub.org/journal/aam
https://doi.org/10.12677/aam.2024.1311489

Hans汉斯

求解大型线性系统的K-均值最大残差块方法

黄柏顺

温州大学数理学院，浙江 温州

收稿日期：2024年10月27日；录用日期：2024年11月21日；发布日期：2024年11月28日

摘　要

求解大型线性系统，带K-均值聚类的贪婪随机块Kaczmarz方法是近几年被广受关注的一类方法。本文在该方法基础上做了进一步的研究即在每一次迭代中优先消除残差向量中的最大块，构建了最大残差块Kaczmarz方法及其加速版本并进行了收敛性分析。数值实验证实了本文算法的有效性。

关键词

一致线性系统，最大残差块Kaczmarz，K-均值算法，收敛性质

K-Means Maximum Residual Block Method for Solving Large Linear Systems

Baishun Huang

College of Mathematics and Physics, Wenzhou University, Wenzhou Zhejiang

Received: Oct. 27th, 2024; accepted: Nov. 21st, 2024; published: Nov. 28th, 2024

Abstract

The greedy random block Kaczmarz method with K-means clustering for solving large linear systems has been widely studied in recent years. This article conducted further research on this method by prioritizing the elimination of the largest block in the residual vector in each iteration, constructing the Kaczmarz method for the maximum residual block and its accelerated version, and conducting convergence analysis. Numerical experiments have confirmed the effectiveness of the algorithm proposed in this paper.

Keywords

Consistent Linear System, Maximum Residual Block Kaczmarz, K-Means Algorithm, Convergence Properties

文章引用: 黄柏顺. 求解大型线性系统的 K-均值最大残差块方法[J]. 应用数学进展, 2024, 13(11): 5063-5072.
DOI: 10.12677/aam.2024.1311489

Copyright © 2024 by author(s) and Hans Publishers Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/
Open Access

1. 引言

随着数据科学的发展和现实世界问题规模的不断扩大，研究求解大型线性系统的有效算法成为一个重要的课题。具体地说，我们考虑如下形式的线性系统

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

其中 $A \in R^{m\times n}$ 是一个 $m\times n$ 矩阵，是已知向量且系统(1)是相容的。研究者通常使用选代方法求解此类问题。其中一种方法是 Kaczmarz 方法[1]，该方法的选代形式如下：

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

其中 $A^{(i)}$ 表示 A 的第 i 行， $b^{(i)}$ 表示 b 的第 i 个元素。

对于任何向量 $y \in \mathbb{C}^n$， $(y)^{(j_k)}$ 表示第 k 次选代的第 j 个元素，用 e_i 表示除了第 i 个位置的元素为 1 其余为零的列向量，给定任意矩阵 B，对于矩阵 $B = (b_{ij}) \in \mathbb{R}^{m\times n}$， $B_{(j)}$ 表示 B 的第 j 列，如果 B 对称的，用 $\sigma_{\min}(B)$ 表示 B 的最小奇异值，用 $\sigma_{\max}(B)$ 表示 B 的最大奇异值。此外，我们定义$[m]$为$1,2,\cdots,m$，其中 m 是任意的正整数。我们考虑集合 $\mathcal{V} = \{\mathcal{V}_1, \mathcal{V}_2, \cdots, \mathcal{V}_k\}$ 作为$[m]$的一个划分，如果索引集 $\mathcal{V}$，其中 $i = 1,2,\cdots,k$，满足条件 $\mathcal{V}_i \cap \mathcal{V}_j = \varnothing$，其中 $\varnothing$ 是空集。对于 $i \neq j$，以及 $U_{i=1}^{k} V_i = [m]$。进一步地，给定一个行索引集 $\mathcal{V}$，我们使用 $A_{\mathcal{V}_i}$ 表示由 $\mathcal{V}$ 索引的矩阵 A 的子矩阵，使用 $b\nu_i$ 表示向量 b 的子向量，$A_{\mathcal{V}_i}$ 和 $b_{\mathcal{V}_i}$ 的大小由 K-means 聚类后的结果决定。

为了便于解释，我们简要说明一下 K-means 方法。K-means 聚类是一种迭代算法，他将输入的数据 X 分为 k 组，每个项分配给具有最近质心的聚类。这些组 $C = \{C_1, \cdots, C_k\}$ 满足

$$\bigcup_{v=1}^{k} C_v = X,\ C_v \cap C_j = \varnothing$$

其中 $1 \le v \ne j \le k$，$\varnothing$ 是一个空集。

在 K-means 均值聚类中，聚类的数量由 K 决定，是我们指定的正整数。最初，k 个聚类是从给定数据中随机选择的然后将一个项分配给质心最近的聚类所谓“最接近”就是指对它们的距离或者相似度最小。对于距离我们定义为：

$$\sum_{r=1}^{k}\sum_{x_i \in C_r} \left\|x_i - \rho_r\right\|_2^2$$

其中 $X = \{x_i\}$ 是给定的数据，是聚类的质心，$i = 1,2,\cdots,m$； $v = 1,2,\cdots,k$。对于相似系数，我们使用以下标准来衡量项目对的接近度，该标准为

$$\sum_{v=1}^{k}\sum_{x_1 \in C_v} \left|1 - \frac{x_i^{\mathrm{T}}\rho_v}{\left\|x_i\right\|_2 \left\|\rho_v\right\|_2}\right|$$

上述过程相继进行，直到不发生分配结束。Jiang 和 Zhanng 在[2]中给出了完整过程。

一个列划分函数： $\mathcal{V}_1, \mathcal{V}_2, \cdots, \mathcal{V}_k$ 是由算法开始的初始步所定得到。 $X = [A, b] \in C^{m\times(n+1)}$ 通过 K-means 方

法我们得到关于 A 和 b 的行指标 $\{1,\cdots,m\}$ 的划分 $T=\{v_1,\cdots,v_k\}$，也就是：$A=\left[A_{v_1};\cdots;A_{v_k}\right]$ 和 $b=\left[b_{v_1};\cdots;b_{v_k}\right]$；$\bar{A}_{v_i}$ 和 $\bar{b}_{v_i}$ 分别是块 A_{v_i} 和 b_{v_i} 相对应的簇中心，$i=1,2,\cdots,k$。我们可以写成紧凑形式即：$\bar{A}=\left[\bar{A}_{v_1};\cdots;A_{v_k}\right]$ 和 $\bar{b}=\left[\bar{b}_{v_1};\cdots;\bar{b}_{v_k}\right]$。

求解线性和非线性问题在各个领域的应用都比较广泛，例如在自动控制[3] [4]，医学[5] [6]，数据测绘[7]-[10]等其他领域[11]-[15]。对于大型线性系统的求解方法。随机 Kaczmarz 算法是一种比较有效的解决办法。Strohmer 和 Vershynin [1]提出了随机 Kaczmarz 算法。遵循 RK 的方法，Bai 和 Wu 提出了贪婪随机化 Kaczmarz 方法(RGRK) [16]，在[2]中作者使用 K-均值聚类来划分系数矩阵，同时采用了 GRK 中使用的贪心技术，构造了求解线性系统(1)的 K-均值聚类确定的随机块 Kaczmarz 方法(RBK(k))。本文在(RBK(k))的基础上进行了改进。在每一次的选代中根据 $i_k=\arg\max_{1\le i\le k}\left\|bv_i-Av_ix_k\right\|_2^2$ 选择工作行块 $\mathcal{V}_{i_k}$ 确保首先消除最大的残差。在本文中提出了两种算法并给出了收敛性的证明，并在最后的数值实验部分与 RBK(k)进行了比较，实验结果表明了算法的优越性。

2. 带 K-均值最大残差块方法

算法 1 是文献[2]提出的随机块 Kaczmarz 方法，该算法如下。

算法 1：带 K-均值随机块 Kaczmarz 方法(RBK(k))

输入：A,b,x_0,k,l 和 θ

输出：x_l

1：对 A 和 b 使用 K-均值方法，得到行索引 $1,\cdots,m$ 的划分 $\{v_1,\cdots,v_k\}$，得到 k 组 A_{v_i} 和 b_{v_i}，$i=1,2,\cdots,k$。A_{v_i} 和 b_{v_i} 再以紧凑形式分配即：$\bar{A}=\left[\bar{A}_{v_1};\cdots;\bar{A}_{v_k}\right]$ 和 $\bar{b}=\left[\bar{b}_{v_1};\cdots;\bar{b}_{v_k}\right]$

2：对 $j=1,2,\cdots,l$ 做

3：计算：$\epsilon_j=\dfrac{\theta}{\left\|\bar{b}-\bar{A}x_j\right\|_2^2}\max_{1\le v_j\le k}\left\{\dfrac{\left|\bar{b}_{v_j}-\bar{A}_{v_j}x_j\right|^2}{\left\|\bar{A}_{v_j}\right\|_2^2}\right\}+\dfrac{1-\theta}{\|A\|_F^2},\theta\in(0,1)$

4：令 $U_j=\left\{v_j\left|\left|\bar{b}_{v_j}-\bar{A}_{v_j}x_j\right|^2\ge\epsilon_j\left\|\bar{b}-\bar{A}x_j\right\|_2^2\left\|\bar{A}_{v_j}\right\|_2^2\right.\right\}$

5：根据下面规则计算 $\tilde{r}_j^{(v)}$

$$\tilde{r}_j^{(v)}=\begin{cases}\bar{b}_v-\bar{A}_vx_j, & \text{如果}v\in U_j\\ 0, & \text{其他}\end{cases}$$

6：依概率 $\Pr\left(v=v_j\right)=\dfrac{\left|\tilde{r}_j^{(v)}\right|^2}{\left\|\tilde{r}_j\right\|_2^2}$ 选择 $v_j\in U_j$

7：计算 $x_{j+1}=x_j+A_{v_j}^{\dagger}\left(b_{v_j}-A_{v_j}x_j\right)$

8：结束

算法 2 是在算法 1 的基础上，直接选择了具有最大残差的行块。

算法 2：K-均值最大残差块方法

输入：A,b,x_0,k,l

输出：x_l

1：使用 K-均值方法对 A 和 b 处理，得到行索引 $1,\cdots,m$ 的划分 $\{v_1,\cdots,v_k\}$，就有 k 组 A_{v_i} 和 b_{v_i}，$i=1,2,\cdots,k$。

2：对 $j=1,2,\cdots,l$ 做

3：选择 $i_j=\arg\max_{1\le i\le k}\left\|bv_i-Av_ix_j\right\|_2^2$

4：计算 $x_{j+1}=x_j+A_{\nu_{i_j}}^{\dagger}\left(b_{\nu_{i_j}}-A_{\nu_{i_j}}x_j\right)$

5：结束

算法 3 是在算法 2 的基础上，避免了求伪逆的计算，大大节省了计算成本。

算法 3：K-均值最大残差块的加速方法

输入： A,b,x_0,k,l

输出： x_l

1：使用 K-均值方法对 A 和 b 处理，得到行索引 $1,\cdots,m$ 的划分 $\{v_1,\cdots,v_k\}$，就有 k 组 A_{v_i} 和 b_{v_i}，$i=1,2,\cdots,k$。

2：对 $j=1,2,\cdots,l$ 做

3：选择 $i_j=\arg\max_{1\le i\le k}\left\|bv_i-Av_ix_j\right\|_2^2$

4：计算 $\alpha_j=\omega\dfrac{\left\|b_{v_{i_j}}-Av_{i_j}x_j\right\|_2^2\left\|Av_{i_j}\right\|_F^2}{\left\|A_{v_{i_j}}^{\mathrm{T}}\left(bv_{i_j}-Av_{i_j}x_j\right)\right\|_2^2}$

5：计算 $x_{j+1}=x_j+\alpha_j\dfrac{A_{v_{i_j}}^{\mathrm{T}}\left(bv_{i_j}-Av_{i_j}x_j\right)}{\left\|Av_{i_j}\right\|_F^2}$

6：结束

3. 算法 2 和算法 3 的收敛性分析

引理 1 [17]：设向量 $z\in\mathcal{R}\left(A^{\mathrm{T}}\right)$，则有 $\|Az\|_2^2\ge\sigma_{\min}^2(A)\|z\|_2^2$。

定理 1：对于相容线性系统(1)，由算法 2 选代生成的序列 $\left\{x_j\right\}_{i=0}^{\infty}$ 收敛到 $x_\star=A^{\dagger}b$，对于任意的 $j\ge0$，满足下列关系：

$$\left\|x_1-x_\star\right\|_2^2\le\left(1-\frac{\sigma_{\min}^2(A)}{\sigma_{\max}\left(A_vA_v^{\mathrm{T}}\right)k}\right)\left\|x_0-x_\star\right\|_2^2$$

以及

$$\left\|x_{j+1}-x_\star\right\|_2^2\le\left(1-\frac{\sigma_{\min}^2(A)}{\sigma_{\max}\left(A_vA_v^{\mathrm{T}}\right)(k-1)}\right)^j\left(1-\frac{\sigma_{\min}^2(A)}{\sigma_{\max}\left(A_vA_v^{\mathrm{T}}\right)k}\right)\left\|x_0-x_\star\right\|_2^2$$

证明：

$$x_{j+1}-x_\star=x_j-x_\star+A_{\nu_{i_j}}^{\dagger}\left(b_{\nu_{i_j}}-A_{\nu_{i_j}}x_j\right)$$

$$A_{\nu_{i_j}}\left(x_{j+1}-x_\star\right)=A_{\nu_{i_j}}\left(x_j-x_\star\right)-A_{\nu_{i_j}}A_{\nu_{i_j}}^{\dagger}A_{\nu_{i_j}}\left(x_j-x_\star\right)=0$$

所以

$$A_{\nu_{i_j}}^{\dagger}A_{\nu_{i_j}}\left(x_{j+1}-x_\star\right)=0$$

由于

$$x_{j+1}-x_j=A_{v_{i_j}}^{\dagger}A_{v_{i_j}}\left(x_\star-x_j\right)$$

所以

$$x_{j+1}-x_j\text{ 与 }x_{j+1}-x_\star\text{ 正交}$$

$$\left\|x_{j+1}-x_\star\right\|_2^2=\left\|x_j-x_\star\right\|_2^2-\left\|A_{v_{i_j}}^{\dagger}A_{v_{i_j}}\left(x_j-x_\star\right)\right\|_2^2$$

下面第一个等式是根据引理 1 得到的

$$\begin{aligned}\left\|A_{v_{i_j}}^{\dagger}Av_{i_j}\left(x_j-x_\star\right)\right\|_2^2 &\geq \sigma_{\min}^2\left(A_{\lambda_j}^{\dagger}\right)\left\|A_{\lambda_j}\left(x_j-x_\star\right)\right\|_2^2\\ &=\frac{1}{\sigma_{\max}^2\left(A_{\lambda_j}\right)}\left\|A_{\lambda_j}\left(x_j-x_\star\right)\right\|_2^2\\ &\geq\frac{1}{\sigma_{\max}\left(A_vA_v^{\mathrm{T}}\right)}\left\|A_{v_{i_j}}\left(x_j-x_\star\right)\right\|_2^2\\ &=\frac{1}{\sigma_{\max}\left(A_vA_v^{\mathrm{T}}\right)}\left\|bv_{i_j}-Av_{i_j}x_j\right\|_2^2\\ &=\frac{1}{\sigma_{\max}\left(A_vA_v^{\mathrm{T}}\right)}\max_{1\leq i\leq k}\left\|b_{v_i}-A_{v_i}x_j\right\|_2^2\end{aligned}$$

此外

$$\begin{aligned}bv_{i_j}-A_{v_{i_j}}x_{j+1}&=b_{\lambda_j}-A_{\lambda_j}\left(x_j+A_{\lambda_j}^{\dagger}\left(b_{\lambda_i}-A_{\lambda_i}x_j\right)\right)\\ &=b_{\lambda_j}-A_{\lambda_j}x_j-A_{\lambda_j}A_{\lambda_j}^{\dagger}\left(b_{\lambda_j}-A_{\lambda_j}x_j\right)\\ &=b_{\lambda_j}-A_{\lambda_j}x_j-A_{\lambda_j}A_{\lambda_j}^{\dagger}b_{\lambda_j}+A_{\lambda_j}A_{\lambda_j}^{\dagger}A_{\lambda_j}x_j\\ &=A_{V_{i_j}}x_\star-A_{V_{i_j}}A_{V_{i_j}}^{\dagger}A_{V_{i_j}}x_\star\\ &=A_{\lambda_j}x_\star-A_{\lambda_j}x_\star\\ &=0\end{aligned}$$

因此我们可以得到对于 $j=1,2,\cdots$ 有

$$\left\|b-Ax_j\right\|_2^2=\sum_{\lambda_j\in\mathcal{V}-\lambda_{j-1}}\left\|b_{\lambda_j}-A_{\lambda_j}x_j\right\|_2^2\leq(k-1)\max_{1\leq i\leq k}\left\|b_{\lambda_i}-A_{\lambda_i}x_j\right\|_2^2$$

当 $j=0$ 我们还可以得到

$$\left\|b-Ax_0\right\|_2^2\leq k\max_{1\leq i\leq k}\left\|b_{\lambda_i}-A_{\lambda_i}x_0\right\|_2^2$$

因此有

$$\max_{1\leq i\leq k}\left\|b_{\lambda_i}-A_{\lambda_i}x_j\right\|_2^2\geq\frac{1}{k-1}\left\|b-Ax_j\right\|_2^2$$

综上所述，我们最终就可以得到

$$\begin{aligned}\left\|x_{j+1}-x_\star\right\|_2^2&\leq\left\|x_j-x_\star\right\|_2^2-\frac{1}{\sigma_{\max}\left(A_vA_v^{\mathrm{T}}\right)}\frac{\left\|b-Ax_j\right\|_2^2}{k-1}\\ &=\left\|x_j-x_\star\right\|_2^2-\frac{1}{\sigma_{\max}\left(A_vA_v^{\mathrm{T}}\right)}\frac{\left\|A\left(x_j-x_\star\right)\right\|_2}{k-1}\\ &\leq\left\|x_j-x_\star\right\|_2^2-\frac{\sigma_{\min}^2\left(A\right)}{\sigma_{\max}\left(A_vA_v^{\mathrm{T}}\right)(k-1)}\left\|x_j-x_\star\right\|_2^2\\ &=\left(1-\frac{\sigma_{\min}^2\left(A\right)}{\sigma_{\max}\left(A_vA_v^{\mathrm{T}}\right)(k-1)}\right)\left\|x_j-x_\star\right\|_2^2\end{aligned}$$

定理 2：对于对于相容线性系统(1)，设 $\omega\in(0,2)$ 由算法 3 选代生成的序列 $\{x_j\}_{j=0}^{\infty}$ 收敛到 $x_\star=A^{\dagger}b$，对于任意的 $j\geq 0$，满足下列关系：

$$\|x_1-x_\star\|_2^2\leq\left(1-(2\omega-\omega^2)\frac{\sigma_{\min}^2(A)}{\sigma_{\max}\left(A_vA_v^{\mathrm{T}}\right)k}\right)\|x_0-x_\star\|_2^2$$

以及

$$\|x_{j+1}-x_\star\|_2^2\leq\left(1-(2\omega-\omega^2)\frac{\sigma_{\min}^2(A)}{\sigma_{\max}\left(A_vA_v^{\mathrm{T}}\right)(k-1)}\right)^j\left(1-(2\omega-\omega^2)\frac{\sigma_{\min}^2(A)}{\sigma_{\max}\left(A_vA_v^{\mathrm{T}}\right)k}\right)\|x_0-x_\star\|_2^2$$

证明：从算法 3 的第 5 步和 $A_{\nu_{i_j}}x_j=b_{\nu_{i_j}}$，我们可以得到

$$\begin{aligned}x_{j+1}-x_\star&=x_j-x_\star-\alpha_j\frac{A_{\nu_{i_j}}^{\mathrm{T}}\left(b\nu_{i_j}-A\nu_{i_j}x_j\right)}{\left\|A\nu_{i_j}\right\|_F^2}\\&=x_j-x_\star-\alpha_j\frac{A_{\nu_{i_j}}^{\mathrm{T}}A_{\nu_{i_j}}\left(x_j-x_\star\right)}{\left\|A_{\nu_{i_j}}\right\|_F^2}\\&=\left(I-\alpha_j\frac{A_{\nu_{i_j}}^{\mathrm{T}}A\nu_{i_j}}{\left\|A_{\nu_{i_j}}\right\|_F^2}\right)\left(x_j-x_\star\right)\end{aligned}$$

对上述等式取欧几里得范数的平方，可以得到

$$\begin{aligned}\|x_{j+1}-x_\star\|_2^2&=\left\|\left(I-\alpha_j\frac{A_{\nu_{i_j}}^{\mathrm{T}}A\nu_{i_j}}{\left\|A_{\nu_{i_j}}\right\|_F^2}\right)\left(x_j-x_\star\right)\right\|_2^2\\&=\|x_j-x_\star\|_2^2-2\alpha_j\frac{A_{\nu_{i_j}}\left(x_j-x_\star\right)}{\left\|A_{\nu_{i_j}}\right\|_F^2}+\alpha_j^2\frac{A_{\nu_{i_j}}^{\mathrm{T}}A_{\nu_{i_j}}\left(x_j-x_\star\right)}{\left\|A_{\nu_{i_j}}\right\|_F^2}\end{aligned}$$

将 α_j 代入上式可以得到

$$\begin{aligned}\|x_{j+1}-x_\star\|_2^2&=\|x_j-x_\star\|_2^2-(2\omega-\omega^2)\frac{\left\|b_{\nu_{i_j}}-A\nu_{i_j}x_j\right\|_2^4}{\left\|A_{\nu_{i_j}}^{\mathrm{T}}\left(b\nu_{i_j}-A\nu_{i_j}x_j\right)\right\|_2^2}\\&\leq\|x_j-x_\star\|_2^2-(2\omega-\omega^2)\frac{\left\|b\nu_{i_j}-A\nu_{i_j}x_j\right\|_2^4}{\sigma_{\max}^2\left(A\nu_{i_j}\right)\left\|b\nu_{i_j}-A\nu_{i_j}x_j\right\|_2^2}\\&\leq\|x_j-x_\star\|_2^2-(2\omega-\omega^2)\frac{\left\|b_{\nu_{i_j}}-A_{\nu_j}x_j\right\|_2^2}{\sigma_{\max}\left(A_vA_v^{\mathrm{T}}\right)}\\&=\|x_j-x_\star\|_2^2-(2\omega-\omega^2)\frac{\max\limits_{1\leq i\leq k}\left\|b_{\nu_i}-A_{\nu_i}x_j\right\|_2^2}{\sigma_{\max}\left(A_vA_v^{\mathrm{T}}\right)}\end{aligned}$$

第一个不等式成立因为 $2\omega-\omega^2>0$， $\omega\in(0,2)$ 以及根据引理 1 可以得到的下面不等式

$$\left\|A_{\lambda_j}^{\mathrm{T}}\left(b_{\lambda_j}-A_{\lambda_j}x_j\right)\right\|_2^2\le\sigma_{\max}^2\left(A_{\lambda_j}\right)\left\|b_{\lambda_j}-A_{\lambda_j}x_j\right\|_2^2$$

从定理 1 的证明可以看出，对于 $i=1,2,\cdots$ 有下面的式子成立 0

$$\begin{aligned}\left\|x_{j+1}-x_\star\right\|_2^2&\le\left\|x_j-x_\star\right\|_2^2-\left(2\omega-\omega^2\right)\frac{\left\|b-Ax_j\right\|_2^2}{\sigma_{\max}\left(A_vA_v^{\mathrm{T}}\right)(k-1)}\\&\le\left\|x_j-x_\star\right\|_2^2-\left(2\omega-\omega^2\right)\frac{\sigma_{\min}^2(A)\left\|x_j-x_\star\right\|_2^2}{\sigma_{\max}\left(A_vA_v^{\mathrm{T}}\right)(k-1)}\\&=\left(1-\left(2\omega-\omega^2\right)\frac{\sigma_{\min}^2(A)}{\sigma_{\max}\left(A_vA_v^{\mathrm{T}}\right)(k-1)}\right)\left\|x_j-x_\star\right\|_2^2\end{aligned}$$

4. 数值实验

用算法 1~3 方法求解线性系统(1)的例子是由 Matlab 函数 randn 服从标准正态分布随机生成的矩阵，以及从文献中选取的 5 个具有应用背景的稀疏矩阵[18]。分别是 Franz1、mk11 b2、Trefethen-300、gen4、p6000。

用 IT 表示选代次数，CPU 表示运行时间(单位：秒)，表格中的 IT 和 CPU 是循环运行 5 次取的平均值，为了更加直观地显示算法 2 和算法 3 较算法-1 的有效性。我们定义 $\text{speed-up1}=\dfrac{\text{CPU of Alogrithm2}}{\text{CPU of Alogrithm1}}$，$\text{speed-up2}=\dfrac{\text{CPU of Alogrithm3}}{\text{CPU of Alogrithm1}}$。对于选取的具有应用背景的矩阵，我们也给出了这些矩阵的基本信息，其中 $\text{cond}(A)$ 和 $\text{density}(A)$ 分别表示矩阵的条件数和稠密性。从表 1 可以知道该信息。我们定义 $\text{density}=\dfrac{\text{number of nonzeros of an } m\times n \text{ matrix}}{mn}$。在数值实验中，解向量是由 $x_\star$ 是由 Matlab 函数 randn 随机生成的，其元素是遵循独立的标准正态分布。根据右侧向量 b，我们取 $b=Ax_\star$，所有计算的初始向量从 $x_0=0$ 开始的。我们取 RES 表示选代值和真实值的相对误差设置停机准则为 $\text{RSE}=\dfrac{\left\|x_j-x_\star\right\|_2^2}{\left\|x_\star\right\|_2^2}\le10^{-6}$ 或超过规定的最大选代次数 $\text{IT}=200000$。

在下列表中我们分别列出了算法 1~3 方法的选代步数(IT)和计算时间(CPU)。对于随机生成的矩阵，从表 2，表 3 我们可以看出算法 2 和算法 3 的方法在选代步数(IT)和计算时间(CPU)都是优于算法 1 的方法。来自 Davis 和 Hu 工作的稀疏列满秩矩阵，我们可以从表 4 可以看出，算法 2 和算法 3 在比较的时候是有竞争力的。

Table 1. Specific matrix related information
表 1. 具体矩阵的相关信息

矩阵	Franz1	mk11-b2	Trefethen-300	gen4	p6000
$m\times n$	2240 × 768	6390 × 990	300 × 300	1537 × 4298	2095 × 7967
density	0.003	0.003	0.052	0.0162	0.0012
cond(A)	7.13E+15	5.04E+15	1772.7	39.19	4710

Table 2. Randomly generated matrix A, numerical results for solving linear systems when m = 8000 and n is different
表 2. 随机生成矩阵 A，当 m = 8000，n 不同时求解线性系统的数值结果

$m \times n$		8000 × 100	8000 × 200	8000 × 500	8000 × 1000	8000 × 2000
RBK(100)	IT	4	18.2	65	148.6	867.9
	CPU	0.3073	0.4655	0.7049	1.2617	3.444
MRBK(100)	IT	2	2	2	2	867.9
	CPU	0.2594	0.406	0.7016	1.6365	3.444
MARBK(100)	IT	13.2	13	10	14.2	1711.6
	CPU	0.1973	0.3211	0.5318	1.0977	2.4463
RBK(200)	IT	20.6	52	150.2	589.1	310
	CPU	0.3884	0.5219	0.8537	1.142	1.8901
MRBK(200)	IT	2	2	2	1136.7	2
	CPU	0.2792	0.457	0.7249	1.1415	1.8686
MARBK(200)	IT	17.2	15	12.6	1136.7	16.2
	CPU	0.2111	0.3506	0.5740	1.1415	1.3085
RBK(400)	IT	68.2	140.6	289.2	1136.7	604
	CPU	0.4616	0.705	1.3092	1.1415	2.9269
MRBK(400)	IT	2	2	2	1136.7	2
	CPU	0.3494	0.4697	0.8575	1.1415	1.797
MARBK(400)	IT	19.2	19	14.2	1136.7	19
	CPU	0.3	0.3436	0.5987	1.1415	1.3888

Table 3. Randomly generated matrix A, numerical results for solving linear systems when n = 8000 and m is different
表 3. 随机生成矩阵 A，当 n = 8000，m 不同时求解线性系统的数值结果

$m \times n$		100 × 8000	200 × 8000	500 × 8000	1000 × 8000	2000 × 8000
RBK(2)	IT	5	6	7	9	13
	CPU	0.0655	0.1124	0.4639	1.1651	2.976
MRBK(2)	IT	5	6	7	9	14
	CPU	0.0503	0.1108	0.3963	0.8916	2.4732
MARBK(2)	IT	8	9	12	17	27
	CPU	0.043	0.0892	0.3412	0.8569	2.4311
RBK(4)	IT	10.4	12.8	18	22	34.6
	CPU	0.0669	0.1505	0.5937	1.3685	3.3253
MRBK(4)	IT	8	9.6	12	15.2	23.2
	CPU	0.0491	0.0992	0.4762	0.9208	2.539
MARBK(4)	IT	11.2	12.2	16.8	22	36.8
	CPU	0.0440	0.0993	0.4071	0.8747	2.8612

续表

RBK(6)	IT	16.4	20.8	28	32.2	57
	CPU	0.0519	0.1121	0.5102	1.3914	3.649
MRBK(6)	IT	12	13.2	15.8	22.2	36.4
	CPU	0.0517	0.1165	0.3808	1.0045	2.757
MARBK(6)	IT	13.4	15.8	20.4	28.4	48.2
	CPU	0.0419	0.0993	0.3554	0.932	2.8281

Table 4. Numerical results of solving linear systems using specific matrices
表 4. 具体矩阵求解线性系统的数值结果

矩阵		Franz1	mk11-b2	Trefethen-300	gen4	p6000
RBK(k)	IT	15.4	5	52	521.8	30
	CPU	2.2341	0.8508	0.1003	1.1908	1.7317
MRBK(k)	IT	4	3.8	52	399.4	29.4
	CPU	0.2671	0.8474	0.0117	1.081	1.71
MARBK(k)	IT	34.4	10	74	931.4	49.6
	CPU	0.0075	0.0362	0.0063	0.8079	0.0244
k	/	4	4	20	4	4
speed-up1	/	8.36428304	1.00401227	8.57264957	1.10157262	1.01269006
speed-up2	/	297.88	23.5027624	15.9206349	1.4739448	70.9713115

5. 总结

本文在 RBK(k)算法基础上进行改进构建了最大残差块 Kaczmarz 算法及其加速版本，分析了算法的收敛性。数值实验证实了该算法的有效性。最优 k 值的选择仍是未来值得研究的方向。

参考文献

[1] Strohmer, T. and Vershynin, R. (2008) A Randomized Kaczmarz Algorithm with Exponential Convergence. *Journal of Fourier Analysis and Applications*, **15**, 262-278. https://doi.org/10.1007/s00041-008-9030-4

[2] Jiang, X., Zhang, K. and Yin, J. (2022) Randomized Block Kaczmarz Methods with K-Means Clustering for Solving Large Linear Systems. *Journal of Computational and Applied Mathematics*, **403**, Article 113828. https://doi.org/10.1016/j.cam.2021.113828

[3] 李雪芳. 基于自动控制步长的病态线性方程组算法研究[D]: [硕士学位论文]. 太原: 太原科技大学, 2013.

[4] 段云岭. 非线性方程组的解法: 局部弧长法[J]. 力学学报, 1997, 29(1): 116-122.

[5] 张广求, 余青, 洪果媛. 线性方程组分光光度法测定痤疮搽剂中甲硝唑和氯霉素的含量[J]. 中国药房, 1995, 6(1): 39-40.

[6] 蔡建新, 包尚联, 王卫东. 核医学影像中单光子定位的最小二乘方法[J]. 核电子学与探测技术, 1996, 16(3): 189-193.

[7] 陶本藻, 张勤. GPS 非线性数据处理的同伦最小二乘模型[J]. 武汉大学学报(信息科学版), 2003, 28(S1): 115-118.

[8] 胡圣荣, 戴纳新. 病态线性方程组新解法: 增广方程组法[J]. 华南农业大学学报, 2009, 30(1): 119-121.

[9] 胡川, 陈义. 非线性整体最小平差迭代算法[J]. 测绘学报, 2014, 43(7): 668-674+760.

[10] 刘经南, 曾文宪, 徐培亮. 整体最小二乘估计的研究进展[J]. 武汉大学学报(信息科学版), 2013, 38(5): 505-512.

[11] Heath, M.T. (1984) Numerical Methods for Large Sparse Linear Least Squares Problems. *SIAM Journal on Scientific and Statistical Computing*, **5**, 497-513. https://doi.org/10.1137/0905037

[12] Farebrother, R.W. (2018) Linear Least Squares Computations. Routledge.

[13] Bai, Z. and Wu, W. (2019) On Greedy Randomized Coordinate Descent Methods for Solving Large Linear Least-Squares Problems. *Numerical Linear Algebra with Applications*, **26**, e2237. https://doi.org/10.1002/nla.2237

[14] Paige, C.C. and Saunders, M.A. (1982) LSQR: An Algorithm for Sparse Linear Equations and Sparse Least Squares. *ACM Transactions on Mathematical Software*, **8**, 43-71. https://doi.org/10.1145/355984.355989

[15] Yongjie, Z. and Qin, S. (2007) A New ICCG Method of Large-Scale Sparse Linear Equation Group. 2007 *International Symposium on Microwave, Antenna, Propagation and EMC Technologies for Wireless Communications*, Hangzhou, 16-17 August 2007, 848-851. https://doi.org/10.1109/mape.2007.4393759

[16] Bai, Z. and Wu, W. (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

[17] Zhang, K., Liu, C.T., Jiang, X.L. (2024) A Greedy Randomized Block Coordinate Descent Algorithm with K-Means Clustering for Solving Large Linear Least-Squares Problems. *International Journal of Computer Science*, **51**, 511-518.

[18] 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