Advances in Pure Mathematics, 2022, 12, 260-270
https://www.scirp.org/journal/apm
ISSN Online: 2160-0384
ISSN Print: 2160-0368

A Count Sketch Maximal Weighted Residual Kaczmarz Method with Oblique Projection for Highly Overdetermined Linear Systems

Peng Zhang*, Longyan Li, Pingping Zhang

School of Science, Chongqing University of Posts and Telecommunications, Chongqing, China
Email: *zp571835001@126.com

How to cite this paper: Zhang, P., Li, L.Y. and Zhang, P.P. (2022) A Count Sketch Maximal Weighted Residual Kaczmarz Method with Oblique Projection for Highly Overdetermined Linear Systems. *Advances in Pure Mathematics*, **12**, 260-270.
https://doi.org/10.4236/apm.2022.124020

Received: February 17, 2022
Accepted: April 8, 2022
Published: April 11, 2022

Abstract

Motivated by the count sketch maximal weighted residual Kaczmarz (CS-MWRK) method presented by Zhang and Li (Appl. Math. Comput., 410, 126486), we combine the count sketch tech with the maximal weighted residual Kaczmarz Method with Oblique Projection (MWRKO) constructed by Wang, Li, Bao and Liu (arXiv: 2106.13606) to develop a new method for solving highly overdetermined linear systems. The convergence rate of the new method is analyzed. Numerical results demonstrate that our method performs better in computing time compared with the CS-MWRK and MWRKO methods.

Keywords

Count Sketch, Oblique Projection, Kaczmarz Method, Linear System

1. Introduction

We consider the following consistent linear system:

$$Ax = b \tag{1.1}$$

where $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ and x is the n-dimensional unknown vector. One of the most popular solvers for consistent linear systems (1.1) is Kaczmarz method, which was first discovered by Stefen Kaczmarz. In 2009, Strohmer and Vershynin [1] proposed the randomized Kaczmarz method with the expected exponential rate of convergence, which has triggered many scholars to research on the Kaczmarz algorithm. See [2] [3] [4]. Due to its simplicity and performance, the Kaczmarz method has many applications ranging from image reconstruction [5], distributed computing [6] to signal process [7].

Since the classical Kaczmarz method cycles through all rows of coefficient matrix A, the convergence rate depends strongly on the row index selection strategy. Mccormick [8] proposed a Maximal Weighted Residual Kaczmarz (MWRK) method, which selects the component of residual with the largest module length at each iteration. Inspired by the proof of the Greedy Randomized Kaczmarz (GRK) method [9] with remarkable convergence, Du and Gao [10] gave a new theoretical estimate for the convergence rate of the MWRK method, dependent on quantities of the coefficient matrix. Another interesting direction of studying Kaczmarz is to combine it with random sketching matrices. In the past decades, many random sketching matrices were found, such as Gaussian random projection [11], the Subsampled Randomized Hadmard Transform [12] and the count sketch [13] [14]. Zhang and Li [15] proposed a Count Sketch Maximal Weighted Residual Kaczmarz (CS-MWRK) method to solve highly overdetermined linear systems. The core of it is that the count sketch matrix can reduce the computation cost with keeping most of the information original problem [12] [16]. Experiments in [15] show that it can speed up the CPU time for solving highly overdetermined linear systems. For more sketch Kaczmarz-type methods, we refer the reader to [17] [18] [19] and the references therein.

Recently, Li, Wang, Bao and Liu [20] proposed a new Kaczmarz method with a new descent direction based on the oblique projection introduced by Constantin Popa in [21] [22], for short as KO. Using the row index selection rule in the MWRK and GRK methods, Wang, Li, Bao and Liu [23] gave two accelerated variants of the KO method: Maximal Weighted Residual Kaczmarz Method with oblique projection (MWRKO) and greedy randomized Kaczmarz method with oblique projection (GRKO). Inspired by the work of Zhang and Li [15], we combine the count sketch tech with the MWRKO method to develop a Count Sketch Maximal Weighted Residual Kaczmarz Method with oblique projection (CS-MWRKO) and obtain the convergence rate of it. Numerical experiments demonstrate that the CS-MWRKO method requires less computing time for highly overdetermined linear systems, especially for near-linear correction structure systems, compared with the CS-MWRK and MWRKO methods.

The organization of the paper is as follows. In Section 2, we propose the CS-MWRKO method and its convergence is analyzed. Section 3 contains experimental results demonstrating the efficiency of the presented method. We end this paper with some conclusions in Section 4.

We end this section with some notation. In this paper, $\langle x, y \rangle$ stands for the scalar product. $\|x\|_2$ is the Euclid norm of $x \in \mathbb{R}^n$. For a given matrix $G = (g_{ij}) \in \mathbb{R}^{m\times n}$, g_i^{T}, G^{T}, $G^{\dagger}$, $\mathbb{R}(G)$, $\mathbb{N}(G)$, $\|G\|_F$, $\sigma_i(G)$ and $\sigma_{\min}(G)$ are used to denote the ith row, the transpose, the Moore-Penrose pseudoinverse, the range space, the null space, the Frobenius norm, ith singular value and smallest nonzero singular value, respectively. We let $r^k = b - Ax^k$ to denote the kth residual vector and $r_{i_k}^k$ represents i_kth entry of r^k. $\tilde{x}$ is any solution of the system (1.1).

2. The Count Sketch Maximal Weighted Residual Kaczmarz Method with Oblique Projection

In this section, we combine the MWRKO method with the CS-MWRK method to construct a new method for (1.1), for short as CS-MWRKO, listed in **Algorithm 1**.

Algorithm 1. The CS-MWRKO method

1. Input $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m$, parameter d, x^0.
2. Output Approximate x solving $Ax = b$.
3. Create a count sketch $\mathbf{S} \in \mathbb{R}^{d\times m}$, with $d < m$ and $\tilde{A} = \mathbf{S}A, \tilde{b} = \mathbf{S}b$, and $\tilde{M}(i) = \|\tilde{a}_i\|_2^2$, $i \in [d]$.
4. Compute $i_1 = \arg\max_{i\in[d]} \frac{\left|\tilde{b}_i - \langle \tilde{a}_i, x^0\rangle\right|}{\|\tilde{a}_i\|_2}$ and $x^1 = x^0 + \frac{\tilde{b}_{i_1} - \langle \tilde{a}_{i_1}, x^0\rangle}{\tilde{M}(i_1)}\tilde{a}_{i_1}$.
5. For $k = 1,2,3,\ldots$ do until satisfy the stopping criteria.
6. Compute $i_{k+1} = \arg\max_{i\in[d]} \frac{\left|\tilde{b}_i - \langle \tilde{a}_i, x^k\rangle\right|}{\|\tilde{a}_i\|_2}$.
7. Compute $\tilde{D}_{i_k} = \langle \tilde{a}_{i_k}, \tilde{a}_{i_{k+1}}\rangle$, and $\tilde{r}_{i_{k+1}}^k = \tilde{b}_{i_{k+1}} - \langle \tilde{a}_{i_{k+1}}, x^k\rangle$.
8. Compute $\tilde{w}^{i_k} = \tilde{a}_{i_{k+1}} - \frac{\tilde{D}_{i_k}}{\tilde{M}(i_k)}\tilde{a}_{i_k}$, $h_{i_k} = \|\tilde{a}_{i_{k+1}}\|_2^2 \sin^2\langle \tilde{a}_{i_k}, \tilde{a}_{i_{k+1}}\rangle$ and $\tilde{\alpha}_{i_k}^k = \frac{\tilde{r}_{i_{k+1}}^k}{h_{i_k}}$.
9. Set $x^{k+1} = x^k + \tilde{\alpha}_{i_k}^k \tilde{w}^{i_k}$.
10. End.

Next, we introduce some lemmas used to analyze the convergence of our method.

Lemma 2.1. ([16], Theorem 1) If $S \in \mathbb{R}^{d\times m}$ is a count sketch transform with $d = (n^2+n)/(\delta\varepsilon^2)$, where $0 < \delta, \varepsilon < 1$, then we have that:

$$(1-\varepsilon)\|Ax-\tilde{x}\|_2^2 \le \|SAx\|_2^2 \le (1+\varepsilon)\|Ax-\tilde{x}\|_2^2$$

for all $x \in \mathbb{R}^n$, and:

$$(1-\varepsilon)\sigma_i(A) \le \sigma_i(SA) \le (1+\varepsilon)\sigma_i(A)$$

for all $1 \le i \le n$, hold with probability $1-\delta$.

Lemma 2.2. ([24], Lemma 1) For any vector $u \in \mathbb{R}(A^{\mathrm{T}})$, it holds that:

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

Lemma 2.3. Let S be given as in Lemma 2.1. Then $\mathbb{R}(A^{\mathrm{T}}S^{\mathrm{T}})$ is equal to $\mathbb{R}(A^{\mathrm{T}})$ with probability $1-\delta$.

Proof. It can be found in the proof of ([15], Theorem 3), we omit it here.

Lemma 2.4. The iteration sequence $\{x^k\}_{k=0}^{\infty}$ generated by the CS-MWRKO method satisfies the following equation:

$$\|x^{k+1}-\tilde{x}\|_2^2 = \|x^k-\tilde{x}\|_2^2 - \|x^{k+1}-x^k\|_2^2, \tag{2.1}$$

and the residual satisfies:

$$\tilde{r}_{i_k}^k = \tilde{b}_{i_k} - \left\langle \tilde{a}_{i_k}, x^k \right\rangle = 0, \forall k > 0, \tag{2.2}$$

$$\tilde{r}_{i_{k-1}}^k = \tilde{b}_{i_{k-1}} - \left\langle \tilde{a}_{i_{k-1}}, x^k \right\rangle = 0, \forall k > 1, \tag{2.3}$$

where $\tilde{x}$ is an arbitrary solution of the system (1.1). Especially, if $P_{\mathbb{N}(A)}\left(x^0\right) = P_{\mathbb{N}(A)}\left(\tilde{x}\right)$ then $x^k - \tilde{x} \in \mathbb{R}\left(A^{\mathrm{T}}\right)$.

Proof. Since the CS-MWRKO method is equal to the MWRKO method for sketch system $SAx = Sb$, the Equation (2.1), the Equations (2.2) and (2.3) are easily obtained by ([23], Lemma 2) and ([23], Lemma 1), respectively.

For the convergence property of the CS-MWRKO method, we establish the following theorem.

Theorem 2.5. Let $x^0 \in \mathbb{R}^n$ be an arbitrary approximation and $\tilde{x}$ is a solution of (1.1) such that $P_{\mathbb{N}(A)}\left(\tilde{x}\right) = P_{\mathbb{N}(A)}\left(x^0\right)$. Let S be given as in Lemma 2.1. Then the sequence $\left\{x^k\right\}_{k=0}^{\infty}$, generated by Algorithm CS-MWRKO, with probability $1-\delta$, obeys:

$$\left\|x^1 - \tilde{x}\right\|_2^2 \le \left(1 - \frac{(1-\varepsilon)^2}{(1+\varepsilon)^2} \frac{\sigma_{\min}^2(A)}{\|A\|_F^2}\right) \left\|x^0 - \tilde{x}\right\|_2^2,$$

and for $k = 1, 2, \cdots$:

$$\left\|x^{k+1} - \tilde{x}\right\|_2^2 \le \prod_{q=1}^{k} \rho_q \left\|x^1 - \tilde{x}\right\|_2^2,$$

where $\rho_1 = 1 - (1-\varepsilon)^2 \frac{\sigma_{\min}^2(A)}{\Delta\gamma_1}$ and $\rho_k = 1 - (1-\varepsilon)^2 \frac{\sigma_{\min}^2(A)}{\Delta\gamma_2}$, $(\forall k > 1)$, with $\Delta = \max_{j \ne k} \sin^2\left\langle \tilde{a}_j, \tilde{a}_k \right\rangle$, $\gamma_1 = \max_{1 \le i \le m} \sum_{i=1, i \ne i_1}^{m} \tilde{M}(i)$ and $\gamma_2 = \max_{1 \le i \le m} \sum_{i=1, i \ne i_k, i_{k-1}}^{m} \tilde{M}(i)$.

Proof. Based on Lemma 2.3, we can drive the convergence rate of the CS-MWRKO method following from ([23], Theroem 2) and ([15], Theroem 3]. For $k = 0$, by Equation (2.1) in Lemma 2.4, we have:

$$\begin{aligned}
\left\|x^1 - \tilde{x}\right\|_2^2 &= \left\|x^0 - \tilde{x}\right\|_2^2 - \left\|x^1 - x^0\right\|_2^2 \\
&= \left\|x^0 - \tilde{x}\right\|_2^2 - \frac{\left|\tilde{b}_{i_1} - \left\langle \tilde{a}_{i_1}, x^0 \right\rangle\right|^2}{\tilde{M}(i_1)} \\
&= \left\|x^0 - \tilde{x}\right\|_2^2 - \frac{\left|\tilde{b}_{i_1} - \left\langle \tilde{a}_{i_1}, x^0 \right\rangle\right|^2}{\tilde{M}(i_1)} \frac{\left\|\tilde{b} - \tilde{A}x^0\right\|_2^2}{\sum_{i=1}^{d} \frac{\left|\tilde{b}_i - \left\langle \tilde{a}_i, x^0 \right\rangle\right|^2}{\tilde{M}(i)} \tilde{M}(i)} \\
&\le \left\|x^0 - \tilde{x}\right\|_2^2 - \frac{\left\|\tilde{A}\left(\tilde{x} - x^0\right)\right\|_2^2}{\left\|\tilde{A}\right\|_F^2} \\
&\le \left\|x^0 - \tilde{x}\right\|_2^2 - \frac{\sigma_{\min}^2\left(\tilde{A}\right)}{\left\|\tilde{A}\right\|_F^2} \left\|x^0 - \tilde{x}\right\|_2^2
\end{aligned}$$

$$=\left(1-\frac{\sigma_{\min}^2(SA)}{\|SA\|_F^2}\right)\left\|x^0-\tilde{x}\right\|_2^2$$
$$\le\left(1-\frac{(1-\varepsilon)^2}{(1+\varepsilon)^2}\frac{\sigma_{\min}^2(A)}{\|A\|_F^2}\right)\left\|x^0-\tilde{x}\right\|_2^2,$$

with probability $1-\delta$. The second inequality comes from Lemma 2.2 and the last inequality with probability $1-\delta$ follows from Lemma 2.1. For $k=1$, it holds that:

$$\left\|x^2-\tilde{x}\right\|_2^2=\left\|x^1-\tilde{x}\right\|_2^2-\left\|x^2-x^1\right\|_2^2$$
$$=\left\|x^1-\tilde{x}\right\|_2^2-\frac{\left|\tilde{b}_{i_2}-\left\langle\tilde{a}_{i_2},x^1\right\rangle\right|^2}{\left\|\tilde{a}_{i_2}\right\|_2^2\sin^2\left\langle\tilde{a}_{i_1},\tilde{a}_{i_2}\right\rangle}$$
$$\le\left\|x^1-\tilde{x}\right\|_2^2-\frac{\left|\tilde{b}_{i_2}-\left\langle\tilde{a}_{i_2},x^1\right\rangle\right|^2}{\Delta\tilde{M}(i_2)}\frac{\left\|\tilde{b}-\tilde{A}x^1\right\|_2^2}{\sum_{i=1,i\ne i_1}^d\frac{\left|\tilde{b}_i-\left\langle\tilde{a}_i,x^1\right\rangle\right|^2}{\tilde{M}(i)}\tilde{M}(i)}$$
$$\le\left\|x^1-\tilde{x}\right\|_2^2-\frac{\left\|\tilde{b}-\tilde{A}x^1\right\|_2^2}{\Delta\sum_{i=1,i\ne i_1}^d\tilde{M}(i)}$$
$$=\left\|x^1-\tilde{x}\right\|_2^2-\frac{\left\|\tilde{A}\left(\tilde{x}-x^1\right)\right\|_2^2}{\Delta\sum_{i=1,i\ne i_1}^d\tilde{M}(i)}$$
$$\le\left(1-\frac{\sigma_{\min}^2(SA)}{\Delta\sum_{i=1,i\ne i_1}^d\tilde{M}(i)}\right)\left\|x^1-\tilde{x}\right\|_2^2 \tag{2.4}$$
$$\le\left(1-(1-\varepsilon)^2\frac{\sigma_{\min}^2(A)}{\Delta\sum_{i=1,i\ne i_1}^d\tilde{M}(i)}\right)\left\|x^1-\tilde{x}\right\|_2^2,$$

with probability $1-\delta$. Here, in the first inequality, we focus on the Equation (2.2) in Lemma 2.4. The third inequality follows from the Lemma 2.2 and the last inequality holds with probability $1-\delta$ by Lemma 2.1. Along the similar lines as in (2.4), we obtain:

$$\left\|x^{k+1}-\tilde{x}\right\|_2^2=\left\|x^k-\tilde{x}\right\|_2^2-\left\|x^{k+1}-x^k\right\|_2^2$$
$$=\left\|x^k-\tilde{x}\right\|_2^2-\frac{\left|\tilde{b}_{i_{k+1}}-\left\langle\tilde{a}_{i_{k+1}},x^k\right\rangle\right|^2}{\left\|\tilde{a}_{i_{k+1}}\right\|_2^2\sin^2\left\langle\tilde{a}_{i_k},\tilde{a}_{i_{k+1}}\right\rangle}$$
$$\le\left\|x^k-\tilde{x}\right\|_2^2-\frac{\left|\tilde{b}_{i_{k+1}}-\left\langle\tilde{a}_{i_{k+1}},x^k\right\rangle\right|^2}{\Delta\tilde{M}(i_{k+1})}\frac{\left\|\tilde{b}-\tilde{A}x^k\right\|_2^2}{\sum_{i=1,i\ne i_k,i_{k-1}}^d\frac{\left|\tilde{b}_i-\left\langle\tilde{a}_i,x^k\right\rangle\right|^2}{\tilde{M}(i)}\tilde{M}(i)}$$
$$\le\left\|x^k-\tilde{x}\right\|_2^2-\frac{\left\|\tilde{b}-\tilde{A}x^k\right\|_2^2}{\Delta\sum_{i=1,i\ne i_k,i_{k-1}}^d\tilde{M}(i)}$$

$$\le \left\|x^k-\tilde{x}\right\|_2^2-\frac{\sigma_{\min}^2\left(\tilde{A}\right)}{\Delta\sum_{i=1,i\ne i_k,i_{k-1}}^{d}\tilde{M}\left(i\right)}\left\|x^k-\tilde{x}\right\|_2^2$$

$$=\left\|x^k-\tilde{x}\right\|_2^2-\frac{\sigma_{\min}^2\left(SA\right)}{\Delta\sum_{i=1,i\ne i_k,i_{k-1}}^{d}\tilde{M}\left(i\right)}\left\|x^k-\tilde{x}\right\|_2^2$$

$$\le\left(1-\left(1-\varepsilon\right)^2\frac{\sigma_{\min}^2\left(A\right)}{\Delta\sum_{i=1,i\ne i_k,i_{k-1}}^{d}\tilde{M}\left(i\right)}\right)\left\|x^k-\tilde{x}\right\|_2^2,$$

with probability $1-\delta$. Thus, we complete the proof.

Remark 2.6. Set $\hat{\rho}_0=1-\frac{\left(1-\varepsilon\right)^2}{\left(1+\varepsilon\right)^2}\frac{\sigma_{\min}^2\left(A\right)}{\left\|A\right\|_F^2}$, $\hat{\rho}_k=1-\left(1-\varepsilon\right)^2\frac{\sigma_{\min}^2\left(A\right)}{\max_{1\le j\le d}\sum_{i=1,i\ne j}^{d}\tilde{M}\left(i\right)}$ and the convergence of the CS-MWRK method in [15] is:

$$\left\|x^{k+1}-\tilde{x}\right\|_2^2\le\prod_{q=1}^{k}\hat{\rho}_q\left\|x^1-\tilde{x}\right\|_2^2.$$

Since $\rho_0=\hat{\rho}_0$, $\rho_1\le\hat{\rho}_1$ and $\rho_k<\hat{\rho}_k,\left(\forall k>2\right)$, the CS-MWRKO method is faster than the CS-MWRK method. Based on the ([15], Remark 4), the convergence rate of CS-MWRKO is indeed larger than that of the MWRKO method. This is why the iteration numbers of the former is worse than that of the latter in numerical examples.

3. Numerical Examples and Results

Since the MWRKO [23] method is more effective than the GRK [9], GRKO [23] and MWRK [10] methods, in this section, we give some examples to illustrate the effectiveness of the CS-MWRKO method compared with the MWRKO and CS-MWRK [15] methods in terms of the iteration numbers (denoted as "IT") and computing time in seconds (denoted as "CPU time") for (1.1). We also report the iteration numbers speedup of the CS-MWRKO method against the MWRKO and CS-MWRK methods defined by:

$$\text{IT speedup1}=\frac{\text{IT of MWRKO}}{\text{IT of CS}-\text{MWRKO}},$$

$$\text{IT speedup2}=\frac{\text{IT of CS}-\text{MWRK}}{\text{IT of CS}-\text{MWRKO}}$$

and the CPU time speedup of the CS-MWRKO method against the MWRKO and CS-MWRK methods defined by:

$$\text{CPU speedup1}=\frac{\text{CPU of MWRKO}}{\text{CPU of CS}-\text{MWRKO}},$$

$$\text{CPU speedup2}=\frac{\text{CPU of CS}-\text{MWRK}}{\text{CPU of CS}-\text{MWRKO}}.$$

For the coefficient matrix A, we use the following two choices: the random matrices generated by MATLAB function rand and the other selected from the University of Florida sparse matrix collection [25]. In the following experiments, the right-hand vector $b = Ax^*$ such that the exact solution $x^* \in \mathbb{R}^n$ is a vector generated by the MATLAB function rand. We repeat 50 experiments and all the experiments start from an initial vector $x^0 = 0$, and terminate once the Relative Solution Error (RES) defined by:

$$\text{RES} = \frac{\left\| x^k - x^* \right\|_2^2}{\left\| x^* \right\|_2^2},$$

satisfies RES < 0.5^{-10} or the number of the iteration steps exceeds 100,000. All experiments presented in this section are performed in MATLAB R2018b on a personal computer with 2.00 GHz central processing unit (Intel(R) Core(TM) i5 CPU), 16.00 GB memory, and Windows operating system (Windows 10).

Example One. In this example, we report iteration numbers and CPU time for the CS-MWRKO, MWRKO and CS-MWRK methods for the randomly generated matrices in $[0,1]$, listed in Table 1. From this table, we show that the CS-MWRKO performs better than the MWRKO and CS-MWRK methods in CPU time. The CPU speedup1 is at least 7.42 and at most 20.68 and the CPU speedup2 is at least 0.95 and at most 1.15 in our experiments. For the iteration numbers, the CS-MWRKO method needs more iterations than the MWRKO method but less than the CS-MWRKO method.

Table 1. Numerical results for the CS-MWRKO, MWRKO, CS-MWRK methods with matrices generated by rand in [0, 1].

c	d	IT					CPU time				
		CS-MWRKO	MWRKO	CS-MWRK	IT speedup1	IT speedup2	CS-MWRKO	MWRKO	CS-MWRK	CPU speedup1	CPU speedup2
50000 × 50	20n	110.0400	48.0000	135.2200	0.4362	1.2288	0.2328	2.1719	0.2231	9.3289	0.9583
	30n	100.4600	48.0000	121.7400	0.4778	1.2118	0.2366	2.1284	0.2381	8.9974	1.0063
	40n	94.4400	48.0000	115.0800	0.5083	1.2185	0.2459	2.1497	0.2519	8.7408	1.0244
	60n	87.4000	48.0000	105.0600	0.5492	1.2020	0.2369	2.1334	0.2506	9.0066	1.0578
	80n	82.8200	48.0000	100.0800	0.5796	1.2084	0.2737	2.0950	0.2859	7.6530	1.0445
50000 × 100	20n	221.6200	100.0000	276.2800	0.4512	1.2466	0.4734	6.7244	0.4769	14.2033	1.0073
	50n	180.2200	99.0000	221.4800	0.5493	1.2289	0.5825	6.7803	0.6059	11.6400	1.0401
	70n	169.0200	99.0000	207.5400	0.5857	1.2279	0.6194	6.6181	0.6512	10.6852	1.0513
	80n	164.7400	98.0000	201.2400	0.5949	1.2215	0.8274	6.6137	0.8781	8.0197	1.0612
	100n	158.9000	101.0000	194.5800	0.6356	1.2245	0.7987	6.8600	0.8250	8.5884	1.0329
50000 × 150	20n	333.3000	153.0000	419.7600	0.4590	1.2594	0.6778	14.0216	0.7312	20.6865	1.0787
	50n	269.4000	153.0000	334.4600	0.5679	1.2415	1.1459	14.0741	1.2566	12.2817	1.0966
	100n	237.5000	154.0000	294.6400	0.6489	1.2405	1.6759	14.1191	1.9325	8.4246	1.1531
	120n	230.4200	155.0000	285.3800	0.6727	1.2385	1.7878	13.9763	1.8713	7.8175	1.0467
	150n	223.0200	154.0000	275.5400	0.6905	1.2354	1.8638	13.8084	2.0325	7.4251	1.0905

Example Two. In this example, we construct a random coefficient matrix with correlated rows $A \in \mathbb{R}^{50000 \times 150}$ in $[c,1]$, c from 0.1 to 0.9, to test the validity of the CS-MWRKO method with different size of count-sketch matrix S. This set of matrices was also done in [26] and [27]. From **Table 2**, we note that the CPU speedup1 is at least 6.14 and at most 12.89 and the CPU speedup2 is at least 1.08 and at most 1.61. This is, the CS-MWRKO method outperforms the MWRKO and CS-MWRK methods in term of computing time. For the iteration numbers,

Table 2. Numerical results for the CS-MWRKO, MWRKO, CS-MWRK methods with matrices generated by rand in [c, 1].

c	d	IT					CPU time				
		CS-MWRKO	MWRKO	CS-MWRK	IT speedup1	IT speedup2	CS-MWRKO	MWRKO	CS-MWRK	CPU speedup1	CPU speedup2
0.1	50n	332.6600	169.0000	439.3000	0.5080	1.3205	1.1731	14.7975	1.3019	12.6137	1.1097
	100n	285.0800	169.0000	359.1600	0.5928	1.2598	1.7353	14.7447	1.8909	8.4968	1.0896
	150n	260.9000	168.0000	327.1000	0.6439	1.2537	2.1209	14.6834	2.3059	6.9231	1.0872
0.2	50n	346.1800	172.0000	468.1800	0.4969	1.3524	1.1597	14.9500	1.3013	12.8914	1.1221
	100n	294.8600	170.0000	376.0600	0.5765	1.2753	1.7878	14.8953	1.9372	8.3316	1.0835
	150n	271.9400	171.0000	341.7000	0.6288	1.2565	2.1044	15.1097	2.3312	7.1801	1.1077
0.3	50n	365.6000	172.0000	507.0400	0.4705	1.3868	1.1922	15.0309	1.3672	12.6079	1.1467
	100n	306.8400	174.0000	399.4800	0.5671	1.3019	1.7497	15.1384	1.9566	8.6521	1.1182
	150n	282.2400	176.0000	356.3400	0.6236	1.2625	2.1437	15.3372	2.3906	7.1544	1.1151
0.4	50n	392.3400	175.0000	570.5600	0.4460	1.4542	1.2206	15.2522	1.4419	12.4954	1.1813
	100n	327.0200	175.0000	426.3800	0.5351	1.3038	1.8166	15.2134	2.0219	8.3748	1.1130
	150n	300.4200	174.0000	381.7800	0.5792	1.2708	2.2400	15.1013	2.5013	6.7416	1.1166
0.5	50n	422.9800	175.0000	652.4200	0.4137	1.5424	1.2644	15.2212	1.5391	12.0386	1.2172
	100n	350.2200	176.0000	461.9000	0.5025	1.3188	1.8631	15.2788	2.1069	8.2006	1.1308
	150n	318.8200	175.0000	408.0800	0.5489	1.2799	2.3078	15.2134	2.5713	6.5921	1.1141
0.6	50n	481.8400	173.0000	771.8800	0.3590	1.6019	1.3106	15.0697	1.6791	11.4981	1.2811
	100n	391.4200	173.0000	522.9200	0.4420	1.3359	1.9484	15.0569	2.2350	7.7277	1.1470
	150n	339.1000	173.0000	433.1400	0.5102	1.2773	2.3700	15.0628	2.6631	6.3556	1.1236
0.7	50n	589.8400	175.0000	999.9000	0.2967	1.6952	1.4544	15.1356	1.9506	10.4070	1.3411
	100n	433.6200	179.0000	596.5800	0.4128	1.3758	2.0372	15.5494	2.3959	7.6328	1.1760
	150n	356.3000	176.0000	444.1000	0.4940	1.2464	2.4341	15.2647	2.6706	6.2713	1.0971
0.8	50n	846.9000	178.0000	1483.6000	0.2102	1.7518	1.7737	15.4012	2.5597	8.6829	1.4431
	100n	474.4000	173.0000	629.3000	0.3647	1.3265	2.1837	15.0256	2.4906	6.8807	1.1405
	150n	357.9400	173.0000	445.5800	0.4833	1.2448	2.4341	15.0328	2.6959	6.1760	1.1075
0.9	50n	1431.3000	178.0000	2350.4000	0.1244	1.6421	2.5125	15.4453	3.5575	6.1474	1.4159
	100n	475.5000	179.0000	626.7800	0.3764	1.3181	2.2019	15.5350	2.4700	7.0554	1.1217
	150n	360.5400	176.0000	447.9800	0.4882	1.2425	2.4097	15.2575	2.6966	6.3317	1.1190

Table 3. Numerical results for the CS-MWRKO, MWRKO, CS-MWRK methods with sparse matrices.

Name	d	IT					CPU time				
		CS-MWRKO	MWRKO	CS-MWRK	IT speedup1	IT speedup2	CS-MWRKO	MWRKO	CS-MWRK	CPU speedup1	CPU speedup2
shar_te2-b1	5n	1267.8000	651.0000	1798.3000	0.5135	1.4184	0.2487	3.3500	0.3075	13.4874	1.2364
	8n	879.5800	644.0000	1192.9000	0.7322	1.3562	0.3481	3.2887	0.3875	9.4470	1.1131
ch6-6-b1	5n	127.2400	87.0000	175.7200	0.6837	1.3810	0.0025	0.0059	0.0028	2.3750	1.1200
	8n	104.9200	87.0000	130.5000	0.8292	1.2438	0.0037	0.0075	0.0041	2.0000	1.1081

we find that iteration numbers of the count sketch MWRK-type methods (CS-MWRK, CS-MWRKO) increase with c growing 0.1 to 0.9 and decrease with the increase of d.

Example Three. In this example, we test CS-MWRKO, MWRKO and CS-MWRK with coefficient matrices from real world data [25]. The two matrices are shar_te2-b1 with 34,320 nonzero elements and *ch*6-6-*b*1 with 900 nonzero elements. From Table 3, we see again that the CS-MWRKO method outperforms the CS-MWRK and MWRKO method in CPU time. The minimum of the CPU speedup1 is 2.00 and the maximum can reach 13.48. The minimum of the CPU speedup2 is 1.11 and the maximum is 1.24. For the iteration numbers, we get the same conclusion reported in Example One.

4. Conclusion

In this paper, we construct the count sketch maximal weighted residual Kaczmarz method with oblique projection for highly overdetermined linear systems. Numerical examples validate that our method needs less computing time compared with the MWRKO and CS-MWRK methods, especially for the system (1.1) with a linear correction structure. As we all know, there are many works about block versions of Kaczmarz-type methods [28] [29] [30] [31] [32]. We will consider the organic combination of block tech and oblique tech in future work. This topic is practically valuable and theoretically meaningful.

Acknowledgements

The authors are grateful to the anonymous referees and the Editor for their detailed and helpful comments that led to a substantial improvement to the paper. And also would like to thank Prof. Hanyu Li and Dr. Yanjun Zhang for providing Matlab codes of [15].

Funding

Longyan Li is supported by the Research and Training Program for College Students (No. A2020-171).

Conflicts of Interest

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

References

[1] 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

[2] Niu, Y.Q. and Zheng, B. (2020) A Greedy Block Kaczmarz Algorithm for Solving Large-Scale Linear Systems. *Applied Mathematics Letters*, **104**, Article ID: 106294. https://doi.org/10.1016/j.aml.2020.106294

[3] Nutini, J., Sepehry, B., Laradji, I., Schmidt, M., Koepke, H. and Virani, A. (2016) Convergence Rates for Greedy Kaczmarz Algorithms, and Faster Randomized Kaczmarz Rules Using the Orthogonality Graph. arXiv:1612.07838.

[4] Eldar, Y. and Needell, D. (2011) Acceleration of Randomized Kaczmarz Method via the Johnson-Lindenstrauss Lemma. *Numerical Algorithms*, **58**, 163-177. https://doi.org/10.1007/s11075-011-9451-z

[5] Eggermont, P.P.B., Herman, G.T. and Lent, A. (1981) Iterative Algorithms for Large Partitioned Linear Systems, with Applications to Image Reconstruction. *Linear Algebra and Its Applications*, **40**, 37-67. https://doi.org/10.1016/0024-3795(81)90139-7

[6] Elble, J.M., Sahinidis, N.V. and Vouzis, P. (2010) GPU Computing with Kaczmarz's and Other Iterative Algorithms for Linear Systems. *Parallel Computing*, **36**, 215-231. https://doi.org/10.1016/j.parco.2009.12.003

[7] Lorenz, D.A., Wenger, S., Schopfer, F. and Magnör, M. (2014) A Sparse Kaczmarz Solver and a Linearized Bregman Method for Online Compressed Sensing. 2014 *IEEE International Conference on Image Processing*, Paris, 27-30 October 2014, 1347-1351. https://doi.org/10.1109/ICIP.2014.7025269

[8] McCormick, S.F. (1977) The Methods of Kaczmarz and Row Orthogonalization for Solving Linear Equations and Least Squares Problems in Hilbert Space. *Indiana University Mathematics Journal*, **26**, 1137-1150. https://doi.org/10.1512/iumj.1977.26.26090

[9] Bai, B.Z. and Wu, W.T. (2018) On Greedy Randomized Kaczmarz Method for Solving Large Sparse Linear Systems. *SIAM Journal on Scientific Computing*, **40**, A592-A605. https://doi.org/10.1137/17M1137747

[10] Du, K. and Gao, H. (2019) A New Theoretical Estimate for the Convergence Rate of the Maximal Weighted Residual Kaczmarz Algorithm. *Numerical Mathematics: Theory, Methods and Applications*, **12**, 627-639. https://doi.org/10.4208/nmtma.OA-2018-0039

[11] Boutsidis, C., Drineas, P. and Magdon-Ismail, M. (2014) Near-Optimal Column-Based Matrix Reconstruction. *SIAM Journal on Computing*, **43**, 687-717. https://doi.org/10.1137/12086755X

[12] Woodruff, D.P. (2014) Sketching as a Tool for Numerical Linear Algebra. *Foundations and Trends in Theoretical Computer Science*, **10**, 1-157.

[13] Charikar, M., Chen, K. and Farach-Colton, M. (2002) Finding Frequent Items in Data Streams. *International Colloquium on Automata, Languages, and Programming* 2002, Málaga, 8-13 July 2002, 693-703. https://doi.org/10.1007/3-540-45465-9_59

[14] Thorup, M. and Zhang, Y. (2012) Tabulation-Based 5-Independent Hashing with Applications to Linear Probing and Second Moment Estimation. *SIAM Journal on Computing*, **41**, 293-331. https://doi.org/10.1137/100800774

[15] Zhang, Y.J. and Li, H.Y. (2021) A Count Sketch Maximal Weighted Residual Kaczmarz Method for Solving Highly Overdetermined Liner Systems. *Applied Mathematics and Computation*, **410**, Article ID: 126486. https://doi.org/10.1016/j.amc.2021.126486

[16] Clarkson, K.L. and Woodruff, D.P. (2017) Low-Rank Approximation and Regression in Input Sparsity Time. *Journal of the ACM*, **63**, Article No. 54. https://doi.org/10.1145/3019134

[17] Wu, N.C. and Xiang, H. (2021) Semiconvergence Analysis of the Randomzied Row Iterative Method and Its Extended Variants. *Numerical Linear Algebra with Applications*, **28**, Article No. e2334. https://doi.org/10.1002/nla.2334

[18] Gower, R.M., Molitor, D., Moorman, J. and Needell, D. (2021) On Adaptive Sketch-and-Projection for Solving Linear Systems. *SIAM Journal on Matrix Analysis and Applications*, **42**, 954-989. https://doi.org/10.1137/19M1285846

[19] Gower, R.M. and Richtárik, P. (2015) Randomized Iterative Methods for Linear Systems. *SIAM Journal on Matrix Analysis and Applications*, **36**, 1660-1690. https://doi.org/10.1137/15M1025487

[20] Li, W.G., Wang, Q.F., Bao, W.B. and Liu, L. (2021) On Kaczmarz Methods with Oblique Projection for Solving Large Overdetermined Lienar Systems. arXiv: 2106.13368.

[21] Popa, C., Preclik, T., Köstler, H. and Rüde, U. (2012) On Kaczmarz's Projection Iteration as a Direct Solver for Linear Least Squares Problems. *Linear Algebra and its Applications*, **436**, 389-404. https://doi.org/10.1016/j.laa.2011.02.017

[22] Popa, C. (2012) Projection algorithms Classical Results and Developments. Lap Lambert Academic Publishing, Saarbrücken.

[23] Wang, F., Li, W.G., Bao, W.B. and Liu, L. (2021) Greedy Randomzied and Maximal Weighted Residual Kaczmarz Methods with Oblique Projection, arXiv: 2106.13606.

[24] Zhang, J.H. and Guo, J.H. (2020) On Relaxed Greedy Randomized Coordinate Descent Methods for Solving Large Linear Least-Squares Problems. *Applied Numerical Mathematics*, **157**, 372-384. https://doi.org/10.1016/j.apnum.2020.06.014

[25] Davis, T.A. and Hu, Y. (2011) The University of Florida Sparse Matrix Collection. *ACM Transactions on Mathematical Software*, **38**, Article No. 1. https://doi.org/10.1007/s11075-021-01104-x

[26] Wu, W.T. (2021) On Two-Subspace Randomized Extended Kaczmarz Method for Solving Large Linear Least-Squares Problems. *Numerical Algorithms*, **89**, 1-31. https://doi.org/10.1007/s11075-021-01104-x

[27] Needell, D. and Ward, R. (2013) Two-Subspace Projection Method for Coherent Overdetermined Systems. *Journal of Fourier Analysis and Applications*, **19**, 256-269. https://doi.org/10.1007/s00041-012-9248-z

[28] Chen, J.Q. and Huang, Z.D. (2022) On a Fast Deterministic Block Kaczmarz Method for Solving Large-Scale Linear Systems. *Numerical Algorithms*, **89**, 1007-1029. https://doi.org/10.1007/s11075-021-01143-4

[29] Necoara, I. (2019) Faster Randomzied Block Kaczmarz Algorithms. *SIAM Journal on Matrix Analysis and Applications*, **40**, 1425-1452. https://doi.org/10.1137/19M1251643

[30] Du, K., Si, W.T. and Sun, X.H. (2020) Randomized Extended Average Block Kaczmarz for Solving Least Squares. *SIAM Journal on Matrix Analysis and Applications*, **42**, A3541-A3559. https://doi.org/10.1137/20M1312629

[31] Li, W., 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**, 2954-2665. https://doi.org/10.4236/jamp.2021.911189

[32] Liao, Y., Yin, F. and Huang, G.X. (2021) A Relaxed Greedy Block Kaczmarz Method for Solving Large Consistent Linear Systems. *Journal of Applied Mathematics and Physics*, **9**, 3032-3044. https://doi.org/10.4236/jamp.2021.912196