Block Kaczmarz–Motzkin Method via Mean Shift Clustering

Yimou Liao 1, Tianxiu Lu 1,2,* and Feng Yin 3

1 College of Mathematical and Statistics, Sichuan University of Science and Engineering, Zigong 643000, China;
liaoymou596@163.com
2 Key Laboratory of Higher Education of Sichuan Province for Enterprise Informationalization and Internet of Things, Zigong 643000, China
3 College of Mathematics and Physics, Chengdu University of Technology, Chengdu 643000, China; fyuin@cdut.edu.cn
* Correspondence: lubeeltx@suse.edu.cn; Tel.: +86-13408138464

Abstract: Solving systems of linear equations is a fundamental problem in mathematics. Combining mean shift clustering (MS) with greedy techniques, a novel block version of the Kaczmarz–Motzkin method (BKMS), where the blocks are predetermined by MS clustering, is proposed in this paper. Using a greedy strategy, which collects the row indices with the almost maximum distance of the linear subsystem per iteration, can be considered an efficient extension of the sampling Kaczmarz–Motzkin algorithm (SKM). The new method linearly converges to the least-norm solution when the system is consistent. Several examples show that the BKMS algorithm is more efficient compared with other methods (for example, RK, Motzkin, GRK, SKM, RBK, and GRBK).

Keywords: consistent linear system; Kaczmarz–Motzkin; clustering method; convergence property

MSC: 65F10; 65F25; 90C25; 15A06

1. Introduction

This paper is concerned with the approximate solution of a large-scale linear system of equations of the form

\[ Ax = b, \ A \in \mathbb{R}^{m \times n}, \ b \in \mathbb{R}^{m}, \]  

where \( x \in \mathbb{R}^{n} \) need to be determined. If \( m > n \), the linear systems (1) with such thin (or tall) coefficient matrices \( A \) are overdetermined and are often inconsistent. In this case, we are interested in the computation of the least-squares solution of systems (1) by

\[ x_{LS} := \arg \min_{x} \| Ax - b \|_{2}^{2}, \]  

where \( \| \cdot \|_{2} \) denotes the Euclidean norm. If \( m < n \), the linear system (1) with such a fat (or flat) matrix \( A \) is underdetermined. If the linear system is consistent and has many solutions, then the least Euclidean-norm solution

\[ x_{LN} := \arg \min_{x} \| x \|_{2}, \ Ax = b \]  

will often be considered. In this paper, we focus on the approximate of the least-norm solution when (1) is consistent.

The Kaczmarz method in [1] is a popular algorithm for solving the linear system (1), which is called the algebraic reconstruction technique (ART), and has a large range of fields of applications, such as image reconstruction in computerized tomography [2,3], signal processing [4], and distributed computing [5,6]. In the classical Kaczmarz method [1], all rows of the matrix \( A \) are circularly passed according to the given order, and the current iteration point is orthogonally projected to the hyperplane formed by the index row for the...
next iteration. For a given initial solution \( x_0 \), the iteration scheme of the Kaczmarz method can be presented as

\[
x_{k+1} = x_k + \frac{b_k - A^T x_k}{\|A^T \|_2^2} (A^T)^T,
\]

where \( i_k = \text{mod}(k, m) + 1, k = 0, 1, \ldots \), \( A^i \) denotes the \( i \)th row of \( A \), \( b_i \) presents the \( i \)th element of \( b \), and \( A^T \) denotes the transpose of \( A \).

For the consistent linear system (1), Strohmer and Vershynin [7] in 2009 presented a randomized Kaczmarz algorithm (RK) with expected exponential convergence. The RK algorithm selects each iteration row with the row index \( i_k = \|A b\|_2^2 / \|A\|_F^2 \). A theoretical analysis showed that, compared with the selection of a row in the natural order, randomly selecting a row of iterations can greatly improve the convergence speed. In [8], Bai and Wu designed a greedy randomized Kaczmarz (GRK) algorithm. GRK uses a greedy probability strategy, which is derived from the maximum distance rule, to grasp larger entries of the residual vector, and the work row with a larger residual error is determined. The theoretical analysis shows that the GRK converges to the least norm solution with the expected exponential, and the numerical experiments illustrate that GRK converges much faster than RK. In [9], De Loera proposed a sampling Kaczmarz–Motzkin (SKM) for linear feasibility problems by combining the RK method with the Motzkin method [10]. This method mainly consists of two stages. The row index set \( \{1, 2, \ldots , m\} \) of matrix \( A \) is divided into multiple subsets \( T = \{\tau_1, \tau_2, \ldots , \tau_p\} \) for the first stage, and the second stage involves determining the working row \( i_k = \arg \max_{i \in \tau_k} \{ |b_{ik} - A^i x_k|^2 \} \) by uniformly extracting one \( (\tau_k, k \in [p]) \) in each iteration, and then updating \( x_k \) by (4). In particular, the SKM method overcomes the weakness of the RK and Motzkin methods. For example, the RK method may be slow to converge because it does not utilize a greedy strategy, and the calculations of all residual vectors are expensive for the Motzkin method. However, the convergence rate of the SKM method may be slow since it enforces only one constraint for the current iteration.

The block Kaczmarz methods, which utilize the multi-row constraints, have received extensive attention for their fast convergence properties. The block iterative methods were first presented by Elfving [11] and Eggermont et al. [12] to solve (1). Then, Needell et al. [13] proposed a randomized block Kaczmarz (RBK) algorithm to solve the least-squares problem by selecting a subsystem from the predetermined partition at random, which linearly converges to the least-norm solution in expectation. The randomized block Kaczmarz algorithm iterates the form

\[
x_{k+1} = x_k + A_{\tau_k}^\dagger (b_{\tau_k} - A_{\tau_k} x_k),
\]

where \( A_{\tau_k} \) and \( b_{\tau_k} \) are the submatrix and subvector of \( A \) and \( b \), respectively, \( \tau_k \subseteq \{1, 2, \ldots , m\} \) and \( A_{\tau_k}^\dagger \) represent the Moore–Penrose pseudoinverse of subsystem matrix \( A_{\tau_k} \). Although the existence of this “good” paving of all rows is theoretically guaranteed, it is not always effortless to find it. Liu and Gu in [14] designed a randomized greedy block Kaczmarz method to choose a “good” target block by embedding a greedy strategy in the predetermined subsystems, which was derived from the probability criterion proposed by Bai and Wu in [8]. The convergence theory of GRBK shows that the iterative solution sequence linearly converges to the least-norm solution in expectation. In addition, the partitioning method may not be a good partition since it only focuses on the average partitioning of the row index set. Many variants of the block Kaczmarz have recently received extensive attention. For example, literature studies [15–21] and references therein.

By exploring the structural properties of the coefficient matrix \( A \), some criteria can be used to determine the row partitions (see [13,14,16,22]). In the context of data science, the mean shift clustering (MS) is an iterative algorithm based on kernel density estimation [23], which is widely used for data clustering. So, the combination of the clustering algorithm and greedy technique is extremely meaningful. Inspired by this, to improve the efficiency of block Kaczmarz, we constructed a novel block Kaczmarz–Motzkin method (BKMS\[\delta\],
we ignore the assumption that the linear system is inconsistent. The second term on the right-hand side of the inequality is zero. So, we remark the convergence factor of RBK as exponentially converging to the least-squares solution $x^\ast$. Moreover, unlike k-means clustering [16,24], it only needs to initialize the bandwidth’s $\delta \in (0,1]$ automatic partitions instead of pre-specifying the number of partitions $\kappa$, which depends on the initialized cluster centers.

This paper is organized as follows. The RBK method in [13] and the GRBK method in [14] are summarized in Section 2. Section 3 introduces the MS clustering briefly, and then the BKMS method is proposed and its convergence theory is constructed when (1) is consistent. Section 4 reports several examples and applications. We complete this work with some conclusions and discuss future work in Section 5.

2. Block Kaczmarz Algorithm and Its Variants

For a matrix $Q = (q_{ij}) \in \mathbb{R}^{m \times n}$, we indicate by $Q^i$, $Q_{\tau_i}$, $\|Q\|_2 := \max_{x \neq 0} \|Qx\|_2 / \|x\|_2$, $\|Q\|_F := \sqrt{\sum_{i=1}^m \sum_{j=1}^n |q_{ij}|^2}$ and $Q^i$ the $i$th row and a row submatrix ($\tau_i$ is a row indicator block), the spectral norm, Frobenius norm, and the transpose of the $Q$, respectively. If any two rows of $Q$ are independently and identically distributed (IID), the correlation coefficient of $Q$ is given by

$$
\mu_k = \frac{|Q^i(Q^p)^T|}{\|Q^i\|_2\|Q^p\|_2}, \quad l \neq p \in \{1, 2, \cdots, m\}.
$$

In addition, $\sigma_{\max}(Q)$ and $\sigma_{\min}(Q)$ are the largest and smallest positive singular values of the $Q$, respectively. Moreover, $p_i$ denotes the $i$th component of a vector $p = (p_1) \in \mathbb{R}^m$. For constant $c \in \mathbb{R}$, we call $[c]$ the largest integer not exceeding $c$.

2.1. The RBK Algorithm

The RBK method [13] is summarized in Algorithm 1, which is described by Needell and Tropp to solve the overdetermined least-squares problem (2).

Algorithm 1 The randomized block Kaczmarz method (RBK)

**Input:** $A \in \mathbb{R}^{m \times n}$ is standardized (i.e., $\|A^i\|_2 = 1, i \in [m]$), $b \in \mathbb{R}^m$, $x_0 \in \mathbb{R}^n$, $l \in \mathbb{N}^+$ and partition $T = \{\tau_1, \tau_2, \cdots, \tau_p\}_{p \leq m}$

**Output:** $x_l$

for $k = 0, 1, \ldots l - 1$ do

Select a block $\tau_k$ uniformly at random from $T$

Update $x_{k+1}$ by (5)

end for

The restatement of Lemma 1 in [13] guarantees the linear convergence of the RBK method.

**Lemma 1.** Assume that the overdetermined linear system (1) with $A$ is of full rank and row normalized (i.e., $\|A^i\|_2 = 1, i \in [m]$). Let $\alpha \leq \sigma^2_{\min}(A_T)$ and $\beta \geq \sigma^2_{\max}(A_T)$ for each $\tau \in T$. Given an initial vector $x_0 \in \mathcal{R}(A^T)$, then the sequence $\{x_k\}_0^\infty$ generated by the RBK method exponentially converges to the least-squares solution $x^\ast = A^Tb$ of (1) in expectation. Furthermore,

$$
\mathbb{E}\|x_k - x^\ast\|_2^2 \leq \left(1 - \frac{\sigma^2_{\min}(A)}{\beta m}\right)^k \|x_0 - x^\ast\|_2^2 + \frac{\beta}{\alpha} \frac{\|Ax^\ast - b\|_2^2}{\sigma^2_{\min}(A)}.
$$

According to Lemma 1, the convergence rate of RBK is limited by $\sigma^2_{\min}(A)$. If $A$ is well-conditioned, the upper bound of the convergence error of RBK is large. In addition, we ignore the assumption that the linear system is inconsistent. The second term on the right-hand side of the inequality is zero. So, we remark the convergence factor of RBK as
\[ \rho_{RBK} = 1 - \frac{\sigma_{\min}^2(A)}{\beta m}. \]

2.2. The GRBK Algorithm

Based on the motivation of the GRK method, Liu and Gu in [14] applied the greedy sampling to the RBK method to replace uniform sampling, and proposed the GRBK method. Algorithm 2 summarizes the GRBK method to solve (1), as follows.

**Algorithm 2** The greedy randomized Kaczmarz block method (GRBK)

**Input:** \( A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m, x_0 \in \mathbb{R}^n, l \in \mathbb{N}^+ \) and partition \( T = \{ \tau_1, \tau_2, \cdots, \tau_p \} \).

**Output:** \( x_l \)

for \( k = 0, 1, \ldots, l - 1 \) do

Compute

\[ \epsilon_k = \frac{1}{2} \left( \frac{1}{\| b - A x_k \|_2^2} \max_{i \in T} \left\{ \frac{|b_{\tau_i} - A_{\tau_i} x_k|^2}{\|A_{\tau_i}\|_2^2} \right\} + \frac{1}{\| A \|_F^2} \right) \]

Determine the partition of the indicator set

\[ U_k = \{ \tau_i \mid |b_{\tau_i} - A_{\tau_i} x_k|^2 \geq \epsilon_k \| b \|_2 \| A_{\tau_i} \|_2^2 \} \]

Compute the block \( \tilde{r}_{\tau_i}^k \) of the entry \( \tilde{r}_i^k \) according to

\[ \tilde{r}_{\tau_i}^k = \begin{cases} b_{\tau_i} - A_{\tau_i} x_k, & \text{if } \tau_i \in U_k, \\ 0, & \text{otherwise} \end{cases} \]

Select \( \tau_i \in U_k \) with probability

\[ P_i(\text{block} = \tau_i) = \frac{|\tilde{r}_{\tau_i}^k|^2}{\| \tilde{r}_i^k \|_2^2} \]

Update \( x_{k+1} \) by (5)

end for

According to Algorithm 2, we consider that, if there is only a row in each block, then GRK in [8] is a special case of the GRBK method. Liu and Gu proved that the GRBK algorithm has a lower convergence upper bound than RBK and demonstrated the effectiveness of GRBK with several numerical experiments. The result in [14] is restated as follows.

**Lemma 2.** For a fix partition \( T = \{ \tau_1, \tau_2, \cdots, \tau_p \} \) of the row indices \( \{1, 2, \cdots, m\} \), we assume that \( \xi = \min_{\tau \in T} \| A \|_F^2 \) and \( \beta \geq \sigma_{\max}^2(A_{\tau}) \) for each \( \tau \in T \). Given an initial vector \( x_0 \in \mathbb{R}^n(A^T) \), then the sequence \( \{x_k\}_0^n \) generated by the GRBK method exponentially converges to the least norm solution \( x_* = A^\dagger b \) in expectation when (1) is consistent. Moreover, it satisfies

\[ E\|x_1 - x_*\|_2^2 \leq \left( 1 - \xi \frac{\sigma_{\min}^2(A)}{\beta \| A \|_F^2} \right) \|x_0 - x_*\|_2^2 \]

and

\[ E\|x_{k+1} - x_*\|_2^2 \leq \left( 1 - \gamma \xi \frac{\sigma_{\min}^2(A)}{\beta \| A \|_F^2} \right)^{k-1} \left( 1 - \xi \frac{\sigma_{\min}^2(A)}{\beta \| A \|_F^2} \right) \|x_0 - x_*\|_2^2, \]

where \( \gamma = \frac{1}{2} \left( \frac{\| A \|_F^2}{\| A \|_F^2 - \xi} + 1 \right). \]
By Lemma 2, the following estimates bound the convergence factor of the GRBK method

$$\rho_{GRBK} = 1 - \frac{\xi}{2} \left( \frac{\|A\|_F^2}{\|A\|_F^4} + 1 \right) \frac{\sigma_{\text{min}}^2(A)}{\beta \|A\|_F^2},$$

which is less than $\rho_{RBK}$ when $A$ is row normalized.

3. Block Kaczmarz–Motzkin Methods with Mean Shift Clustering

The greedy strategy in the SKM method ensures that the entry with the largest residual is prioritized. We hope to exploit the correlations between subsystem matrices to determine the row partition of the coefficient matrix $A$ in an efficient way. In the framework of big data, the mean shift method (MS) in [25] is one of the clustering algorithms that divides the data into different categories. Therefore, the combination of the MS clustering and the Kaczmarz method with greedy techniques comes naturally. Inspired by this, a novel block Kaczmarz–Motzkin method based on MS clustering is proposed. In this section, we first restate the sketch of the MS clustering, then design a new, more realistic block Kaczmarz–Motzkin method and construct its convergence theory.

3.1. The Mean Shift Clustering

The MS clustering is an iterative algorithm based on kernel density estimation. It mainly consists of three steps, as follows. The first step is to select a point $\tilde{x}_i, i \in \{1, 2, \cdots m\}$ randomly from the unclassified data items $X = \{\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_m\}$ as the center point $y_j$. The second step is to find out all the data points in the circle entered on $y_j$ and the bandwidth $\delta/2$ as the radius. It can be written as

$$S_{\delta}(y_j) = \{\tilde{x}_i : \|y_j - \tilde{x}_i\|_2 < \delta/2\}, j = 1.$$

Considering these points belong to the cluster $C_j$, calculate the vector from the center point $y_j$ to each element $x_i$ in the set $S_{\delta}(y_j)$, and add these vectors to obtain the mean shift vector $m(y_j)$, as follows

$$m(y_j) = \frac{\sum_{i=1}^n \tilde{x}_i g\left(\|y_j - \tilde{x}_i\|_2 / \delta\right)}{\sum_{i=1}^n g\left(\|y_j - \tilde{x}_i\|_2 / \delta\right)} - y_j,$$

where $\tilde{x}_i \in S_{\delta}(\tilde{x})$ and the Gaussian kernel function $g(\cdot)$ are designed to assign different weights to each data point. The center point is moved in the direction of the mean shift vector $m(y_j)$ by $|m(y_j)|$. Update the current center point by $y_{j+1} = m(y_j) + y_j (j = 1, 2, \cdots)$, then repeat the above process until the iterative sequence $\{y_j\}_{j \geq 1}$ satisfies $\|y_{j+1} - y_j\|_2^2 < \epsilon$, where $\epsilon$ is a user-preset threshold. The last step is to repeat the above two steps until all points are classified. Algorithm 3 summarizes the MS algorithm in [26] for data clustering.

According to Algorithm 3, one knows that, unlike the k-means algorithm in [16] and the random partition method in [13], the MS clustering algorithm does not require us to predetermine the number of clusters. For details, we refer to [25,26].
Algorithm 3 The mean shift method for clustering (MS)

Input: A dataset $X = \{\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_m\}$ containing $m$ data items, the threshold $\epsilon$ and the bandwidth $\delta > 0$.

Output: Estimate cluster centers $y^*_1, y^*_2, \cdots, y^*_k$, where $k$ is the number of clusters.

for $i = 1, 2, \cdots, n$ do

Select an initial data point $\tilde{x}_j$ randomly from the data set $X$, let $y_j = \tilde{x}_j$ and $j \leftarrow 1$.

repeat

Estimate the mean shift vector by

$$m(y_j) = \frac{\sum_{i=1}^{n} \tilde{x}_i g(\|y_j - \tilde{x}_i\|_2^2)}{\sum_{i=1}^{n} g(\|y_j - \tilde{x}_i\|_2^2)} - y_j$$

Update the cluster centers

$$y_{j+1} = m(y_j) + y_j$$

until $\|y_{j+1} - y_j\|_2^2 \leq \epsilon$

$y^*_j \leftarrow y_j$

end for

Merge the cluster centers with distances smaller than $\delta$

3.2. Block Kaczmarz–Motzkin Method with Mean Shift

The following describe in detail how to combine MS clustering and the Kaczmarz–Motzkin method. Considering the row partitions of $A$, we first construct the dataset $X = \{F(\text{row}), \mu_k\} \in \mathbb{R}^{m \times 2}$. The $\mu_k$ is calculated by

$$\mu_k = \frac{|A^i(A^i)^T|}{\|A^i\|_2^2 \|A^i\|_2^2},$$

where $i \neq I \in \{1, 2, \cdots, m\}$ and the superscript $I$ refer to selecting an index randomly from the set formed by the maximum row norm of $A$ (if there are multiple maximum row norm values). $F(\text{row})$ means linear mapping from the row index set of $A$. By implementing the MS algorithm, we have partition $\hat{T} = \{\tau_1, \tau_2, \cdots, \tau_p\}$ of $\{1, 2, \cdots, m\}$, i.e.,

$$A = [A_{\tau_1}; A_{\tau_2}; \cdots, A_{\tau_p}]$$

and $b = [b_{\tau_1}; b_{\tau_2}; \cdots, b_{\tau_p}]$.

Similar to SKM, at the $k$th iteration, if $\|b_{\tau_i} - A_{\tau_i}x_k\|_2^2 > \|b_{\tau_s} - A_{\tau_s}x_k\|_2^2$, then the $\tau_i$ block is to be sampled rather than block $\tau_s$ for all $i \neq s \in [p]$. Considering that a block extracted may be very large, we introduce an almost-maximum distance criterion in one iteration to control the iterative index set. Following this idea, the block Kaczmarz–Motzkin method with MS clustering is summarized in Algorithm 4.
Algorithm 4 Block Kaczmarz–Motzkin method with mean shift (BKMS).

Input: $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $x_0 \in \mathbb{R}^n$, $I \in \mathbb{N}^+$, the bandwidth $\delta \in (0, 1]$, and the parameter $\eta \in (0, 1]$.
Output: $x_f$

Data set $\mathcal{X}$ is obtained by data preprocessing on $A$

Apply the MS($\delta$) method to $\mathcal{X}$ to obtain partition $\hat{\mathcal{F}} = \{\hat{\tau}_1, \hat{\tau}_2, \ldots, \hat{\tau}_p\}$ from the set of row index sets $\{1, 2, \ldots, m\}$ for $k = 0, 1, \ldots, l - 1$ do

1. Determine the partition of the indicator set $\mathcal{R}_k = \{\tau_k | \tau_k = \arg \max_{1 \leq j \leq p} \|r_j\|_2^2\}$

2. Compute $\epsilon_k = \eta \max_{j \in \tau_k} \left\{ \frac{|b_k - A\tau_k|_2^2}{\|A\|_2^2} \right\}$

3. Choose $\tau_k = \{i_k | |b_k - A\tau_k|_2^2 \geq \epsilon_k \|A\|_2^2\}$

4. Update $x_{k+1} = x_k + A_{\tau_k}^+ (b_{\tau_k} - A_{\tau_k} x_k)$

Remark 1. We compress the row index set of $A$ to the interval of $\mu_k$, i.e., $[1, m] \xrightarrow{\mathcal{F}} [\mu_k^{\min}, \mu_k^{\max}]$, and remark that the $\mathcal{F}$ is a linear map. In fact, if $\forall x, y \in \{1, 2, \cdot \cdot \cdot, m\}$ and $a, b \in \mathbb{R}$, the following conditions hold:

$$\mathcal{F}(ax + by) = a\mathcal{F}(x) + b\mathcal{F}(y).$$

Now, the convergence of the BKMS algorithm is analyzed. The fact guaranteed in [2] is required.

Lemma 3. If $A \in \mathbb{R}^{m \times n}$ is a nonzero real matrix, then

$$\sigma_{\min}^2(A) \|u\|_2^2 \leq \|Au\|_2^2 \leq \sigma_{\max}^2(A) \|u\|_2^2$$

for any $u \in \text{range}(A^\top)$.

The following theorem gives the convergence rate of the BKMS Algorithm 4 when (1) is consistent.

Theorem 1. Let $\{\hat{\tau}_p\}_{p \geq 1}$ be the target blocks generated by Algorithm 3. Assume that $\zeta \geq \max_{\tau} \|A_{\tau}\|_2^2$ and $\alpha \geq \max_{\tau} |\tau|$ for each $\tau \in \{\hat{\tau}_p\}_{p \geq 1}$, where $|\tau|$ is the cardinality of $\tau$.

Given an initial guess $x_0 \in \mathcal{R}(A^\top)$, then the sequence $\{x_k\}_0^n$ generated by the BKMS Algorithm 4 converges linearly to the least norm solution $x^* = A^+b$ of (1). Furthermore,

$$\|x_1 - x^*\|_2^2 \leq \left[1 - \frac{\eta \|A_{\hat{\tau}_0}\|_2^2 \sigma_{\min}^2(A)}{\alpha \zeta \sigma_{\max}^2(A_{\hat{\tau}_0})}\right] \|x_0 - x^*\|_2^2$$

and

$$\|x_{k+1} - x^*\|_2^2 \leq \left[1 - \frac{\eta \|A_{\hat{\tau}}\|_2^2 \sigma_{\min}^2(A)}{\alpha (m - N_{k-1}) \zeta \sigma_{\max}^2(A_{\hat{\tau}})}\right] \|x_k - x^*\|_2^2, \quad k = 1, 2, \cdot \cdot \cdot .$$

Then,

$$\|x_{k+1} - x^*\|_2^2 \leq \left[1 - \frac{\eta \|A_{\hat{\tau}}\|_2^2 \sigma_{\min}^2(A)}{\alpha (m - N_{k-1}) \zeta \sigma_{\max}^2(A_{\hat{\tau}})}\right]^k \left[1 - \frac{\eta \|A_{\hat{\tau}}\|_2^2 \sigma_{\min}^2(A)}{\alpha \zeta \sigma_{\max}^2(A_{\hat{\tau}})}\right] \|x_0 - x^*\|_2^2,$$ (6)

where $\sigma_{\max}^2(A_{\tau})$ is the largest nonzero singular value of $A_{\tau}$ and $N_{k-1}$ is the cardinality of the set $\hat{\tau}_{k-1}$.
Proof. According to the (5) of Algorithm 4

\[ x_{k+1} - x_k = A^\dagger_{\tau_k} (b_{\tau_k} - A_{\tau_k} x_k). \]

Since \( A_{\tau_k} x_s = b_{\tau_k} \), one can see that

\[ x_{k+1} - x_k = A_{\tau_k}^\dagger (A_{\tau_k} x_s - A_{\tau_k} x_k) = A_{\tau_k}^\dagger A_{\tau_k} (x_s - x_k), \]

which implies that \( x_{k+1} - x_k \) belongs to the column space of matrix \( A_{\tau_k}^\dagger A_{\tau_k} \).

From the iteration scheme of the BKMS,

\[
A_{\tau_k} (x_{k+1} - x_s) = A_{\tau_k} [x_k + A_{\tau_k}^\dagger (b_{\tau_k} - A_{\tau_k} x_k)] - A_{\tau_k} x_s
= A_{\tau_k} x_k + A_{\tau_k} A_{\tau_k}^\dagger b_{\tau_k} - A_{\tau_k} A_{\tau_k}^\dagger A_{\tau_k} x_s - A_{\tau_k} x_s
= A_{\tau_k} x_k + A_{\tau_k} x_s - A_{\tau_k} x_k - A_{\tau_k} x_s
= 0.
\]

(7)

One can obtain that \( A_{\tau_k}^\dagger A_{\tau_k} (x_{k+1} - x_s) = 0 \), which shows \( x_{k+1} - x_s \) is orthonormal to \( A_{\tau_k}^\dagger A_{\tau_k} \). So, \((x_{k+1} - x_k)^T (x_{k+1} - x_s) = 0\) is clearly established.

Thus, the following formula can be obtained

\[
\|x_{k+1} - x_s\|_2^2 = \|x_k - x_s\|_2^2 - \|x_{k+1} - x_k\|_2^2
= \|x_k - x_s\|_2^2 - \|A_{\tau_k}^\dagger A_{\tau_k} (x_k - x_s)\|_2^2. \tag{8}
\]

On the other hand, according to the algorithm flow, we know that

\[
\|r_{\tau_k}^T\|_2^2 = \max_{i \in [p]} \|r_{\tau_k}^i\|_2^2 \quad \text{and} \quad \frac{\|r_{\tau_k}^i\|_2^2}{\|A_{\tau_k}^\dagger\|_F} = \max_{\tau_k \in R_k} \frac{\|r_{\tau_k}^i\|_2^2}{\|A_{\tau_k}\|_F}
\]

So,

\[
\|A_{\tau_k}^\dagger A_{\tau_k} (x_k - x_s)\|_2^2 \geq \sigma^2_{\min}(A_{\tau_k}^\dagger) \|A_{\tau_k} (x_k - x_s)\|_2^2 = \frac{1}{\sigma^2_{\max}(A_{\tau_k})} \sum_{i \in \tau_k} |A^i(x_k - x_s)|^2
= \frac{1}{\sigma^2_{\max}(A_{\tau_k})} \sum_{i \in \tau_k} |r_{\tau_k}^i|^2
\geq \frac{1}{\sigma^2_{\max}(A_{\tau_k})} \sum_{i \in \tau_k} \eta \max_{j \in \tau_k} \|A^j\|_2 |r_{\tau_k}^j|^2
\geq \eta \|A_{\tau_k}\|_F^2 \frac{\max_{j \in \tau_k} |r_{\tau_k}^j|^2}{\|A_{\tau_k}\|_F^2}.
\]

Let \( \alpha \geq \max_{i \in [p]} |\tau_i| \), where \( |\tau_i| \) is the cardinality of \( \tau_i \), then

\[
\max_{j \in \tau_k} |r_{\tau_k}^j|^2 = \frac{\max_{j \in \tau_k} |r_{\tau_k}^j|^2}{\sum_{j \in \tau_k} |r_{\tau_k}^j|^2} \|r_{\tau_k}^i\|_2 \geq \frac{1}{|\tau_k|} \|r_{\tau_k}^i\|_2^2 \geq \frac{1}{\alpha} \|r_{\tau_k}^i\|_2^2.
\]

So, one can obtain

\[
\|A_{\tau_k}^\dagger A_{\tau_k} (x_k - x_s)\|_2^2 \geq \frac{\eta \|A_{\tau_k}\|_F^2}{\alpha \sigma^2_{\max}(A_{\tau_k}) \|A_{\tau_k}\|_F} \max_{i \in [p]} \|r_{\tau_k}^i\|_2^2.
\]
For $k = 0$,

$$\max_{i \in [p]} \| r_k^i \|_2^2 = \max_{i \in [p]} \| r_0^i \|_2^2 \cdot \frac{\| r_0 \|_2^2}{\sum_{i \in [p]} \| r_0^i \|_2^2} \geq \frac{1}{p} \| r_0 \|_2^2,$$

then,

$$\| A_{10} A_{10} (x_1 - x_*) \|_2^2 \geq \frac{\eta \| A_{10} \|_2^2 \sigma^2_{\min}(A)}{\alpha \rho \sigma^2 \max(A)} \| x_0 - x_* \|_2^2. \quad (9)$$

Assume that $\zeta \geq \max_{\tau \in \{\tau_1, \tau_2, \ldots, \tau_p\}} \| A_{10} \|_2^2$ for each $\tau \in \{\tau_1, \tau_2, \ldots, \tau_p\}$. Substituting (9) into (8), one has

$$\| x_1 - x_* \|_2^2 \leq \left( 1 - \frac{\eta \| A_{10} \|_2^2 \sigma^2_{\min}(A)}{\alpha \rho \sigma^2 \max(A)} \right) \| x_0 - x_* \|_2^2. \quad (10)$$

For $k = 1, 2, \ldots$,

$$\max_{i \in [p]} \| r_k^i \|_2^2 = \max_{i \in [p]} \| r_k^i \|_2^2 \| r_k \|_2 \sum_{i \in [p]} \| r_k^i \|_2^2 \| r_k \|_2 \geq \frac{1}{m - N_{k-1}} \| r_k \|_2^2,$$

the establishment of the second equation above is guaranteed by the (7), then

$$\| A_{10} A_{10} (x_k - x_*) \|_2^2 \geq \frac{\eta \| A_{10} \|_2^2 \sigma^2_{\min}(A)}{\alpha (m - N_{k-1}) \zeta \sigma^2 \max(A)} \| x_k - x_* \|_2^2. \quad (11)$$

Substituting (11) into (8), one has

$$\| x_{k+1} - x_* \|_2^2 \leq \left( 1 - \frac{\eta \| A_{10} \|_2^2 \sigma^2_{\min}(A)}{\alpha (m - N_{k-1}) \zeta \sigma^2 \max(A)} \right) \| x_k - x_* \|_2^2. \quad (12)$$

The recurrence relation with the inequalities (10) and (12) results in (6). \[\square\]

**Remark 2.** The convergence rate of the BKMS, which is denoted by $\rho_{BKMS} = \| x_{k+1} - x_* \|_2^2 / \| x_k - x_* \|_2^2$, is subject to $0 < \rho_{BKMS} < 1$ under the conditions of Theorem 1. In fact,

$$1 - \frac{\eta \| A_{10} \|_2^2 \sigma^2_{\min}(A)}{\alpha (m - N_{k-1}) \zeta \sigma^2 \max(A)} \leq 1 - \frac{\eta \| A_{10} \|_2^2}{\alpha (m - N_{k-1}) \sigma^2 \max(A)} \frac{\sigma^2_{\min}(A)}{\max_{i \in [p]} \| A_{10} \|_2^2} \leq 1 - \frac{\eta \sigma^2_{\min}(A)}{\alpha (m - N_{k-1}) \sigma^2 \max(A)} \frac{\| A_{10} \|_2^2}{\| A \|_F^2} \leq 1 - \frac{\eta \sigma^2_{\min}(A)}{\alpha (m - N_{k-1}) \sigma^2 \max(A)} \frac{\sigma^2_{\min}(A)}{\| A \|_F^2}.$$

The second inequality holds because $\| A_{10} \|_2^2 \geq 1$. Moreover, $0 < \eta \leq 1, 0 < N_{k-1}, \alpha \leq m$ and $0 < \sigma^2_{\min}(A) \| A \|_F^2 \leq 1$. Then it holds that

$$0 < \frac{\eta \sigma^2_{\min}(A)}{\alpha (m - N_{k-1}) \sigma^2 \max(A)} < 1.$$

Thus $0 < \rho_{BKMS} < 1$, which implies that the BKMS has a linear convergence.

**4. Numerical Examples**

In this section, the numerical performances of the RK, Motzkin, GRK, RBK, SKM, GRBK, and BKMS algorithms are compared in different settings of the least-squares problem (1). The operating environment of all experiments is the MATLAB 2020b version on a
private computer with AMD Ryzen 5 4600H and 16 GB of memory. The relative solution error is denoted by
\[ RSE := \frac{\| x_k - x^* \|_2^2}{\| x^* \|_2^2}, \]
which is used to measure the efficiency of all methods.

The initial vector is set as \( x_0 = (0, 0, \cdots, 0)^T \) for BKMS and other methods. We terminate the iteration process of each method once \( RSE \leq 10^{-6} \). The concept of a random partition sampling in [13,14] is sampling \( \tau_k \) randomly from row partition \( T = \{ \tau_1, \tau_2, \cdots, \tau_p \} \) at the \( k \)-th iteration, and for each \( i \), the subset \( \tau_i \) can be denoted as
\[ \tau_i = \left\{ \left\lfloor \left( i - 1 \right) \frac{m}{p} \right\rfloor + 1, \left\lfloor \left( i - 1 \right) \frac{m}{p} \right\rfloor + 2, \cdots, \left\lfloor \frac{m}{p} \right\rfloor \right\}, \quad i = 1, 2, \cdots, p. \]

The iterative least-squares solver CGLS in [14] is utilized to solve the underdetermined linear subsystem at each iteration since it requires fewer floating-point numbers between matrix–vector products than the MATLAB function “\texttt{pinv}”.

This section consists of three parts. In the first part (Examples 1–3), we implement MS clustering to visualize the row partition of the coefficient matrix \( A \) and verify that BKMS is more competitive than other methods to solve consistent linear systems (1) with different \( A \). In the second part (Example 4), we consider the effects of parameters \( \delta \) and \( \eta \) on the performance of BKMS. In the last part (Examples 5 and 6), we provide two practical applications of the BKMS method.

**Example 1.** This example implements the partitions of different system matrices by adopting the MS Algorithm 3. It is known that an ill-conditioned matrix means a high linear correlation between the rows of the matrix [16]. In the left graph of Figure 1a, we tested a dense matrix \( A \in \mathbb{R}^{1000 \times 500} \) with the condition number (cond\( (A) \)) being 2.60, in the middle graph of Figure 1b, we tested a sparse matrix named “Cities” with cond\( (A) \) being 207.15, and in the right graph of Figure 1c, we tested an ill-conditioned sparse matrix named “relat6” with the cond\( (A) \) being \( 4.03 \times 10^{16} \). Figure 1b plots the number of rows contained in each block of the matrices above, respectively.

As shown in Figure 1a, the rows with similar correlation coefficients \( \mu_k \) are likely to be allocated in a block. This may lead to the considerable control index set \( \tau_k \) in the BKMS method being very large, which is verified in Figure 1b. To alleviate this problem, we continue to partition the current block, so it is reasonable to introduce an almost-maximum distance rule in BKMS.

![Figure 1.](image-url)
Example 2. This example considers a linear system (1) with a tall (or fat) coefficient matrix \( A \in \mathbb{R}^{m \times n} \), i.e., \( m > n \) (or \( m < n \)). Its elements are pseudorandom numbers from a standard normal distribution generated by the MATLAB function `randn`. Table 1 reports the iteration steps (IT) and CPU time in seconds (CPU(s)) of RK, Motzkin, GRK, SKM, RBK(p), and GRBK(p) with the same number of blocks \( p = 8 \) and BKMS (\( \delta_{\text{opt}}, \eta \)). The \( \delta_{\text{opt}} \) refers to the numerical optimum parameter rather than a theoretical one. We will further discuss the \( \delta_{\text{opt}} \) in Example 4. Figure 2 shows the numerical performance of all methods, in particular, we control RBK(p), GRBK(p), and BKMS(0.5, 0.1) to have the same number of blocks \( p = 5 \). Figure 2a shows the number of rows contained in each subblock of BKMS. Figure 2b,c depict the curves of RSE versus IT and RSE versus CPU(s) of Algorithm 4, applied to solve (1) with the coefficient matrix \( A \in \mathbb{R}^{1000 \times 500} \). In addition, we default \( \eta = 0.01 \) for BKMS(\( \delta \)) in all examples unless otherwise specified.

It is observed from Table 1 that RBK(p), GRBK(p), and BKMS(\( \delta \)) are almost superior to RK, Motzkin, GRK, and SKM in terms of iteration steps and CPU time, thanks to the fact that block iteration takes multiple constraints. In addition, in block Kaczmarz methods, we find that BKMS(\( \delta \)) is more competitive than RBK(p) and GRBK(p). For example, the CPU time speed-up of the BKMS(\( \delta \)) over GRBK(p) ranges from 1.206 to 1.610. Specifically, the value 1.610 is calculated by a ratio of 2.790 (CPU time consumed by GRBK(5) with the size of \( A \) is 30,000 \( \times \) 5000) over 1.733 (CPU time used by BKMS(0.45, 0.1) for the same size of \( A \)). Furthermore, from Figure 2, BKMS(0.5) converges in less iterations and CPU time than other methods.

Example 3. This example executes the RK, Motzkin, GRK, SKM, RBK, GRBK, and BKMS methods to solve the sparse linear Equation (1). Table 2 summarizes different sparse matrices with different properties, which are taken from the Florida sparse matrix collection in [27]. The density of a matrix \( A \) means the percentage of nonzero elements of \( A \). The IT and CPU times for all methods are reported in Table 2. Similar to Example 2, Figure 3a shows the number of rows contained in each subblock of BKMS(0.5), Figure 3b,c display the plots of RSE versus IT and RSE versus the CPU times of all methods, which are applied to solve a sparse linear system (1) named “WorldCities”.
Table 1. IT and CPU of all methods for the $m$-by-$n$ consistent linear system.

| Name      | $m \times n$ | 10,000 $\times$ 5000 | 15,000 $\times$ 5000 | 20,000 $\times$ 5000 | 25,000 $\times$ 5000 | 30,000 $\times$ 5000 |
|-----------|---------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
|           | $\delta$     | 0.45                  | 0.40                  | 0.50                  | 0.55                  | 0.45                  |
| RK        | IT            | $>2 \times 10^5$       | $>2 \times 10^5$       | $>2 \times 10^5$       | $>2 \times 10^5$       | $>2 \times 10^5$       |
|           | CPU(s)        | -                     | -                     | -                     | -                     | -                     |
| Motzkin   | IT            | 54,496                | 25,199                | 17,208                | 13,473                | 11,566                |
|           | CPU(s)        | 670.145               | 432.064               | 414.186               | 448.784               | 407.672               |
| GRK       | IT            | 58,073                | 25,151                | 17,593                | 13,887                | 11,934                |
|           | CPU(s)        | 710.741               | 462.309               | 431.177               | 421.825               | 427.727               |
| SKM       | IT            | 55,246                | 25,083                | 17,134                | 13,308                | 11,587                |
|           | CPU(s)        | 665.832               | 418.554               | 408.865               | 440.814               | 406.462               |
| RBK       | IT            | 271                   | 107                   | 64                    | 49                    | 39                    |
|           | CPU(s)        | 9.491                 | 4.522                 | 3.492                 | 3.341                 | 3.234                 |
| GRBK      | IT            | 180                   | 76                    | 50                    | 39                    | 32                    |
|           | CPU(s)        | 6.527                 | 3.486                 | 2.977                 | 2.883                 | 2.790                 |
| BKMS($\delta$) | IT | 55 | 52 | 28 | 22 | 18 | 18 |
|           | CPU(s)        | 4.589                 | 2.698                 | 2.018                 | 1.886                 | 1.733                 |
| name      | $m \times n$ | 5000 $\times$ 10,000  | 5000 $\times$ 20,000  | 5000 $\times$ 30,000  | 5000 $\times$ 40,000  | 5000 $\times$ 50,000  |
| $\delta$  | 0.65          | -                     | 0.70                  | 0.80                  | 0.75                  | 0.85                  |
| RK        | IT            | $>2 \times 10^5$       | $>2 \times 10^5$       | $>2 \times 10^5$       | $>2 \times 10^5$       | $>2 \times 10^5$       |
|           | CPU(s)        | -                     | -                     | -                     | -                     | -                     |
| Motzkin   | IT            | 61,796                | 25,685                | 19,437                | 17,011                | 15,248                |
|           | CPU(s)        | 734.014               | 609.400               | 696.297               | 798.004               | 910.382               |
| GRK       | IT            | 60,858                | 26,458                | 19,697                | 17,048                | 15,393                |
|           | CPU(s)        | 726.831               | 633.951               | 773.126               | 918.833               | 954.293               |
| SKM       | IT            | 61,924                | 25,907                | 19,327                | 16,918                | 15,299                |
|           | CPU(s)        | 725.727               | 598.987               | 684.460               | 785.677               | 906.202               |
| RBK       | IT            | 357                   | 174                   | 122                   | 132                   | 90                    |
|           | CPU(s)        | 14.832                | 14.124                | 14.812                | 23.130                | 19.744                |
| GRBK      | IT            | 173                   | 66                    | 52                    | 44                    | 41                    |
|           | CPU(s)        | 5.490                 | 4.336                 | 5.062                 | 6.261                 | 7.265                 |
| BKMS($\delta$) | IT | 98 | 33 | 25 | 26 | 14 | 14 |
|           | CPU(s)        | 4.552                 | 3.550                 | 4.154                 | 4.434                 | 4.524                 |

Figure 2. (a) The number of rows contained in each partition in BKMS with $\delta = 0.5$. The RSE versus IT (b) and RSE versus CPU (c) of BKMS compared with other methods for a consistent linear system with $A \in \mathbb{R}^{1000 \times 500}$. 
Table 2. IT and CPU of all methods for the sparse consistent linear system.

| Name               | abtaha1 | relat6 | Model1 | mk11-b2 | Cites | WorldCities |
|--------------------|---------|--------|--------|---------|-------|-------------|
| $m \times n$      |         |        |        |         |       |             |
| density            | 1.68%   | 2.21%  | 1.05%  | 6.52%   | 53.04%| 23.87%      |
| $\text{cond}(A)$   | 12.23   | $4.03 \times 10^{16}$ | 17.57  | $1.05 \times 10^{16}$ | 207.15| 66.0        |
| full rank          | Yes     | No     | Yes    | No      | Yes   | Yes         |
| $p$                | 4       | 3      | 4      | 10      | 3     | 7           |
| $\delta$           | 0.5     | 0.6    | 0.4    | 0.45    | 0.8   | 0.5         |

| Method             | IT      | CPU(s) |
|--------------------|---------|--------|
| RK                 | 30,654  | 61.699 |
| Motzkin            | 704     | 1.232  |
| GRK                | 608     | 0.585  |
| SKM                | 2074    | 1.795  |
| RBK($p$)           | 167     | 0.681  |
| GRBK($p$)          | 70      | 0.248  |
| BKMS($\delta$)     | 36      | 0.071  |

From Table 2, RBK($p$), GRBK($p$), and BKMS($\delta$) require fewer iteration steps and CPU times. Similar to the block Kaczmarz methods above, we believe that the BKMS method has better performance. For example, we calculate the CPU time speed-up of the BKMS($\delta$) over GRBK($p$) ranges from 1.692 to 3.493 for solving the (1) with a different sparse matrix $A$ listed in Table 2. From Figure 3, we can see that BKMS converges faster than other methods. By carefully comparing Figure 3c, it is not difficult to find that SKM and RBK have similar time efficiencies. The reason is that calculating the Moore–Penrose pseudoinverse of large blocks reduces the efficiency of RBK. So, the number of coefficient matrix partitions affects the convergence speed of RBK, GRBK, and BKMS. Based on this, in Example 4, we will analyze the sensitivity of the parameters ($\delta$, $\eta$) in BKMS.

Example 4. This example investigates the effects of the parameters $\delta$ and $\eta$ on the BKMS method. We use BKMS to solve system (1) with $A \in \mathbb{R}^{5000 \times 1000}$ and the sparse system (1) named “abtaha1”, respectively. Tables 3 and 4 list the iteration steps and CPU time of BKMS($\delta$, $\eta$) once the RSE < $10^{-6}$. The “- -” means abandoning the introduction of the almost maximum distance rule (i.e., removing steps 5 and 6 in Algorithm 4). In fact, we usually care more about the CPU time, so we use enumeration to select optimal parameters $\delta$ and $\eta$. This tuning method is also called the grid
search in machine learning. That is, find the corresponding different parameter values by locking the optimal value. For example, we select initialization seeds $\delta = 0.7$ and $\eta = 0.1$ for BKMS($\delta, \eta$) in Table 3.

Table 3. Sensitivity analysis of BKMS($\delta, \eta$) for system (1) with $A \in \mathbb{R}^{5000 \times 1000}$.

| $\eta$ | $\delta$ | 0.1 | 0.3 | 0.5 | 0.7 | 0.9 |
|-------|----------|-----|-----|-----|-----|-----|
| IT    |          | 199 | 10  | 2   | 2   | 2   |
| CPU(s)|          | 2.196 | 1.227 | 0.323 | 0.453 | 0.469 |
| IT    | 0.1      | 255 | 43  | 9   | 2   | 2   |
| CPU(s)|          | 1.716 | 0.808 | 0.646 | 0.207 | 0.226 |
| IT    | 0.3      | 337 | 72  | 21  | 14  | 12  |
| CPU(s)|          | 1.673 | 0.741 | 0.440 | 0.555 | 0.552 |
| IT    | 0.5      | 900 | 256 | 87  | 56  | 57  |
| CPU(s)|          | 2.542 | 1.126 | 0.388 | 0.267 | 0.253 |
| IT    | 0.7      | 1742 | 636 | 246 | 158 | 144 |
| CPU(s)|          | 3.796 | 1.820 | 0.564 | 0.379 | 0.346 |
| IT    | 0.9      | 4141 | 2276 | 933 | 694 | 634 |
| CPU(s)|          | 8.221 | 4.353 | 1.540 | 1.182 | 1.075 |
| Block Size | 148 | 14 | 5 | 3 | 1 |

Table 4. Sensitivity analysis of BKMS($\delta, \eta$) for the sparse system (1) with the matrix $A$ named “abtaha1”.

| $\eta$ | $\delta$ | 0.1 | 0.3 | 0.5 | 0.7 | 0.9 |
|-------|----------|-----|-----|-----|-----|-----|
| IT    |          | 66  | 65  | 160 | 170 | 134 |
| CPU(s)|          | 0.1440 | 0.2360 | 0.8680 | 0.8730 | 1.2940 |
| IT    | 0.1      | 67  | 54  | 58  | 62  | 57  |
| CPU(s)|          | 0.0900 | 0.0850 | 0.0890 | 0.0990 | 0.0990 |
| IT    | 0.3      | 87  | 88  | 70  | 87  | 76  |
| CPU(s)|          | 0.1020 | 0.1080 | 0.0850 | 0.1160 | 0.1020 |
| IT    | 0.5      | 103 | 109 | 111 | 108 | 91  |
| CPU(s)|          | 0.1170 | 0.1220 | 0.1260 | 0.1320 | 0.1140 |
| IT    | 0.7      | 145 | 184 | 149 | 172 | 98  |
| CPU(s)|          | 0.1560 | 0.2040 | 0.1680 | 0.2020 | 0.1200 |
| IT    | 0.9      | 343 | 391 | 270 | 519 | 195 |
| CPU(s)|          | 0.3720 | 0.4140 | 0.2920 | 0.5980 | 0.2380 |
| Block Size | 26 | 9 | 5 | 2 | 1 |

As depicted in Table 3, fix $\delta = 0.3$, for the different $\eta$; although the number of iteration steps increases, the CPU time shows a trend of first falling and then rising. For each column, this law is roughly the same. Finding a suitable $\delta$ is crucial. When $\delta$ becomes larger, the number of blocks becomes smaller, which requires computing the Moore–Penrose pseudoinverse of a large block to be time-consuming. On the contrary, when $\delta$ becomes smaller, the number of blocks increases. Considering an extreme case, when $A$ is row-normalized and there is only one row index in each block, BKMS is mathematically equivalent to the Motzkin method. A similar result also occurs in Table 4. The initialization seeds $\delta = 0.3$ and $\eta = 0.1$ can be selected for BKMS($\delta, \eta$) in Table 4.

Example 5 (Discrete ill-posed problem). This example uses the BKMS method to the solution of Phillips ill-posed problem in [28] and compares it with RK, Motzkin, GRK, SKM, RBK, and GRBK
methods. The linear systems (1) are obtained by the discretization of the Fredholm integral equation of the first kind,

\[ \int_{-6}^{6} K(s, t) \phi(t) \, dt = f(s) \]

on the square \([-6, 6] \times [-6, 6]\), where the kernel function is presented by \(K(s, t) = \phi(s - t)\) with

\[ \phi(x) = \begin{cases} 1 + \cos\left(\frac{\pi x}{3}\right), & |x| < 3, \\ 0, & |x| \geq 3, \end{cases} \]

and the right-hand side

\[ f(s) = (6 - |s|)(1 + \frac{1}{2} \cos\left(\frac{5\pi s}{3}\right)) + \frac{9}{2\pi} \sin\left(\frac{|s|\pi}{3}\right), \]

which is often severely ill-conditioned. In fact, ordinary linear systems (2) may yield poor approximate solutions in this case, then one of the effective ways to overcome this issue is ridge regression, also called the Tikhonov regularization, namely

\[ \min_{x \in \mathbb{R}^n} \{ \|Ax - b\|_2^2 + \lambda \|x\|_2^2 \}, \]

which can be restated as

\[ \min_{x \in \mathbb{R}^n} \{ \|Ax - b\|_2^2 + \lambda \|x\|_2^2 \} = \min_{x \in \mathbb{R}^n} \left\| \begin{bmatrix} A \\ \sqrt{\lambda} I_n \end{bmatrix} x - \begin{bmatrix} b \\ 0 \end{bmatrix} \right\|_2^2 = \min_{x \in \mathbb{R}^n} \|\tilde{A}x - \tilde{b}\|_2^2, \]

where \(A \in \mathbb{R}^{(n+n) \times n}\) and \(\lambda > 0\) is a regularization parameter. The measurement matrix \(A \in \mathbb{R}^{n \times n}\), exact solution \(x^* \in \mathbb{R}^n\), and \(b \in \mathbb{R}^n\) are all calculated by the MATLAB function phillips(n) in [28].

Set \(n = 1024\) and \(\lambda = 0.01\). Then the size of the square matrix \(A\) is 1024 × 1024 and the condition number of \(A\) is \(2.9043 \times 10^{10}\) while \(\tilde{A}\) is 58.04. In Figure 4, the (a) reports the approximation solution calculated by RK, Motzkin, GRK, SKM, RBK(4), GRBK(4), and BKMS(0.4, 0.8) together with the exact solution for the phillips test problem when the maximum number of iteration steps is 100. The (b) shows the partition performance of BKMS(0.4, 0.8). Table 5 lists the RSE, CPU time, and iteration of BKMS(0.4, 0.8) compared with RK, Motzkin, GRK, SKM, RBK(4), and GRBK(4) for the “phillips” problem when \(RSE \leq 10^{-10}\).
Table 5. Numerical results for the “phillips” test problem.

| Method        | RSE     | CPU Time  | Iterations |
|---------------|---------|-----------|------------|
| RK            | $5.360 \times 10^{-3}$ | $1.220 \times 10^{-1}$ | 100         |
| Motzkin       | $1.934 \times 10^{-7}$ | $6.385 \times 10^{-2}$ | 100         |
| GRK           | $3.382 \times 10^{-5}$ | $6.473 \times 10^{-2}$ | 100         |
| SKM           | $1.826 \times 10^{-7}$ | $5.904 \times 10^{-2}$ | 100         |
| RBK(4)        | $3.880 \times 10^{-13}$ | $7.183 \times 10^{-2}$ | 2           |
| GRBK(4)       | $3.821 \times 10^{-13}$ | $6.999 \times 10^{-2}$ | 2           |
| BKMS(0.4, 0.8)| $3.447 \times 10^{-15}$ | $5.320 \times 10^{-3}$ | 2           |

Figure 4 shows that the recovered solutions by RBK(4), GRBK(4), and BKMS(0.4, 0.8) are much closer to the exact solution than RK, Motzkin, GRK, and SKM do. It can be seen from Table 5 that BKMS obtains less RSE than other methods. In addition, BKMS needs less CPU time and iterations to reach the stop rule.

Example 6 (CT reconstruction problem). This example implements the reconstruction of a computer tomography (CT) image. This test problem can be implemented by the MATLAB function `paralleltomo(N, θ, p)` in the ART Tools package in [29]. We set $N = 50$, $θ = 0° : 1° : 178°$ and $p = 60$, then resulting in the size of $A$ is $10,740 \times 2500$, the exact solution $x^*$ (See the (a) of Figure 5) and the $b$ we obtained by $b = Ax^*$. The BKMS(0.25, 0.3) is applied to reconstruct $x^*$ from $b$ and compared with the RK, SKM, RBK(7), and GRBK(7) methods.

![Image](image_url)

Figure 5 shows the recovered images by RK, SKM, RBK(7), GRBK(7), and BKMS(0.25, 0.3) together with the original image. Table 6 reports the RSE, the CPU time, and iteration steps of BKMS(0.25, 0.3) compared with RK, SKM, RBK(7), and GRBK(7) once RSE $\leq 10^{-6}$. We set the maximum iterative number (IT$_{max}$) to 8000.

It is shown from Figure 5 that all methods obtained well-restored images above. It can be seen from Table 6 that BKMS(0.25, 0.3) needs less CPU time and iteration steps than other methods for restoring images.
Table 6. Numerical results for restoring the “phantom” image.

| Method   | RSE        | CPU Time | Iterations |
|----------|------------|----------|------------|
| RK       | $6.994 \times 10^{-2}$ | 104.653 | 8000       |
| Motzkin  | $2.681 \times 10^{-3}$ | 87.496  | 8000       |
| SKM      | $2.645 \times 10^{-3}$ | 82.193  | 8000       |
| RBK(7)   | $9.935 \times 10^{-7}$ | 136.756 | 221        |
| GRBK(7)  | $9.818 \times 10^{-7}$ | 72.386  | 118        |
| BKMS(0.25, 0.3) | $9.740 \times 10^{-7}$ | 32.361  | 104        |

5. Conclusions

This work proposes a block Kaczmarz–Motzkin method with mean shift clustering as an efficient extension of the Kaczmarz algorithm to solve large consistent linear equations. The row correlation coefficient of the system matrix is utilized as the clustering criterion and this method does not need to specify the number of partitions in advance. The convergence theory of the BKMS method is established when the linear system is consistent. Several numerical examples demonstrate the effectiveness of the current method in this paper.

Although the effects of the clustering bandwidth $\delta$ and the parameter $\eta$ are investigated in the numerical example, the theoretical analysis of $\delta$ affecting the performance of BKMS needs further discussion. According to the properties of the linear system, choosing a more appropriate clustering criterion may improve the algorithm performance; we will use these problems as part of our future work.

Author Contributions: Y.L.: software, validation, formal analysis, investigation, writing—original draft. T.L.: conceptualization, validation, formal analysis, writing—review and editing, supervision, funding acquisition. F.Y.: validation, formal analysis, writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Project of the Department of Science and Technology of Sichuan Province (No. 2021ZYD0005), the Opening Project of the Key Laboratory of Higher Education of Sichuan Province (No. 2020WZJ01), the Scientific Project of SUSE (No. 2020RC24), and the Postgraduate Research Fund (No. y2021101).

Institutional Review Board Statement: Not applicable

Informed Consent Statement: Not applicable

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

Conflicts of Interest: The authors have no financial or proprietary interests in any material discussed in this article.

References
1. Kaczmarz, S. Angenäherte auflösung von systemen linearer Gleichungen. Bull. Int. Acad. Polon. Sci. Lett. 1937, 35, 355–357.
2. Byrne, C. A unified treatment of some iterative algorithms in signal processing and image reconstruction. Inverse Probl. 2004, 20, 103–120. [CrossRef]
3. Censor, Y. Parallel application of block-iterative methods in medical imaging and radiation therapy. Math. Program. 1988, 42, 307–325. [CrossRef]
4. Marijan, M.; Ignjatovic, Z. Nonlinear reconstruction of delta-sigma modulated signals: Randomized surrogate constraint decoding algorithm. IEEE Trans. Signal Process. 2013, 61, 5361–5373. [CrossRef]
5. Elble, J.M.; Sahinidis, N.V.; Vouzis, P. GPU computing with Kaczmarz and other iterative algorithms for linear systems. Parallel Comput. 2010, 36, 215–231. [CrossRef]
6. Pasqualetti, F.; Carli, R.; Bullo, F. Distributed estimation via iterative projections with application to power network monitoring. Automatica 2012, 48, 747–758. [CrossRef]
7. Strohmer, T.; Vershynin, R. A randomized Kaczmarz algorithm with exponential convergence. Fourier Anal. Appl. 2009, 15, 262–278. [CrossRef]
8. Bai, Z.Z.; Wu, W.T. On greedy randomized Kaczmarz method for solving large sparse linear systems. SIAM J. Sci. Comput. 2018, 40, A592–A606. [CrossRef]
9. De Loera, J.A.; Haddock, J.; Needell, D. A sampling Kaczmarz–Motzkin algorithm for linear feasibility. SIAM J. Sci. Comput. 2017, 39, S66–S87. [CrossRef]
10. Motzkin, T.S.; Schoenberg, I.J. The relaxation method for linear inequalities. Canad. J. Math. 1954, 6, 393–404. [CrossRef]
11. Elfving, T. Block-iterative methods for consistent and inconsistent linear equations. Numer. Math. 1980, 35, 1–12. [CrossRef]
12. Eggermont, P.; Herman, G.; Lent, A. Iterative algorithms for large partitioned linear systems with applications to image reconstruction. Linear Algebra Appl. 1981, 40, 37–67. [CrossRef]
13. Needell, D.; Tropp, J.A. Paved with good intentions: Analysis of a randomized block Kaczmarz method. Linear Algebra Appl. 2014, 441, 199–221. [CrossRef]
14. Liu, Y.; Gu, C.Q. On greedy randomized block Kaczmarz method for consistent linear systems. Linear Algebra Appl. 2021, 616, 178–200. [CrossRef]
15. Chen, J.Q.; Huang, Z.D. On a fast deterministic block Kaczmarz method for solving large-scale linear systems. Numer. Algorithms 2022, 89, 1007–1029. [CrossRef]
16. Jiang, X.L.; Zhang, K.; Yin, J.F. Randomized block Kaczmarz methods with k-means clustering for solving large linear systems. Comput. Appl. Math. 2022, 413, 113828. [CrossRef]
17. Ma, A.; Needell, D.; Ramdas, A. Convergence properties of the randomized extended Gauss-Seidel and Kaczmarz methods. SIAM J. Matrix Anal. Appl. 2015, 36, 1590–1604. [CrossRef]
18. Necoara, I. Faster randomized block Kaczmarz algorithms. SIAM J. Matrix Anal. Appl. 2019, 40, 1425–1452. [CrossRef]
19. Needell, D.; Zhao, R.; Zouzias, A. Randomized block Kaczmarz method with projection for solving least squares. Linear Algebra Appl. 2015, 484, 322–343. [CrossRef]
20. Wen, L.; Yin, F.; Liao, Y.M.; Huang, G.X. A greedy average block Kaczmarz method for the large scale consistent system of linear equations. AIMS Math. 2022, 7, 6792–6806. [CrossRef]
21. Zhang, Y.J.; Li, H.Y. Block sampling Kaczmarz-Motzkin methods for consistent linear systems. Calcolo 2021, 58, 39. [CrossRef]
22. Needell, D.; Ward, R. Two-Subspace Projection Method for Coherent Overdetermined Systems. Fourier Anal. Appl. 2013, 19, 256–269. [CrossRef]
23. Izenman, A.J. Density Estimation for Statistics and Data Analysis. J. Am. Stat. Assoc. 1988, 83, 269. [CrossRef]
24. Jain, A.K. Data clustering: 50 years beyond k-means. Pattern Recogn. Lett. 2010, 31, 651–666. [CrossRef]
25. Fukunaga, K.; Hostetler, L. The estimation of the gradient of a density function with applications in pattern recognition. IEEE Trans. Inform. Theory 1975, 21, 32–40. [CrossRef]
26. Ghassabeh, Y.A.; Rudzicz, F. Modified mean shift algorithm. IET Image Process. 2018, 12, 2172–2177. [CrossRef]
27. Davis, T.A.; Hu, Y. The university of florida sparse matrix collection. ACM Trans. Math. Softw. 2011, 38, 1–25. [CrossRef]
28. Hansen, P.C. Regularization Tools: A Matlab package for analysis and solution of discrete ill-posed problems. Numer. Algorithms 1994, 6, 1–35. [CrossRef]
29. Gordon, R.; Bender, R.; Herman, G. Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography. J. Theor. Biol. 1970, 29, 471–481. [CrossRef]