Journal of Computational Mathematics
Vol.xx, No.x, 2025, 1–24.
http://www.global-sci.org/jcm
doi:10.4208/jcm.2506-m2025-0047

ON THE ADAPTIVE DETERMINISTIC BLOCK COORDINATE DESCENT METHOD WITH MOMENTUM FOR SOLVING LARGE LINEAR LEAST-SQUARES PROBLEMS[*]

Longze Tan
School of Mathematical Sciences, University of Science and Technology of China, Hefei 230026, China
Email: ba23001003@mail.ustc.edu.cn
Mingyu Deng
School of Statistics and Applied Mathematics, Anhui University of Finance & Economics, Bengbu 233030, China
Email: my_deng@stu.ecnu.edu.cn
Jiali Qiu
School of Mathematics and Statistics, Xi'an Jiaotong University, Shaanxi 710049, China
Email: qsq2110748243@stu.xjtu.edu.cn
Xueping Guo[1)]
School of Mathematical Sciences, Key Laboratory of MEA (Ministry of Education) & Shanghai Key Laboratory of PMMP, East China Normal University, Shanghai 200241, China
Email: xpguo@math.ecnu.edu.cn

Abstract

Inspired by Polyak's heavy-ball method, this paper proposes an adaptive deterministic block coordinate descent method with momentum (mADBCD) for efficiently solving large-scale linear least-squares problems. The proposed method introduces a novel column selection criterion based on the Euclidean norm of the residual vector of the normal equation. In contrast to classical block coordinate descent methods, mADBCD does not require a fixed pre-partitioning of the column indices of the coefficient matrix and avoids the expensive computation of Moore-Penrose pseudoinverses of submatrices at each iteration. The method adaptively updates the block index set at each step, thereby improving both flexibility and scalability. When the coefficient matrix is of full column rank, we prove that mADBCD converges linearly to the unique solution of the least-squares problem. Numerical experiments are conducted to show that mADBCD outperforms several recent block coordinate descent methods in terms of iteration count and CPU time. In particular, when solving extremely sparse least-squares problems, mADBCD is the first block coordinate descent method reported to achieve CPU time nearly comparable to that of the classical least squares QR (LSQR) method [Paige and Saunders, ACM Trans. Math. Softw., 8 (1982)].

Mathematics subject classification: 65F10, 65F20, 65K05, 90C25, 15A06.
Key words: Linear least-squares problem, Coordinate descent method, Block coordinate descent method, Heavy ball method, Convergence.

[*] Received February 7, 2024 / Revised version received May 23, 2025 / Accepted June 23, 2025 / Published online December 10, 2025 /

[1)] Corresponding author

1. Introduction

The least-squares problem has long played a fundamental role in scientific computing and optimization; see [1, 7, 12, 25, 27] and the references therein. In this work, we focus on the following large-scale linear least-squares problem:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \|\mathbf{b}-\mathbf{A}\mathbf{x}\|_2, \tag{1.1}$$

where the coefficient matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ $(m \geq n)$ has full column rank, $\mathbf{b} \in \mathbb{R}^m$ is a given vector, and $\|\cdot\|_2$ denotes the Euclidean norm. Under the full-rank assumption, problem (1.1) admits a unique solution $\mathbf{x}_\star$, which can be explicitly written as $\mathbf{x}_\star = \mathbf{A}^\dagger\mathbf{b}$, where

$$\mathbf{A}^\dagger = (\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top$$

is the Moore-Penrose pseudoinverse of $\mathbf{A}$. This solution is equivalently obtained by solving the normal equations

$$\mathbf{A}^\top\mathbf{A}\mathbf{x} = \mathbf{A}^\top\mathbf{b} \tag{1.2}$$

as they are mathematically equivalent.

1.1. The coordinate descent methods

The problem (1.1) can be solved by various direct methods, such as QR factorization (where Q is an orthogonal matrix and R is an upper triangular matrix) and singular value decomposition (SVD) [3, 11]. However, when the problem dimensions become extremely large, these methods require substantial memory and incur significant computational cost, resulting in impractical runtimes [3]. As a result, a wide range of iterative methods have been developed to address large-scale least-squares problems.

Among them, the coordinate descent (CD) method is one of the most economical and effective approaches. It can be viewed as a direct application of the classical Gauss-Seidel method to the normal equations (1.2) [24]. Starting from an initial guess $\mathbf{x}^{(0)}$, the CD iteration takes the form

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \frac{\mathbf{A}_{(j_k)}^\top(\mathbf{b}-\mathbf{A}\mathbf{x}^{(k)})}{\|\mathbf{A}_{(j_k)}\|_2^2}\mathbf{e}_{j_k}, \quad k = 0, 1, 2, \ldots,$$

where $\mathbf{A}_{(j_k)}$ is the j_k-th column of $\mathbf{A}$ and $\mathbf{e}_{j_k}$ is the j_k-th standard basis vector in $\mathbb{R}^n$.

Leventhal and Lewis [13] introduced the randomized coordinate descent (RCD) method, which achieves an expected linear convergence rate. This method is also referred to as the randomized Gauss-Seidel (RGS) method in the literature on randomized algorithms for solving large-scale linear systems [6, 10, 17]. Its simplicity has inspired extensive developments in both theory and applications; see [9, 15, 18, 21] and references therein.

In 2019, Bai and Wu [2] proposed the greedy randomized coordinate descent (GRCD) method by designing a probability-based criterion for selecting working columns. Later, Zhang and Guo [29] introduced a relaxation parameter $\theta \in [0, 1]$ and developed the relaxed greedy randomized coordinate descent (RGRCD) method, along with a convergence analysis. Zhang and Li [30] further improved GRCD by proposing the greedy Gauss-Seidel (GGS) method, which selects columns based on the maximum entries of the residual vector associated with the normal equations. Most recently, Tan and Guo [26] developed a multi-step version, the MGRCD method, which achieves a faster convergence rate than GRCD.

1.2. The block coordinate descent methods

Let $[z]$ denote the set $\{1, 2, \ldots, z\}$ for any positive integer z. Consider a partition $\{\tau_1, \tau_2, \ldots, \tau_p\}$ of the column index set $[n]$ such that for all $i \neq j$, we have $\tau_i \cap \tau_j = \emptyset$, $\tau_i \neq \emptyset$, and $\bigcup_{i=1}^{p} \tau_i = [n]$. Extending the idea of the randomized block Kaczmarz method [19, 20], Wu [28] proposed the randomized block Gauss-Seidel (RBGS) method, which applies a randomized strategy to select a column submatrix of the coefficient matrix $\mathbf{A}$ at each iteration. The RBGS iteration is given by

$$\begin{aligned}\mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \mathbf{I}_{\tau_{j_k}} \left(\mathbf{A}_{\tau_{j_k}}^{\top} \mathbf{A}_{\tau_{j_k}}\right)^{-1} \mathbf{A}_{\tau_{j_k}}^{\top} (\mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}) \\ &= \mathbf{x}^{(k)} + \mathbf{I}_{\tau_{j_k}} \mathbf{A}_{\tau_{j_k}}^{\dagger} (\mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}), \quad k = 0, 1, 2, \ldots, \end{aligned} \tag{1.3}$$

where the subscript $(\cdot)_{\tau_{j_k}}$ refers to the column submatrix corresponding to the index set τ_{j_k} in the given partition $\{\tau_1, \tau_2, \ldots, \tau_p\}$.

Li and Zhang [14] proposed the greedy block Gauss-Seidel (GBGS) method by adopting a column selection strategy from the RGRCD method. Subsequently, Liu *et al.* [16] developed the maximal residual block Gauss-Seidel (MRBGS) method and established its convergence properties. In 2022, Chen and Huang [4] introduced the fast block coordinate descent (FBCD) method, which employs a novel update scheme distinct from that of RBGS and MRBGS. Their numerical experiments demonstrate that the FBCD method outperforms both GBGS and MRBGS in solving problem (1.1).

Inspired by Polyak's heavy-ball method [23], this paper proposes an adaptive deterministic block coordinate descent method with momentum (mADBCD) for efficiently solving large-scale linear least-squares problems. The method employs a novel column selection criterion based on the Euclidean norm of the residual of the normal equations. Unlike classical block coordinate descent methods, mADBCD avoids fixed pre-partitioning of column indices and the costly computation of Moore-Penrose pseudoinverses at each iteration. It adaptively updates the block index set at each step, enhancing both flexibility and scalability. Assuming the coefficient matrix has full column rank, we establish that mADBCD converges linearly to the unique least-squares solution $\mathbf{x}_\star$ of problem (1.1). Numerical experiments demonstrate that mADBCD outperforms GBGS, MRBGS, and FBCD in terms of both iteration count and CPU time. In particular, when applied to extremely sparse problems, mADBCD is the first block coordinate descent method reported to achieve CPU performance nearly comparable to that of the classical LSQR method [22].

The rest of the paper is organized as follows. Section 2 introduces the notations, reviews the FBCD and heavy-ball methods, and presents two auxiliary lemmas. The proposed mADBCD method, together with its convergence analysis and error estimation, is detailed in Section 3. Section 4 presents numerical experiments that demonstrate the effectiveness of mADBCD and its advantages over existing block coordinate methods. Finally, Section 5 concludes the paper.

2. Necessary Notation and Preliminary Results

Notation. For any vector $\mathbf{a} \in \mathbb{R}^n$, we denote by $\mathbf{a}_i$ its i-th entry, and by $\|\mathbf{x}\|_2$ the Euclidean norm of a vector $\mathbf{x} \in \mathbb{R}^n$. The symbol $\mathbf{e}_j$ refers to the j-th standard basis vector in $\mathbb{R}^n$, which has a 1 in the j-th position and 0 in all other entries. The Euclidean inner product between two n-dimensional column vectors $\mathbf{x}$ and $\mathbf{y}$ is denoted by $\langle \mathbf{x}, \mathbf{y} \rangle$. Given a matrix $\mathbf{G} \in \mathbb{R}^{m \times n}$, we use $\mathbf{G}_{(j)}$ to denote its j-th column, $\mathbf{G}^{\top}$ its transpose, $\mathbf{G}^{\dagger}$ its Moore-Penrose pseudoinverse,

$\mathcal{R}(\mathbf{G})$ its column space, and $\|\mathbf{G}\|_F$ its Frobenius norm. The singular value decomposition of $\mathbf{G}$ is given by

$$\mathbf{G} = \mathbf{U}\begin{bmatrix}\mathbf{\Sigma}\\ \mathbf{0}\end{bmatrix}\mathbf{V}^\top,$$

where $\mathbf{U} \in \mathbb{R}^{m\times m}$ and $\mathbf{V} \in \mathbb{R}^{n\times n}$ are orthogonal matrices, and

$$\mathbf{\Sigma} = \operatorname{diag}\left(\sigma_1(\mathbf{G}), \sigma_2(\mathbf{G}), \dots, \sigma_n(\mathbf{G})\right) \in \mathbb{R}^{n\times n}$$

is a diagonal matrix of singular values. The nonzero singular values satisfy

$$\sigma_{\max}(\mathbf{G}) := \sigma_1(\mathbf{G}) \geq \sigma_2(\mathbf{G}) \geq \cdots \geq \sigma_r(\mathbf{G}) := \sigma_{\min}(\mathbf{G}) > 0,$$

where $r = \operatorname{rank}(\mathbf{G})$, and $\sigma_{\max}(\mathbf{G})$ and $\sigma_{\min}(\mathbf{G})$ denote the largest and smallest nonzero singular values, respectively. For any vector $\mathbf{x} \in \mathbb{R}^n$, we define the energy norm induced by a symmetric positive-definite matrix $\mathbf{G} \in \mathbb{R}^{n\times n}$ as $\|\mathbf{x}\|_{\mathbf{G}} := \sqrt{\mathbf{x}^\top\mathbf{G}\mathbf{x}}$. Let the residual vector $\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}$ and $\mathbf{s}^{(k)} = \mathbf{A}^\top\mathbf{r}^{(k)}$. For an index set $\tau \subseteq [n]$, the column submatrix formed by selecting the columns of $\mathbf{A}$ indexed by τ is denoted by $\mathbf{A}\tau$, and the cardinality of τ is denoted by $|\tau|$.

The FBCD method. Chen and Huang [4] proposed the FBCD method for solving problem (1.1), based on the greedy column selection criterion introduced in [2] and the following iterative update:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \frac{\eta_k^\top \mathbf{s}^{(k)}}{\|\mathbf{A}\eta_k\|_2^2}\eta_k, \tag{2.1}$$

where

$$\eta_k = \sum_{j_k \in \mathcal{V}_k} (\mathbf{s}^{(k)})_{j_k}\mathbf{e}_{j_k},$$

see [4] for the definition of $\mathcal{V}_k$.

The heavy ball method. Consider the unconstrained optimization problem $\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$, where $f(\mathbf{x})$ is a differentiable convex function. A widely used approach is the gradient descent (GD) method, which starts from an initial point $\mathbf{x}^{(0)}$ and updates iteratively via

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \nabla f(\mathbf{x}^{(k)}), \quad k = 0, 1, \dots,$$

where $\alpha_k > 0$ is the step size and $\nabla f(\mathbf{x}^{(k)})$ denotes the gradient of f at $\mathbf{x}^{(k)}$.

Polyak [23] enhanced the GD method by adding a momentum term $\beta(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)})$, known as the heavy ball term. This leads to the momentum-based gradient descent method (mGD), commonly referred to as the heavy ball method

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \nabla f(\mathbf{x}^{(k)}) + \beta(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}), \quad k = 0, 1, \dots. \tag{2.2}$$

Finally, we conclude this section with the following two lemmas, which are crucial for the convergence analysis of the mADBCD method.

Lemma 2.1. *If $\mathbf{G} \in \mathbb{R}^{m\times n}$ has full column rank, we have*

$$\sigma_{\min}^2(\mathbf{G})\|\mathbf{x}\|_2^2 \leq \|\mathbf{G}\mathbf{x}\|_2^2 \leq \sigma_{\max}^2(\mathbf{G})\|\mathbf{x}\|_2^2, \quad \forall \mathbf{x} \in \mathbb{R}^n.$$

Lemma 2.2 ([8]). *Fix $F^{(1)} = F^{(0)} \geq 0$, and let $\{F^{(k)}\}_{k\geq 0}$ be a sequence of nonnegative real numbers satisfying the relation*

$$F^{(k+1)} \leq \gamma_1 F^{(k)} + \gamma_2 F^{(k-1)}, \quad \forall\, k \geq 1,$$

where $\gamma_2 \geq 0, \gamma_1 + \gamma_2 < 1$. Then the sequence satisfies the relation

$$F^{(k+1)} \leq q^k (1+\tau) F^0, \quad \forall\, k \geq 0,$$

where

$$q = \begin{cases} \dfrac{\gamma_1 + \sqrt{\gamma_1^2 + 4\gamma_2}}{2}, & \text{if} \quad \gamma_2 > 0, \\ \gamma_1, & \text{if} \quad \gamma_2 = 0, \end{cases}$$

and $\tau = q - \gamma_1 \geq 0$. Moreover, $\gamma_1 + \gamma_2 \leq q < 1$, with equality if and only if $\gamma_2 = 0$.

3. The mADBCD Method and Its Convergence

In this section, we first present the mADBCD method. Subsequently, our focus will be on establishing its convergence analysis. To construct the mADBCD method, we need to introduce the following novel block control index set:

$$\tau_k = \left\{ j_k \,\middle|\, \left|(\mathbf{s}^{(k)})_{j_k}\right|^2 \geq \frac{\|\mathbf{s}^{(k)}\|_2^2}{n} \right\}.$$

It is easy to know that the set τ_k is a non-empty set. To expedite the convergence speed of the iterative scheme (2.1), we apply the idea of the heavy ball method (2.2) to obtain the following mADBCD method.

Algorithm 3.1: The mADBCD Method.

```
  Require: $\mathbf{A}, \mathbf{b}, \ell, \beta \geq 0, \mathbf{x}^{(0)} = \mathbf{x}^{(1)} \in \mathbb{R}^n$ and $\mathbf{s}^{(1)} = \mathbf{A}^\top(\mathbf{b} - \mathbf{A}\mathbf{x}^{(1)})$.
  Ensure : Approximate solution $\mathbf{x}^{(\ell)}$.
1 for $k = 1, 2, \ldots, \ell - 1$ do
2     Determine the control index set
          $\tau_k = \left\{ j_k \,\middle|\, \left|(\mathbf{s}^{(k)})_{j_k}\right|^2 \geq \frac{\|\mathbf{s}^{(k)}\|_2^2}{n} \right\}.$   (3.1)
3     Compute
          $\eta_k = \sum_{j_k \in \tau_k} (\mathbf{s}^{(k)})_{j_k} \mathbf{e}_{j_k}.$   (3.2)
4     Set
          $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \frac{\eta_k^\top \mathbf{s}^{(k)}}{\|\mathbf{A}\eta_k\|_2^2} \eta_k + \beta(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}).$   (3.3)
5 end
```

Remark 3.1. The proposed mADBCD method is primarily designed for overdetermined linear least squares problems, i.e. the number of rows m is larger than the number of columns n. In this setting, the index set τ_k defined in Eq. (3.1) typically selects only a small subset of coordinates, making the algorithm computationally efficient. Nevertheless, we observe that the method also performs well in more general scenarios. As shown in Table 4.8, the mADBCD method exhibits good convergence behavior even when applied to square systems (e.g. cage8 and cage10, where $m = n$). Furthermore, Table 4.4 demonstrates that mADBCD remains efficient on dense least squares problems where the number of columns exceeds half the number of rows.

As follows is the important property for the mADBCD method.

Proposition 3.1. *From any initial vector* $\mathbf{x}^{(0)}$*, the iterative sequence* $\{\mathbf{x}^{(k)}\}_{k=0}^{\infty}$ *generated by the mADBCD method is well-defined.*

Proof. For any $\mathbf{x}^{(0)} = \mathbf{x}^{(1)} \in \mathbb{R}^n$, $\|\mathbf{A}^\top(\mathbf{b} - \mathbf{A}\mathbf{x}^{(1)})\|_2$ either equals zero or not. If

$$\|\mathbf{A}^\top(\mathbf{b} - \mathbf{A}\mathbf{x}^{(1)})\|_2 = 0,$$

the sequence contains only $\mathbf{x}^{(0)}$ and $\mathbf{x}^{(1)}$. When $\|\mathbf{A}^\top(\mathbf{b} - \mathbf{A}\mathbf{x}^{(1)})\|_2 \neq 0$, it is easy to show that the set τ_1 defined by (3.1) is non-empty. To demonstrate the existence of $\mathbf{x}^{(2)}$ as defined in (3.3), we just need to show that $\|\mathbf{A}\eta_1\|_2 \neq 0$. Assuming $\|\mathbf{A}\eta_1\|_2 = 0$, then $\mathbf{A}\eta_1 = \mathbf{0}$ and

$$\eta_1 \mathbf{s}^{(1)} = \eta_1^\top \mathbf{A}^\top(\mathbf{b} - \mathbf{A}\mathbf{x}^{(1)}) = (\mathbf{A}\eta_1)^\top(\mathbf{b} - \mathbf{A}\mathbf{x}^{(1)}) = 0. \tag{3.4}$$

However, according to (3.1) and (3.2), it follows that

$$\eta_1 \mathbf{s}^{(1)} = \sum_{j_1 \in \tau_1} \left|(\mathbf{s}^{(1)})_{j_1}\right|^2 \geq \frac{1}{n}|\tau_1| \, \|\mathbf{s}^{(1)}\|_2^2 > 0, \tag{3.5}$$

which generates a contradiction to (3.4), thus this shows the existence of $\mathbf{x}^{(2)}$.

Suppose $\mathbf{x}^{(k)}$ $(k \geq 2)$ has been computed by the mADBCD method, then we can repeat the above derivation process by using η_k and $\mathbf{x}^{(k)}$ to replace η_1 and $\mathbf{x}^{(1)}$, respectively. Thus, when

$$\|\mathbf{s}^{(k)}\|_2 = \|\mathbf{A}^\top(\mathbf{b} - \mathbf{A}\mathbf{x}^{(k)})\|_2 = 0,$$

i.e. iterations stop and the sequence generated by the mADBCD method is $\{\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(k)}\}$. On the other hand, if $\|\mathbf{A}^\top(\mathbf{b} - \mathbf{A}\mathbf{x}^{(k)})\|_2 \neq 0$, (3.4) and (3.5) still remain true after $\eta_1, \mathbf{x}^{(1)}$ and τ_1 are replaced by $\eta_k, \mathbf{x}^{(k)}$ and τ_k, respectively, which means that the same contradictions will still occur. This can introduce $\|\mathbf{A}\eta_k\|_2 \neq 0$, and thus we can show the existence of $\mathbf{x}^{(k+1)}$ in the same way as above. It follows by induction that the iterative sequence $\{\mathbf{x}\}_{k\geq 0}$ generated by the mADBCD method exists. □

The error estimate of the mADBCD method is characterized in the following theorem.

Theorem 3.1. *Consider the large linear least-squares problem* $\min_{\mathbf{x}\in\mathbb{R}^n} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2$*, where* $\mathbf{A} \in \mathbb{R}^{m\times n}$ $(m \geq n)$ *is of full column rank and* $\mathbf{b} \in \mathbb{R}^m$ *is a given vector. Suppose that the following expressions:*

$$\gamma_1 = 1 + 3\beta + 2\beta^2 - (3\beta + 1)\frac{|\tau_k|\sigma_{\min}^2(\mathbf{A})}{n\sigma_{\max}^2(\mathbf{A}_{\tau_k})}, \quad \gamma_2 = 2\beta^2 + \beta$$

satisfy $\gamma_1 + \gamma_2 < 1$*. Then the iteration sequence* $\{\mathbf{x}^{(k)}\}_{k=0}^{\infty}$*, generated by the mADBCD method starting from any initial guess* $\mathbf{x}^{(0)}$*, exists and converges to the unique least-squares solution* $\mathbf{x}_\star = \mathbf{A}^\dagger\mathbf{b}$*, with the error estimate*

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}_\star\|_{\mathbf{A}^\top\mathbf{A}}^2 \leq q^k(1 + \tau)\|\mathbf{x}^{(0)} - \mathbf{x}_\star\|_{\mathbf{A}^\top\mathbf{A}}^2, \tag{3.6}$$

where $q = (\gamma_1 + \sqrt{\gamma_1^2 + 4\gamma_2})/2$ *and* $\tau = q - \gamma_1$*. Moreover,* $\gamma_1 + \gamma_2 \leq q < 1$*.*

Proof. Based on the property (Proposition 3.1), it can be deduced that the iterative sequence $\{\mathbf{x}^{(k)}\}_{k\geq 0}$ indeed exists. For certain values of $0 \leq k \leq \infty$, when $\|\mathbf{A}^\top(\mathbf{b}-\mathbf{A}\mathbf{x}^{(k)})\|_2 = 0$, $\{\mathbf{x}^{(k)}\}_{k\geq 0}$ is simply a sequence containing only a finite number of elements, in which case the iteration terminates with $\mathbf{x}^{(k)} = \mathbf{x}_\star$. If, for any $k \geq 0$, it holds that $\|\mathbf{A}^\top(\mathbf{b}-\mathbf{A}\mathbf{x}^{(k)})\|_2 \neq 0$, then we will prove that the sequence $\{\mathbf{x}^{(k)}\}_{k=0}^{\infty}$ converges to $\mathbf{x}_\star$.

For $k \geq 0$, set

$$\mathbf{P}_k = \frac{\mathbf{A}\eta_k\eta_k^\top\mathbf{A}^\top}{\|\mathbf{A}\eta_k\|_2^2},$$

it is easy to prove that $\mathbf{P}_k$ satisfies $\mathbf{P}_k = \mathbf{P}_k^\top$ and $\mathbf{P}_k^2 = \mathbf{P}_k$. According to the definition of $\mathbf{P}_k$ and $\mathbf{A}^\top\mathbf{A}\mathbf{x}_\star = \mathbf{A}^\top\mathbf{b}$, it holds that

$$\begin{aligned}\frac{\eta_k^\top\mathbf{s}^{(k)}}{\|\mathbf{A}\eta_k\|_2^2}\mathbf{A}\eta_k &= \frac{\eta_k^\top(\mathbf{A}^\top\mathbf{b}-\mathbf{A}^\top\mathbf{A}\mathbf{x}^{(k)})}{\|\mathbf{A}\eta_k\|_2^2}\mathbf{A}\eta_k = \frac{\eta_k^\top(\mathbf{A}^\top\mathbf{A}\mathbf{x}_\star-\mathbf{A}^\top\mathbf{A}\mathbf{x}^{(k)})}{\|\mathbf{A}\eta_k\|_2^2}\mathbf{A}\eta_k\\ &= \frac{\eta_k^\top\mathbf{A}^\top(\mathbf{A}\mathbf{x}_\star-\mathbf{A}\mathbf{x}^{(k)})}{\|\mathbf{A}\eta_k\|_2^2}\mathbf{A}\eta_k = \frac{\mathbf{A}\eta_k\eta_k^\top\mathbf{A}^\top(\mathbf{A}\mathbf{x}_\star-\mathbf{A}\mathbf{x}^{(k)})}{\|\mathbf{A}\eta_k\|_2^2}\\ &= -\mathbf{P}_k\big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big).\end{aligned} \tag{3.7}$$

Then we can deduce that

$$\begin{aligned}&\|\mathbf{x}^{(k+1)}-\mathbf{x}_\star\|_{\mathbf{A}^\top\mathbf{A}}^2 = \|\mathbf{A}(\mathbf{x}^{(k+1)}-\mathbf{x}_\star)\|_2^2\\ &= \big\|\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star) - \mathbf{P}_k\big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big) + \beta\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}^{(k-1)})\big\|_2^2\\ &= \big\|(\mathbf{I}-\mathbf{P}_k)\big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big) + \beta\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}^{(k-1)})\big\|_2^2\\ &= \big\|(\mathbf{I}-\mathbf{P}_k)\big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big)\big\|_2^2 + \beta^2\|\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}^{(k-1)})\|_2^2\\ &\quad + 2\beta\big\langle(\mathbf{I}-\mathbf{P}_k)\big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big), \mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}^{(k-1)})\big\rangle\\ &= \big\|(\mathbf{I}-\mathbf{P}_k)\big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big)\big\|_2^2 + \beta^2\|\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}^{(k-1)})\|_2^2\\ &\quad + 2\beta\langle\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star), \mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}^{(k-1)})\rangle\\ &\quad - 2\beta\big\langle\mathbf{P}_k\big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big), \mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}^{(k-1)})\big\rangle.\end{aligned} \tag{3.8}$$

We now carefully analyze each term on the right-hand side of the last equation in (3.8). By using $\mathbf{P}_k = \mathbf{P}_k^\top$ and $\mathbf{P}_k^2 = \mathbf{P}_k$, the first term satisfies

$$\begin{aligned}&\big\|(\mathbf{I}-\mathbf{P}_k)\big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big)\big\|_2^2\\ &= \big\langle(\mathbf{I}-\mathbf{P}_k)\big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big), (\mathbf{I}-\mathbf{P}_k)\big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big)\big\rangle\\ &= \big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big)^\top(\mathbf{I}-\mathbf{P}_k)^\top(\mathbf{I}-\mathbf{P}_k)\big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big)\\ &= \big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big)^\top(\mathbf{I}-\mathbf{P}_k)\big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big)\\ &= \big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big)^\top\big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big) - \big\langle\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star), \mathbf{P}_k\big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big)\big\rangle\\ &= \|\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\|_2^2 - \big\langle\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star), \mathbf{P}_k\big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big)\big\rangle.\end{aligned} \tag{3.9}$$

Then, by

$$\begin{aligned}&\big\langle\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star), \mathbf{P}_k\big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big)\big\rangle = \big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big)^\top\mathbf{P}_k\big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big)\\ &= \big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big)^\top\mathbf{P}_k^2\big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big) = \big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big)^\top\mathbf{P}_k^\top\mathbf{P}_k\big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big)\\ &= \big\langle\mathbf{P}_k\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star), \mathbf{P}_k\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big\rangle = \big\|\mathbf{P}_k\big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big)\big\|_2^2,\end{aligned} \tag{3.10}$$

and (3.9), it yields

$$\big\|(\mathbf{I}-\mathbf{P}_k)\big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big)\big\|_2^2=\|\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\|_2^2-\big\|\mathbf{P}_k\big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big)\big\|_2^2. \tag{3.11}$$

For the second term, by Cauchy's inequality we can deduce that

$$\begin{aligned}
&\beta^2\|\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}^{(k-1)})\|_2^2=\beta^2\|\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)+\mathbf{A}(\mathbf{x}_\star-\mathbf{x}^{(k-1)})\|_2^2\\
&=\beta^2\|\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\|_2^2+2\beta^2\big\langle\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star),\mathbf{A}(\mathbf{x}_\star-\mathbf{x}^{(k-1)})\big\rangle+\|\mathbf{A}(\mathbf{x}^{(k-1)}-\mathbf{x}_\star)\|_2^2\\
&\le\beta^2\|\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\|_2^2+2\beta^2\|\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\|_2^2\|\mathbf{A}(\mathbf{x}^{(k-1)}-\mathbf{x}_\star)\|_2^2+\beta^2\|\mathbf{A}(\mathbf{x}^{(k-1)}-\mathbf{x}_\star)\|_2^2\\
&\le 2\beta^2\|\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\|_2^2+2\beta^2\|\mathbf{A}(\mathbf{x}^{(k-1)}-\mathbf{x}_\star)\|_2^2.
\end{aligned} \tag{3.12}$$

The third term can be rewritten as

$$\begin{aligned}
&2\beta\big\langle\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star),\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}^{(k-1)})\big\rangle\\
&=\beta\big\langle\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star),\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}^{(k-1)})\big\rangle+\beta\big\langle\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star),\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}^{(k-1)})\big\rangle\\
&=\beta\big\langle\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star),\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star+\mathbf{x}_\star-\mathbf{x}^{(k-1)})\big\rangle\\
&\quad+\beta\big\langle\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}^{(k-1)}+\mathbf{x}^{(k-1)}-\mathbf{x}_\star),\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}^{(k-1)})\big\rangle\\
&=\beta\big(\|\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\|_2^2+\|\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}^{(k-1)})\|_2^2-\|\mathbf{A}(\mathbf{x}^{(k-1)}-\mathbf{x}_\star)\|_2^2\big).
\end{aligned} \tag{3.13}$$

For the last term, by (3.11) and the inequality $\|\mathbf{a}+\mathbf{b}\|_2^2\le 2\|\mathbf{a}\|_2^2+2\|\mathbf{b}\|_2^2$, we have the following estimate:

$$\begin{aligned}
&-2\beta\big\langle\mathbf{P}_k\big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big),\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}^{(k-1)})\big\rangle\\
&=\beta\Big(\big\|\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}^{(k-1)})-\mathbf{P}_k\big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big)\big\|^2\\
&\qquad-\|\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}^{(k-1)})\|_2^2-\big\|\mathbf{P}_k\big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big)\big\|_2^2\Big)\\
&=\beta\Big(\big\|(\mathbf{I}-\mathbf{P}_k)\big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big)-\mathbf{A}(\mathbf{x}^{(k-1)}-\mathbf{x}_\star)\big\|_2^2\\
&\qquad-\|\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}^{(k-1)})\|_2^2-\big\|\mathbf{P}_k\big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big)\big\|_2^2\Big)\\
&\le\beta\Big(2\big\|(\mathbf{I}-\mathbf{P}_k)\big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big)\big\|_2^2+2\|\mathbf{A}(\mathbf{x}^{(k-1)}-\mathbf{x}_\star)\|_2^2\\
&\qquad-\|\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}^{(k-1)})\|_2^2-\big\|\mathbf{P}_k\big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big)\big\|_2^2\Big)\\
&=\beta\Big(2\|\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\|_2^2-3\big\|\mathbf{P}_k\big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big)\big\|_2^2\\
&\qquad+2\|\mathbf{A}(\mathbf{x}^{(k-1)}-\mathbf{x}_\star)\|_2^2-\|\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}^{(k-1)})\|_2^2\Big).
\end{aligned} \tag{3.14}$$

Replacing (3.11)-(3.14) into (3.8), we obtain the following inequality:

$$\begin{aligned}
\|\mathbf{x}^{(k+1)}-\mathbf{x}_\star\|_{\mathbf{A}^\top\mathbf{A}}^2\le{}&(1+3\beta+2\beta^2)\|\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\|_2^2\\
&+(2\beta^2+\beta)\|\mathbf{A}(\mathbf{x}^{(k-1)}-\mathbf{x}_\star)\|_2^2\\
&-(3\beta+1)\big\|\mathbf{P}_k\big(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\big)\big\|_2^2.
\end{aligned} \tag{3.15}$$

We now establish inequality relationship between $\|\mathbf{P}_k(\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star))\|_2^2$ and $\|\mathbf{A}(\mathbf{x}^{(k)}-\mathbf{x}_\star)\|_2^2$ by detailed derivation. Let $\mathbf{E}_k\in\mathbb{R}^{n\times|\tau_k|}$ denote a matrix whose columns in turn consist of all vectors $\mathbf{e}_{j_k} \in \mathbb{R}^n$ with $j_k \in \tau_k$, then $\mathbf{A}_{\tau_k} = \mathbf{A}\mathbf{E}_k$ and $\mathbf{E}_k^\top \mathbf{E}_k = \mathbf{I}_{|\tau_k|}$, where $\mathbf{I}_{|\tau_k|} \in \mathbf{R}^{|\tau_k|\times|\tau_k|}$ is the identity matrix. Denote by $\hat{\eta}_k = \mathbf{E}_k^\top \eta_k$, we can get

$$\begin{aligned} \|\hat{\eta}_k\|_2^2 &= \left\|\mathbf{E}_k^\top \eta_k\right\|_2^2 = \left(\mathbf{E}_k^\top \eta_k\right)^\top \left(\mathbf{E}_k^\top \eta_k\right) = \eta_k^\top \mathbf{E}_k \mathbf{E}_k^\top \eta_k \\ &= \eta_k^\top \eta_k = \|\eta_k\|_2^2 = \sum_{j_k \in \tau_k} \left|(\mathbf{s}^{(k)})_{j_k}\right|^2. \end{aligned} \tag{3.16}$$

On the one hand, based on Lemma 2.1, we can deduce

$$\|\mathbf{A}_{\tau_k}\hat{\eta}_k\|_2^2 = (\mathbf{A}_{\tau_k}\hat{\eta}_k)^\top(\mathbf{A}_{\tau_k}\hat{\eta}_k) = \hat{\eta}_k^\top \mathbf{A}_{\tau_k}^\top \mathbf{A}_{\tau_k}\hat{\eta}_k \leq \sigma_{\max}^2(\mathbf{A}_{\tau_k})\|\hat{\eta}_k\|_2^2. \tag{3.17}$$

While on the other hand, we have

$$\begin{aligned} \|\mathbf{A}_{\tau_k}\hat{\eta}_k\|_2^2 &= \hat{\eta}_k^\top \mathbf{A}_{\tau_k}^\top \mathbf{A}_{\tau_k}\hat{\eta}_k = \left(\mathbf{E}_k^\top \eta_k\right)^\top \mathbf{A}_{\tau_k}^\top \mathbf{A}_{\tau_k} \mathbf{E}_k^\top \eta_k \\ &= \left(\mathbf{E}_k^\top \eta_k\right)^\top \mathbf{E}_k^\top \mathbf{A}^\top \mathbf{A} \mathbf{E}_k \mathbf{E}_k^\top \eta_k \\ &= \left(\mathbf{E}_k \mathbf{E}_k^\top \eta_k\right)^\top \mathbf{A}^\top \mathbf{A}\left(\mathbf{E}_k \mathbf{E}_k^\top \eta_k\right) \\ &= \eta_k^\top \mathbf{A}^\top \mathbf{A}\eta_k = \|\mathbf{A}\eta_k\|_2^2. \end{aligned} \tag{3.18}$$

Hence, based on Eqs. (3.17) and (3.18), we can derive the following inequality:

$$\|\mathbf{A}\eta_k\|_2^2 \leq \sigma_{\max}^2(\mathbf{A}_{\tau_k})\|\hat{\eta}_k\|_2^2. \tag{3.19}$$

From the definition of η_k in (3.2) as well as (3.16), it yields that

$$\begin{aligned} \eta_k^\top \mathbf{s}^{(k)} &= \left(\sum_{j_k \in \tau_k} (\mathbf{s}^{(k)})_{j_k} \mathbf{e}_{j_k}^\top\right) \mathbf{s}^{(k)} = \sum_{j_k \in \tau_k} \left((\mathbf{s}^{(k)})_{j_k} \mathbf{e}_{j_k}^\top \mathbf{s}^{(k)}\right) \\ &= \sum_{j_k \in \tau_k} \left|(\mathbf{s}^{(k)})_{j_k}\right|^2 = \|\eta_k\|_2^2 = \|\hat{\eta}_k\|_2^2. \end{aligned} \tag{3.20}$$

It can be inferred from Lemma 2.1 that

$$\begin{aligned} \|\mathbf{s}^{(k)}\|_2^2 &= \|\mathbf{A}^\top(\mathbf{b} - \mathbf{A}\mathbf{x}^{(k)})\|_2^2 = \|\mathbf{A}^\top \mathbf{A}(\mathbf{x}_\star - \mathbf{x}^{(k)})\|_2^2 \\ &\geq \sigma_{\min}^2(\mathbf{A}^\top)\|\mathbf{A}(\mathbf{x}^{(k)} - \mathbf{x}_\star)\|_2^2 \\ &= \sigma_{\min}(\mathbf{A}^\top \mathbf{A})\|\mathbf{A}(\mathbf{x}^{(k)} - \mathbf{x}_\star)\|_2^2 = \sigma_{\min}^2(\mathbf{A})\|\mathbf{A}(\mathbf{x}^{(k)} - \mathbf{x}_\star)\|_2^2. \end{aligned} \tag{3.21}$$

According to (3.21) and the definition of the set τ_k in (3.1), it holds that

$$\begin{aligned} \|\eta_k\|_2^2 = \eta_k^\top \mathbf{s}^{(k)} &= \sum_{j_k \in \tau_k} \left|(\mathbf{s}^{(k)})_{j_k}\right|^2 \\ &\geq |\tau_k| \frac{\|\mathbf{s}^{(k)}\|_2^2}{n} \geq |\tau_k| \frac{\sigma_{\min}(\mathbf{A}^\top \mathbf{A})}{n} \|\mathbf{A}(\mathbf{x}^{(k)} - \mathbf{x}_\star)\|_2^2 \\ &= |\tau_k| \frac{\sigma_{\min}^2(\mathbf{A})}{n} \|\mathbf{A}(\mathbf{x}^{(k)} - \mathbf{x}_\star)\|_2^2. \end{aligned} \tag{3.22}$$

Therefore, from Eqs. (3.19), (3.20) and (3.22), we have

$$\begin{aligned} &\left\|\mathbf{P}_k\left(\mathbf{A}(\mathbf{x}^{(k)} - \mathbf{x}_\star)\right)\right\|_2^2 \\ &= \frac{\left|\eta_k \mathbf{s}^{(k)}\right|^2}{\|\mathbf{A}\eta_k\|_2^2} = \frac{\left|\eta_k \mathbf{s}^{(k)}\right| \|\hat{\eta}_k\|_2^2}{\|\mathbf{A}\eta_k\|_2^2} \\ &\geq \frac{\|\eta_k\|_2^2}{\sigma_{\max}^2(\mathbf{A}_{\tau_k})} \geq \frac{|\tau_k|\sigma_{\min}^2(\mathbf{A})}{n\sigma_{\max}^2(\mathbf{A}_{\tau_k})} \|\mathbf{A}(\mathbf{x}^{(k)} - \mathbf{x}_\star)\|_2^2. \end{aligned} \tag{3.23}$$

By substituting (3.23) into (3.15), the following error estimation equation:

$$\begin{aligned}
\|\mathbf{x}^{(k+1)} - \mathbf{x}_\star\|^2_{\mathbf{A}^\top\mathbf{A}} &\le (1+3\beta+2\beta^2)\|\mathbf{A}(\mathbf{x}^{(k)} - \mathbf{x}_\star)\|_2^2 + (2\beta^2+\beta)\|\mathbf{A}(\mathbf{x}^{(k-1)} - \mathbf{x}_\star)\|_2^2 \\
&\quad - (3\beta+1)\big\|\mathbf{P}_k\big(\mathbf{A}(\mathbf{x}^{(k)} - \mathbf{x}_\star)\big)\big\|_2^2 \\
&\le \left(1+3\beta+2\beta^2 - (3\beta+1)\frac{|\tau_k|\sigma^2_{\min}(\mathbf{A})}{n\sigma^2_{\max}(\mathbf{A}_{\tau_k})}\right)\|\mathbf{A}(\mathbf{x}^{(k)} - \mathbf{x}_\star)\|_2^2 \\
&\quad + (2\beta^2+\beta)\|\mathbf{A}(\mathbf{x}^{(k-1)} - \mathbf{x}_\star)\|_2^2 \\
&= \gamma_1\|\mathbf{A}(\mathbf{x}^{(k)} - \mathbf{x}_\star)\|_2^2 + \gamma_2\|\mathbf{A}(\mathbf{x}^{(k-1)} - \mathbf{x}_\star)\|_2^2 \\
&= \gamma_1\|\mathbf{x}^{(k)} - \mathbf{x}_\star\|^2_{\mathbf{A}^\top\mathbf{A}} + \gamma_2\|\mathbf{x}^{(k-1)} - \mathbf{x}_\star\|^2_{\mathbf{A}^\top\mathbf{A}} \qquad (3.24)
\end{aligned}$$

can be derived. For any nonzero vector $\hat{\xi} \in \mathbb{R}^{|\tau_k|}$, it follows that

$$\begin{aligned}
\sigma^2_{\min}(\mathbf{A}) = \sigma_{\min}(\mathbf{A}^\top\mathbf{A}) &\le \frac{(\mathbf{E}_k\hat{\xi})^\top\mathbf{A}^\top\mathbf{A}(\mathbf{E}_k\hat{\xi})}{(\mathbf{E}_k\hat{\xi})^\top(\mathbf{E}_k\hat{\xi})} = \frac{\hat{\xi}^\top\mathbf{E}_k^\top\mathbf{A}^\top\mathbf{A}\mathbf{E}_k\hat{\xi}}{\hat{\xi}^\top\mathbf{E}_k^\top\mathbf{E}_k\hat{\xi}} \\
&= \frac{\hat{\xi}^\top\mathbf{A}_{\tau_k}^\top\mathbf{A}_{\tau_k}\hat{\xi}}{\hat{\xi}^\top\hat{\xi}} \le \sigma_{\max}\big(\mathbf{A}_{\tau_k}^\top\mathbf{A}_{\tau_k}\big) = \sigma^2_{\max}(\mathbf{A}_{\tau_k}). \qquad (3.25)
\end{aligned}$$

Accordingly, $(|\tau_k|\sigma^2_{\min}(\mathbf{A}))/(n\sigma^2_{\max}(\mathbf{A}_{\tau_k})) \in [0,1]$ holds, which makes $\gamma_1 \ge 0$. Let

$$F^{(k)} := \|\mathbf{x}^{(k)} - \mathbf{x}_\star\|^2_{\mathbf{A}^\top\mathbf{A}},$$

then (3.14) can be written as

$$F^{(k+1)} \le \gamma_1 F^{(k)} + \gamma_2 F^{(k-1)}.$$

The admissible range of β ensures $\gamma_2 \ge 0$. If $\gamma_2 = 0$, then $\beta = 0$ and $q = \gamma_1 \ge 0$. Since $\gamma_1 + \gamma_2 < 1$ by assumption, the conditions of Lemma 2.2 are satisfied. Consequently, Lemma 2.2 guarantees the error bound (3.6) and the convergence of the mADBCD iterates $\{\mathbf{x}^{(k)}\}_{k=0}^\infty$ to $\mathbf{x}_\star$. The proof is complete. □

Remark 3.2. Next, we discuss the existence of $\beta > 0$ in Theorem 3.1 to ensure that the crucial condition $\gamma_1 + \gamma_2 < 1$ holds. Let

$$\alpha_k = \frac{|\tau_k|\sigma^2_{\min}(A)}{n\sigma^2_{\max}(A_{\tau_k})},$$

then $\gamma_1 + \gamma_2 < 1$ can be equivalently rewritten as the quadratic inequality

$$4\beta^2 + (4 - 3\alpha_k)\beta - \alpha_k < 0.$$

We analyze the roots of the corresponding quadratic equation

$$4\beta^2 + (4 - 3\alpha_k)\beta - \alpha_k = 0.$$

By the quadratic formula, the roots are

$$\beta = \frac{-(4-3\alpha_k) \pm \sqrt{\Delta_k}}{8},$$

where the discriminant is

$$\Delta_k = (4-3\alpha_k)^2 + 16\alpha_k = 9\alpha_k^2 - 8\alpha_k + 16.$$

We evaluate Δ_k at the boundary points and critical point:

- At $\alpha_k = 0$, $\Delta_k = 16$.
- At $\alpha_k = 1$, $\Delta_k = 9(1)^2 - 8(1) + 16 = 17$.
- Finding the critical point, we differentiate

$$\frac{d}{d\alpha_k}\left(9\alpha_k^2 - 8\alpha_k + 16\right) = 18\alpha_k - 8.$$

  Setting it to zero gives

$$18\alpha_k - 8 = 0 \ \Rightarrow\ \alpha_k = \frac{4}{9}.$$

  Evaluating Δ_k at $\alpha_k = 4/9$,

$$\Delta_k = 9\left(\frac{4}{9}\right)^2 - 8\left(\frac{4}{9}\right) + 16.$$

  Computing,

$$\Delta_k = \frac{144}{81} - \frac{288}{81} + \frac{1296}{81} = \frac{1152}{81} \approx 14.22.$$

Thus, the range of Δ_k is approximately

$$14.22 \lessapprox \Delta_k \leq 17.$$

Since $\Delta_k > 0$ for all $\alpha_k \in [0,1]$, the quadratic equation always has two real roots. Among the two roots, the nonnegative root is given by

$$\beta_1 = \frac{-(4-3\alpha_k) + \sqrt{\Delta_k}}{8}.$$

It is easy to see that $\beta_1 > 0$, which confirms the existence of $\beta > 0$ such that $\gamma_1 + \gamma_2 < 1$ holds.

Remark 3.3. When $\beta = 0$, basis on (3.24) it can be easily deduced that

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}_\star\|^2_{\mathbf{A}^\top\mathbf{A}} \leq \left(1 - \frac{|\tau_k|\sigma^2_{\min}(\mathbf{A})}{n\sigma^2_{\max}(\mathbf{A}_{\tau_k})}\right)\|\mathbf{x}^{(k)} - \mathbf{x}_\star\|^2_{\mathbf{A}^\top\mathbf{A}}.$$

Theorem 3.1 establishes that the upper bound on the convergence rate of the mADBCD method depends on the momentum parameter β, the minimum singular value of the system matrix in the linear least-squares problem (1.1), and the maximum singular value of the selected column submatrix. It is important to note, however, that this bound may be conservative, and in practice, the mADBCD method often converges significantly faster than the theoretical estimate suggests.

4. Numerical Experiments

In this section, we present a series of numerical experiments to demonstrate the superior performance of the proposed mADBCD method, in comparison with several recently developed block coordinate descent algorithms – including GBGS [14], MRBGS [16], and FBCD [4] – as well as the classical method LSQR [22]. The performance of these methods is evaluated in terms of the average number of iterations (denoted by "IT") and average CPU time in seconds (denoted by "CPU"). Each method is executed 10 times, and both IT and CPU represent the arithmetic means of the corresponding results. All experiments are conducted on a Windows 10 machine with an Intel(R) Core(TM) i7-10510U processor (1.83 GHz) and 16 GB RAM, using MATLAB R2020b with machine precision 10^{-16}. In this experiment, we consider solving the following four different linear least-squares problems.

Example 4.1. Linear least-squares problems with their coefficient matrices being twenty dense matrices generated by using the MATLAB function `randn`(m, n) and the right-hand side vectors b being set as $\mathbf{b} = \mathbf{A}\mathbf{x}_\star$, where the solution $\mathbf{x}_\star$ is randomly generated by the function `randn`$(n, 1)$.

Example 4.2. Linear least-squares problems with their coefficient matrices being twenty sparse matrices from the Florida sparse matrix collection [5] with their dimensions, densities and condition numbers listed in Table 4.5 and the right-hand side vectors b being set as Example 4.1.

Example 4.3. Image reconstruction problems from the 2D seismic travel-time tomography by using the function `seismictomo`(N, s, p) in the MATLAB AIR Tools package [9] with $N = 50$, $s = 80$ and $p = 120$. That means a sparse matrix with 9600 rows and 2500 columns is generated. The right-hand side vectors b are set as $\mathbf{b} = \mathbf{A}\mathbf{x}_\star$ with the true solutions $\mathbf{x}_\star$ shown as images in Fig. 4.17 (top left).

Example 4.4. The Shepp-Logan medical model problems generated by function `fancurved-tomo`(N, θ, P) in the MATLAB package AIR Tools II [9] with $N = 50, \theta = 0 : 1 : 300^\circ$ and $P = 50$, which generate a sparse matrix with 15050 rows and 2500 columns. A corresponding true solution $\mathbf{x}_\star$ is shown as images in Fig. 4.18 (top left) and $\mathbf{b} = \mathbf{A}\mathbf{x}_\star$.

All methods start with an initial vector $\mathbf{x}^{(0)} = \mathbf{0}$. The termination criterion for Examples 4.1 and 4.2 is that the relative solution error (RSE) at the approximate solution $\mathbf{x}^{(k)}$ satisfies

$$\text{RSE} = \frac{\|\mathbf{x}^{(k)} - \mathbf{x}_\star\|_2^2}{\|\mathbf{x}_\star\|_2^2} < 10^{(-6)}, \tag{4.1}$$

while for the examples of image reconstruction (Examples 4.3 and 4.4), it is based on the CPU time reaching 180 seconds.

Both the GBGS and MRBGS methods require computing the product $\mathbf{A}{\tau_k}^\dagger \mathbf{r}^{(k)}$, where $\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}$ and $\mathbf{A}{\tau_k}^\dagger$ is the Moore-Penrose pseudoinverse of $\mathbf{A}\tau_k$. To reduce computational cost, we use the MATLAB function `lsqminnorm`$(\mathbf{A}\tau_k, \mathbf{r}^{(k)})$ rather than computing `pinv`$(\mathbf{A}_{\tau_k})$ explicitly and then multiplying it by $\mathbf{r}^{(k)}$. For the MRBGS method, the block index set is defined as

$$\tau_k = \left\{ j_k \,\middle|\, \left|(\mathbf{s}^{(k)})_{j_k}\right|^2 \geq 0.3 \max_{1 \leq j \leq n} \left|(\mathbf{s}^{(k)})_j\right|^2 \right\},$$

as in the numerical setup of [16]. The parameter θ in the GBGS method is set according to [14]. In our experiments, we observed that the mADBCD method tends to diverge when the momentum parameter β exceeds 0.9. Therefore, all implementations use $\beta \in [0, 0.9]$.

We compare the numerical performances of the GBGS, MRBGS, FBCD and mADBCD methods on Examples 4.1-4.4, see Tables 4.1-4.4 and 4.6-4.9. To better compare the convergence speeds of these four methods conveniently, we also present the speed-up of the mADBCD method against the other methods, defined by

$$\text{speed-up\_Method} = \frac{\text{CPU of Method}}{\text{CPU of mADBCD}}.$$

Figs. 4.1-4.5 illustrate the effect of the momentum parameter β on the iteration count and CPU time of the mADBCD method. When the row-to-column ratio $m/n \geq 5$ (Figs. 4.1-4.2), the method shows low sensitivity to β for $\beta \leq 0.5$, with the optimal value typically lying in $[0, 0.3]$, as also supported by Tables 4.1-4.2. However, for $\beta \in [0.55, 0.9]$, both metrics increase with β. In contrast, when $m/n \leq 0.5$ (Figs. 4.3-4.5), the performance becomes more sensitive to β. In this regime, increasing β first improves and then degrades the efficiency of the method. The best results are observed for $\beta \in [0.5, 0.65]$, as confirmed by Tables 4.3-4.4.

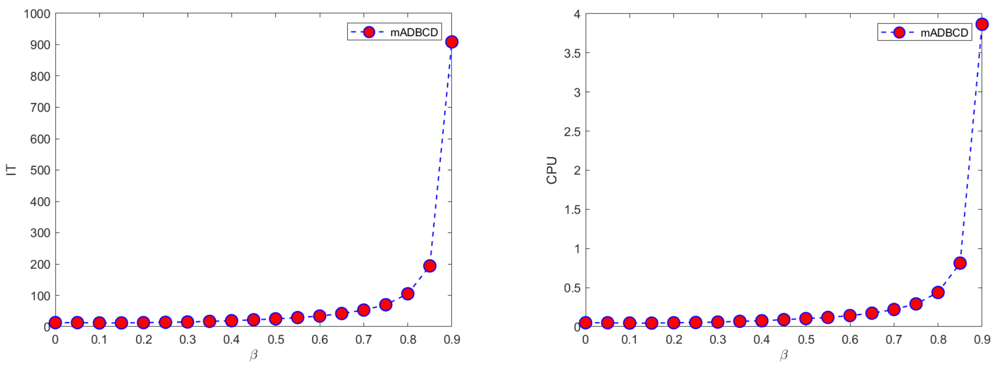

Fig. 4.1. Pictures of β versus IT (left) and CPU (right) for mADBCD when $\mathbf{A}$=`randn`$(7500, 750)$.

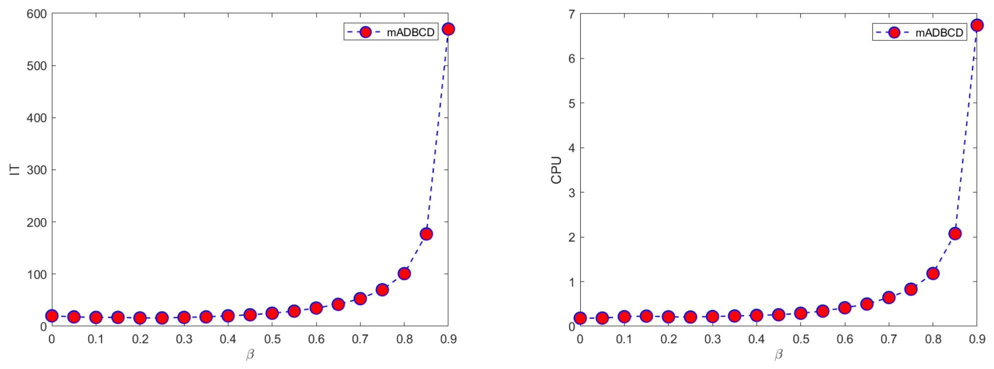

Fig. 4.2. Pictures of β versus IT (left) and CPU (right) for mADBCD when $\mathbf{A}$= `randn`$(7500, 1500)$.

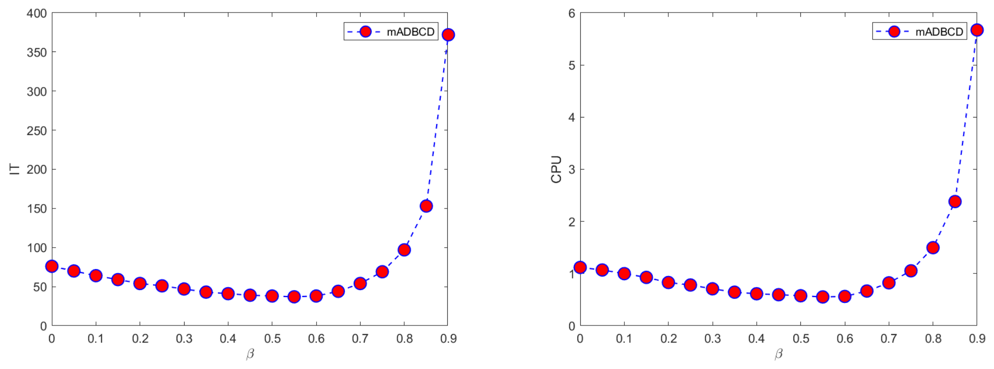

Fig. 4.3. Pictures of β versus IT (left) and CPU (right) for mADBCD when $\mathbf{A}$=`randn`$(6000, 3000)$.

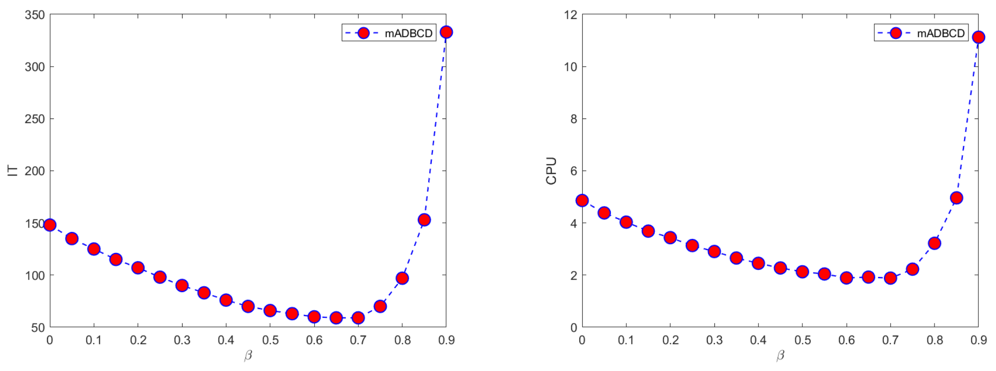

Fig. 4.4. Pictures of β versus IT (left) and CPU (right) for mADBCD when $\mathbf{A}=$ `randn`$(8000, 5000)$.

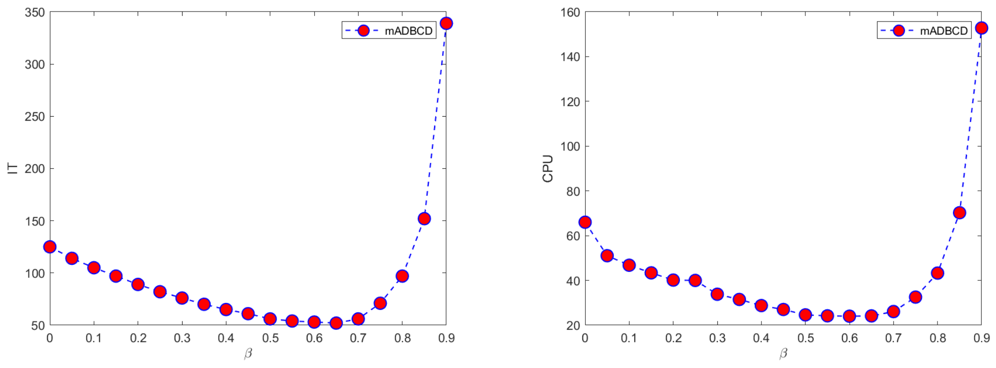

Fig. 4.5. Pictures of β versus IT (left) and CPU (right) for mADBCD when $\mathbf{A}=$`randn`$(21000, 12500)$.

Table 4.1: IT and CPU of GBGS, MRBGS, FBCD and mADBCD for different $\mathbf{A}=$`randn`(m, n).

	m	3500	4500	5500	6500	7500
	n	350	450	550	650	750
GBGS	IT	47	51	49	54	52
	CPU	0.0896	0.1476	0.2191	0.5432	0.6154
MRBGS	IT	20	22	22	22	22
	CPU	0.0614	0.1297	0.1909	0.3065	0.4079
FBCD	IT	48	53	53	53	52
	CPU	0.0275	0.0742	0.1189	0.1901	0.3318
mADBCD	β	0.10	0.20	0.10	0.15	0.15
	IT	12	13	12	12	12
	CPU	0.0055	0.0163	0.0246	0.0351	0.0486
speed-up_GBGS		16.29	9.06	8.91	15.47	12.66
speed-up_MRBGS		11.16	7.96	7.76	8.73	8.39
speed-up_FBCD		5.00	4.55	4.83	5.42	6.83

Tables 4.1-4.4 show that all four block iterative methods – GBGS, MRBGS, FBCD, and mADBCD – successfully converge on the 20 tested least-squares problems. Among them, the mADBCD method achieves the best overall performance in both iteration count and CPU time when the momentum parameter β is properly chosen. The FBCD method consistently

Table 4.2: IT and CPU of GBGS, MRBGS, FBCD and mADBCD for different $\mathbf{A}$=`randn`(m, n).

	m n	3500 700	4500 900	5500 1100	6500 1300	7500 1500
GBGS	IT CPU	84 0.4387	89 0.5182	99 1.2411	94 2.2123	99 3.3781
MRBGS	IT CPU	32 0.3575	34 0.4770	36 0.7900	35 1.9296	37 2.6369
FBCD	IT CPU	85 0.1582	96 0.3426	97 0.5269	95 1.0664	100 1.3669
mADBCD	β IT CPU	0.25 16 0.0256	0.25 16 0.0480	0.25 16 0.0744	0.30 16 0.1042	0.25 16 0.1436
speed-up_GBGS speed-up_MRBGS speed-up_FBCD		17.14 13.96 6.18	10.78 9.94 7.14	16.68 10.62 7.08	21.23 18.52 10.23	23.52 18.36 9.52

Table 4.3: IT and CPU of GBGS, MRBGS, FBCD and mADBCD for different $\mathbf{A}$=`randn`(m, n).

	m n	4000 1000	5000 2000	6000 3000	7000 4000	8000 5000
GBGS	IT CPU	112 0.5404	283 5.6765	508 12.4093	768 59.3751	1164 141.3525
MRBGS	IT CPU	42 0.4224	78 3.6299	129 23.7199	190 100.0796	277 277.5756
FBCD	IT CPU	122 0.3635	285 2.2697	505 7.4719	783 36.4584	1166 79.3965
mADBCD	β IT CPU	0.30 18 0.0516	0.45 27 0.2069	0.55 37 0.5545	0.65 47 1.0630	0.65 59 1.9208
speed-up_GBGS speed-up_MRBGS speed-up_FBCD		10.47 8.19 7.04	27.44 17.54 10.97	22.38 42.78 13.48	55.86 94.15 34.30	73.59 144.51 41.34

Table 4.4: IT and CPU of GBGS, MRBGS, FBCD and mADBCD for different $\mathbf{A}$=`randn`(m, n).

	m n	10000 5000	13000 6500	16000 8500	19000 10500	21000 12500
GBGS	IT CPU	554 71.5051	597 163.6368	738 265.7505	835 403.0384	1110 1045.7000
MRBGS	IT CPU	142 140.5002	153 358.1978	180 608.0710	198 1083.0000	255 2409.0000
FBCD	IT CPU	552 28.5252	613 83.6194	756 154.0855	852 213.7058	1103 437.9306
mADBCD	β IT CPU	0.50 37 1.6140	0.55 37 2.6188	0.60 41 5.0988	0.50 44 7.5382	0.65 52 12.7072
speed-up_GBGS speed-up_MRBGS speed-up_FBCD		44.30 87.05 17.67	62.49 136.78 31.93	52.12 119.26 30.22	53.47 143.67 28.35	82.29 189.58 34.46

outperforms GBGS and MRBGS in convergence speed. Moreover, when $m/n \leq 0.5$, GBGS outperforms MRBGS (see Tables 4.3 and 4.4). The speed-up ranges relative to mADBCD for GBGS, MRBGS, and FBCD lie in $[8.91, 82.29]$, $[7.76, 189.58]$, and $[4.55, 34.46]$, respectively. These results further confirm that mADBCD delivers the best numerical performance among the four methods, followed by FBCD.

Figs. 4.6-4.7 report the relative solution error (RSE) versus iteration count (left) and CPU time (right) for GBGS, MRBGS, FBCD, LSQR, and mADBCD on dense least-squares problems. Among these methods, LSQR exhibits the fastest convergence, achieving both the fewest iterations and the lowest CPU time under the termination criterion (4.1). The proposed mADBCD method performs slightly behind LSQR. In contrast, for highly sparse problems (Figs. 4.13-4.16), while LSQR still requires the fewest iterations, the CPU time of mADBCD is nearly comparable, underscoring its practical efficiency in sparse regimes.

Figs. 4.8-4.12 demonstrate that the momentum parameter β has a substantial influence on the performance of the mADBCD method when solving sparse least-squares problems in Example 4.2. A well-chosen β can significantly accelerate convergence and improve overall efficiency. Tables 4.6-4.9 indicate that the mADBCD method delivers the most competitive numerical performance, followed by FBCD, with GBGS being the least effective. Although GBGS and MRBGS require fewer iterations than FBCD, FBCD is more efficient in terms of

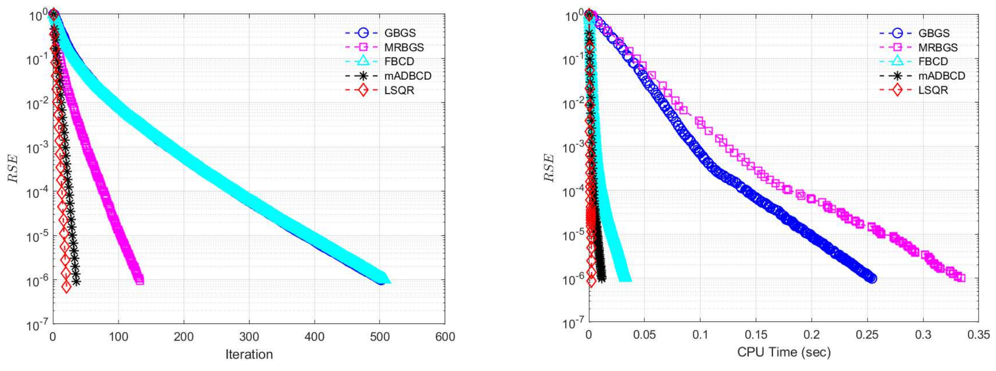

Fig. 4.6. RSE versus IT (left) and CPU (right) of GBGS, MRBGS, FBCD, LSQR and mADBCD for coefficient matrix $\mathbf{A}$=$\texttt{randn}(6000, 3000)$.

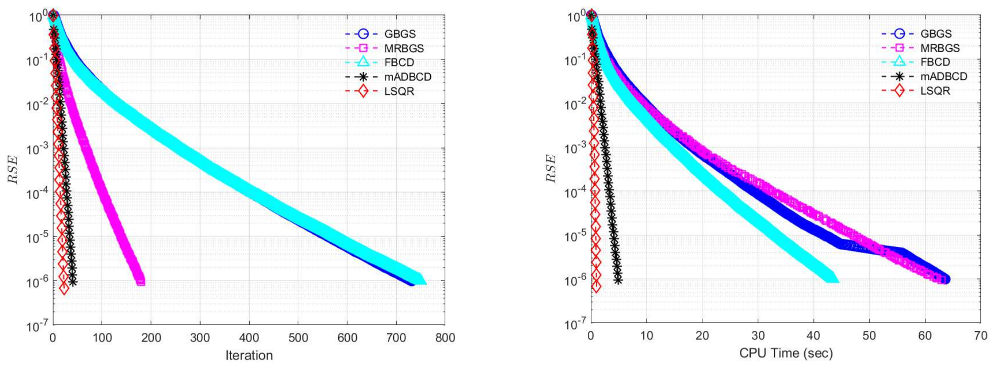

Fig. 4.7. RSE versus IT (left) and CPU (right) of GBGS, MRBGS, FBCD, LSQR and mADBCD for coefficient matrix $\mathbf{A}$=$\texttt{randn}(16000, 8500)$.

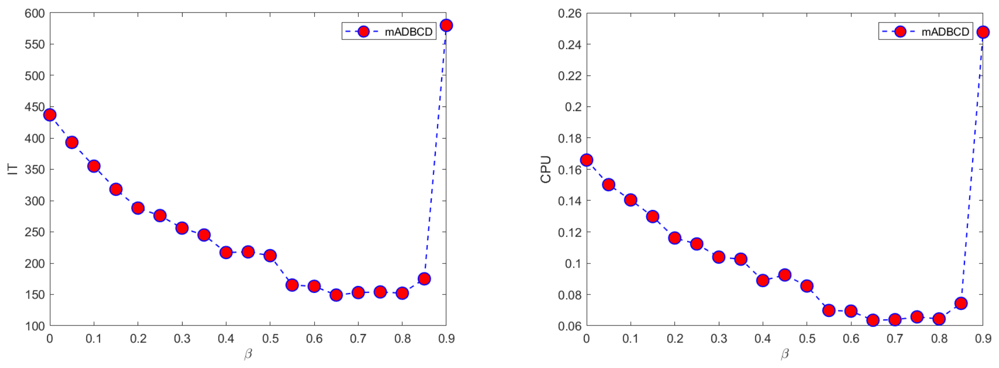

Fig. 4.8. Pictures of β versus IT (left) and CPU (right) for mADBCD when $\mathbf{A}$ is abtaha2.

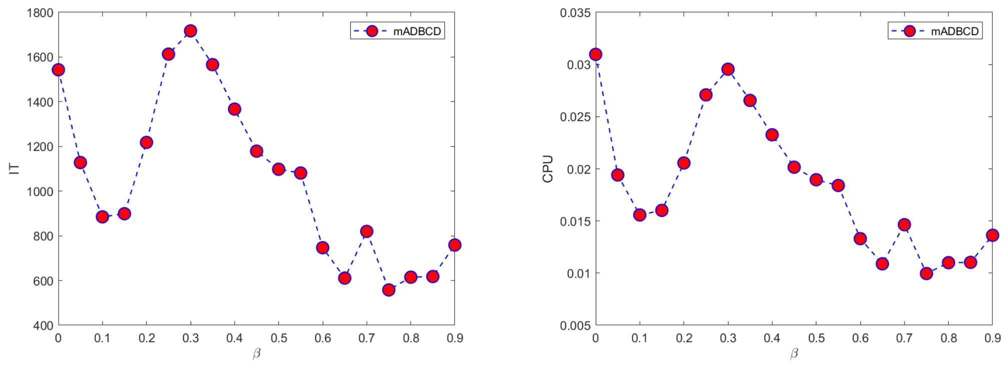

Fig. 4.9. Pictures of β versus IT (left) and CPU (right) for mADBCD when $\mathbf{A}$ is WorldCities.

Table 4.5: Information of the matrices from Florida space matrix collection.

Name	ash958	abtaha1	abtaha2	ash608	WorldCities
$m \times n$	958×292	14596×209	37932×331	608×188	315×100
density	0.68%	1.68%	1.09%	1.06%	23.87%
cond($\mathbf{A}$)	3.20	12.23	12.22	3.37	66.00
Name	well1033	well1850	rail516	rail582	r05
$m \times n$	1033×320	1850×712	516×47827	582×56097	5190×9690
density	1.43%	0.66%	1.28%	1.28%	1.23%
cond($\mathbf{A}$)	166.13	11.13	143.89	185.91	121.82
Name	cage8	cage9	cage10	nemsafm	lp22
$m \times n$	1015×1015	5226×14721	11397×11397	334×2348	2958×16392
density	1.07%	0.33%	0.16%	0.36%	0.14%
cond($\mathbf{A}$)	11.41	12.60	11.02	4.77	25.78
Name	model1	model8	nemscem	P05	pgp2
$m \times n$	362×798	2896×6464	651×1712	5090×9590	4034×13254
density	1.05%	0.14%	0.31%	0.12%	0.14%
cond($\mathbf{A}$)	17.57	53.63	44.63	85.37	31.95

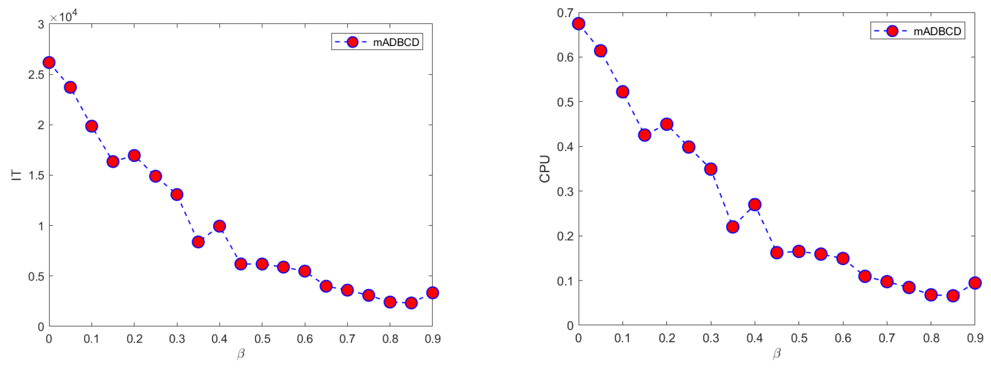

Fig. 4.10. Pictures of β versus IT (left) and CPU (right) for mADBCD when $\mathbf{A}$ is well1850.

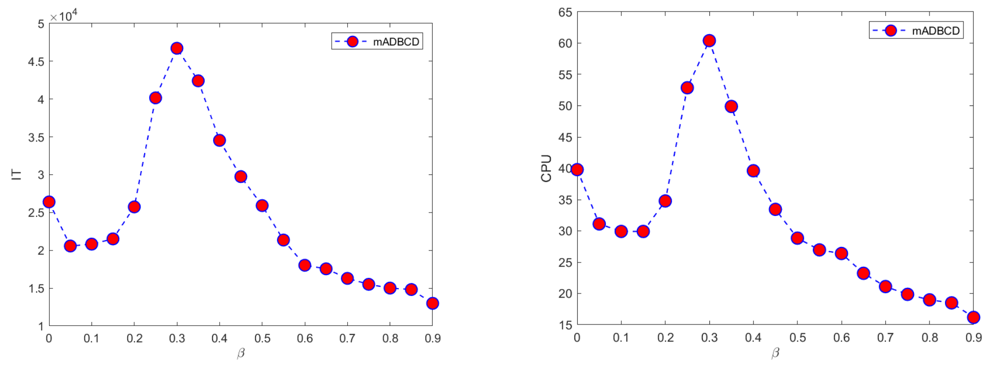

Fig. 4.11. Pictures of β versus IT (left) and CPU (right) for mADBCD when $\mathbf{A}$ is rail516$^\top$.

Table 4.6: Numerical results of GBGS, MRBGS, FBCD and mADBCD for Florida sparse matrix collection.

Name		ash958	abtaha1	abtaha2	ash608	WorldCities
GBGS	IT	53	771	1277	47	1448
	CPU	0.0078	1.6182	5.8035	0.0049	0.4238
MRBGS	IT	24	345	498	23	1338
	CPU	0.0060	1.3409	5.6426	0.0029	0.4141
FBCD	IT	69	450	1245	63	2078
	CPU	0.0022	0.0699	0.9450	0.0013	0.0536
mADBCD	β	0.30	0.35	0.65	0.30	0.75
	IT	17	116	149	19	558
	CPU	0.0005	0.0196	0.0540	0.0005	0.0127
speed-up_GBGS		15.60	82.56	107.47	9.80	33.37
speed-up_MRBGS		12.00	68.41	104.49	5.80	32.61
speed-up_FBCD		4.40	3.57	17.50	2.60	4.22

CPU time. The speed-up ranges of GBGS, MRBGS, and FBCD relative to mADBCD lie in $[9.80, 668.03]$, $[5.80, 546.07]$, and $[2.60, 81.86]$.

For the image reconstruction problems in Examples 4.3 and 4.4, the GBGS, MRBGS, FBCD, and mADBCD methods are run for 180 seconds, with results shown in Figs. 4.17 and 4.18,

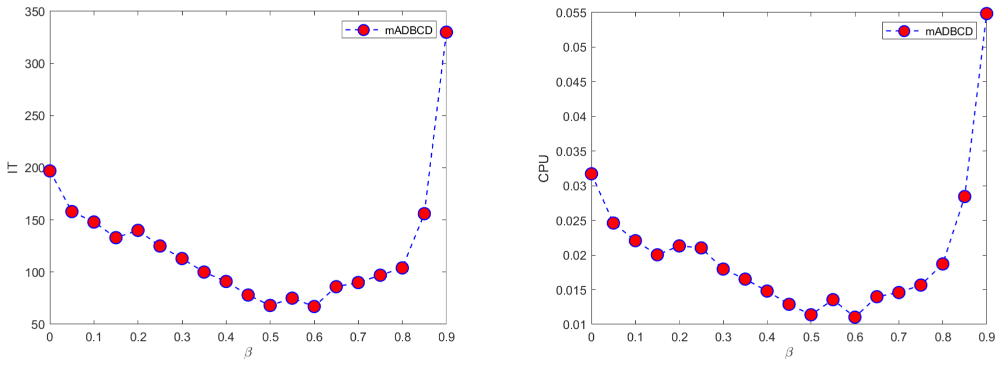

Fig. 4.12. Pictures of β versus IT (left) and CPU (right) for mADBCD when $\mathbf{A}$ is cage9.

Table 4.7: Numerical results of GBGS, MRBGS, FBCD and mADBCD for Florida sparse matrix collection.

Name		well1033	well1850	rail516$^\top$	rail582$^\top$	r05$^\top$
GBGS	IT	73057	113952	62672	101317	19922
	CPU	8.2718	32.5437	434.3091	716.4277	39.4501
MRBGS	IT	50625	22714	27246	96403	4921
	CPU	5.9303	12.2070	362.0436	1164.4000	17.6058
FBCD	IT	103731	142306	94850	165856	52648
	CPU	1.7977	4.2263	124.6174	219.4461	21.1645
mADBCD	β	0.90	0.85	0.90	0.90	0.90
	IT	6927	2334	12990	19866	2387
	CPU	0.1039	0.0624	13.5245	24.7864	0.7799
speed-up_GBGS		79.61	521.53	32.11	28.90	50.58
speed-up_MRBGS		57.08	195.63	26.77	46.98	22.57
speed-up_FBCD		17.30	67.73	9.21	8.85	27.14

Table 4.8: Numerical results of GBGS, MRBGS, FBCD and mADBCD for Florida sparse matrix collection.

Name		cage8	cage9	cage10	nemsafm$^\top$	lp22$^\top$
GBGS	IT	109	227	145	73	19922
	CPU	0.0445	0.3244	1.7932	0.0191	39.4501
MRBGS	IT	61	97	53	41	4921
	CPU	0.0496	0.2945	1.7576	0.0086	17.6058
FBCD	IT	280	425	250	132	52648
	CPU	0.0101	0.0550	0.1599	0.0026	21.1645
mADBCD	β	0.55	0.60	0.45	0.45	0.05
	IT	83	67	52	31	6028
	CPU	0.0026	0.0088	0.0238	0.0005	1.2944
speed-up_GBGS		17.11	36.86	75.34	38.20	47.39
speed-up_MRBGS		19.08	33.47	73.85	17.20	45.54
speed-up_FBCD		3.88	6.25	6.72	5.20	11.21

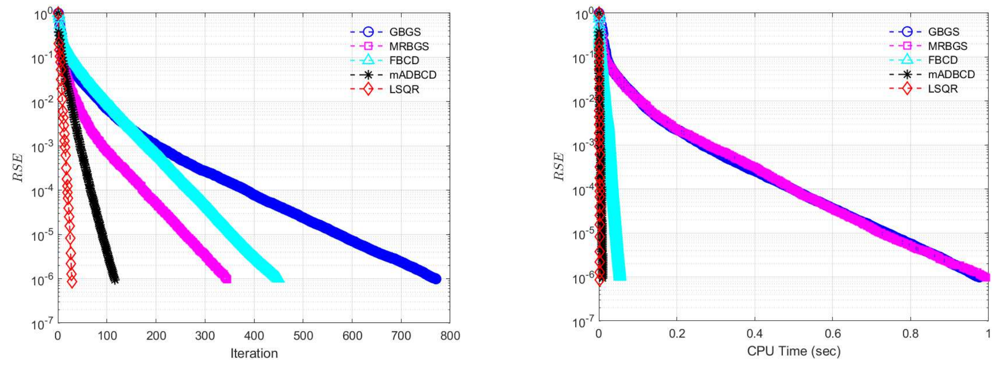

Fig. 4.13. RSE versus IT (left) and CPU (right) of GBGS, MRBGS, FBCD, LSQR and mADBCD for coefficient matrix abtaha1.

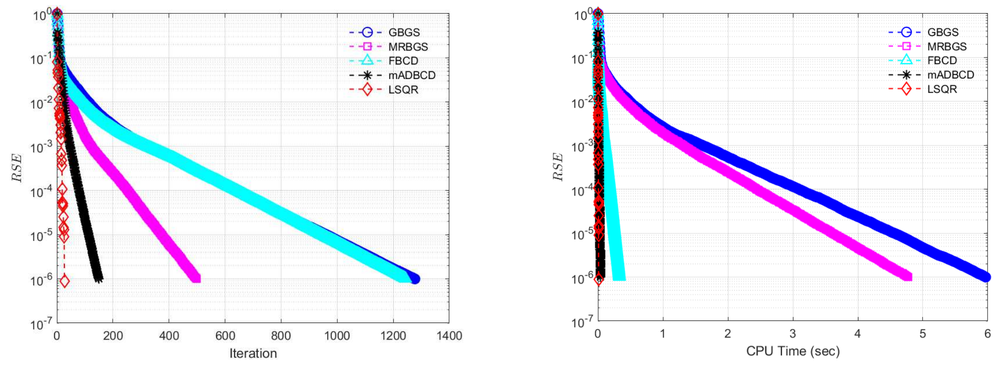

Fig. 4.14. RSE versus IT (left) and CPU (right) of GBGS, MRBGS, FBCD, LSQR and mADBCD for coefficient matrix abtaha2.

Table 4.9: Numerical results of GBGS, MRBGS, FBCD and mADBCD for Florida sparse matrix collection.

Name		model1$^\top$	model8$^\top$	nemscem$^\top$	p05$^\top$	pgp2$^\top$
GBGS	IT	485	49551	1090	16581	1894
	CPU	0.0701	41.7820	0.2240	34.9072	36.2071
MRBGS	IT	177	8214	519	4685	989
	CPU	0.0417	12.7197	0.1398	21.3427	29.5970
FBCD	IT	776	60880	2705	48300	2512
	CPU	0.0195	5.9184	0.0513	18.4968	0.5408
mADBCD	β	0.80	0.90	0.80	0.90	0.80
	IT	118	832	620	1354	388
	CPU	0.0013	0.0723	0.0089	0.3384	0.0542
speed-up_GBGS		53.92	577.90	25.17	103.15	668.03
speed-up_MRBGS		32.08	175.93	15.71	63.07	546.07
speed-up_FBCD		15.00	81.86	5.76	54.66	9.98

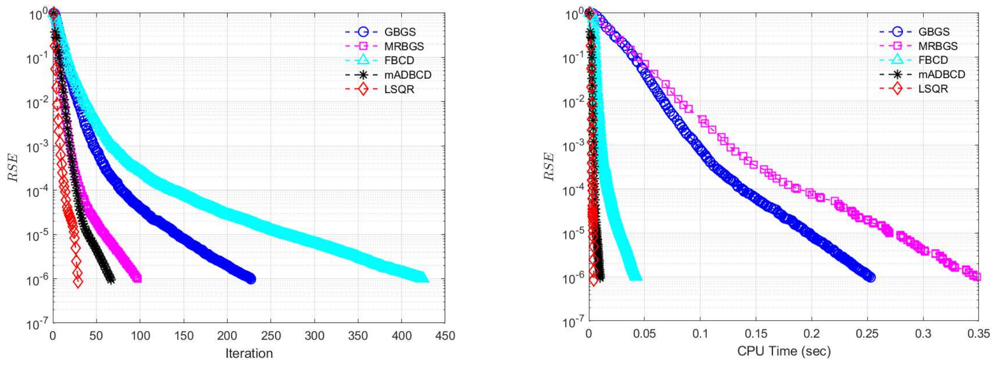

Fig. 4.15. RSE versus IT (left) and CPU (right) of GBGS, MRBGS, FBCD, LSQR and mADBCD for coefficient matrix cage9.

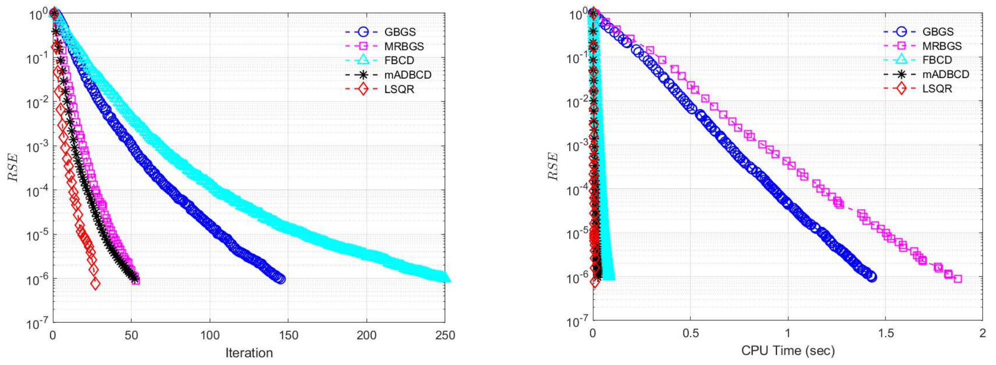

Fig. 4.16. RSE versus IT (left) and CPU (right) of GBGS, MRBGS, FBCD, LSQR and mADBCD for coefficient matrix cage10.

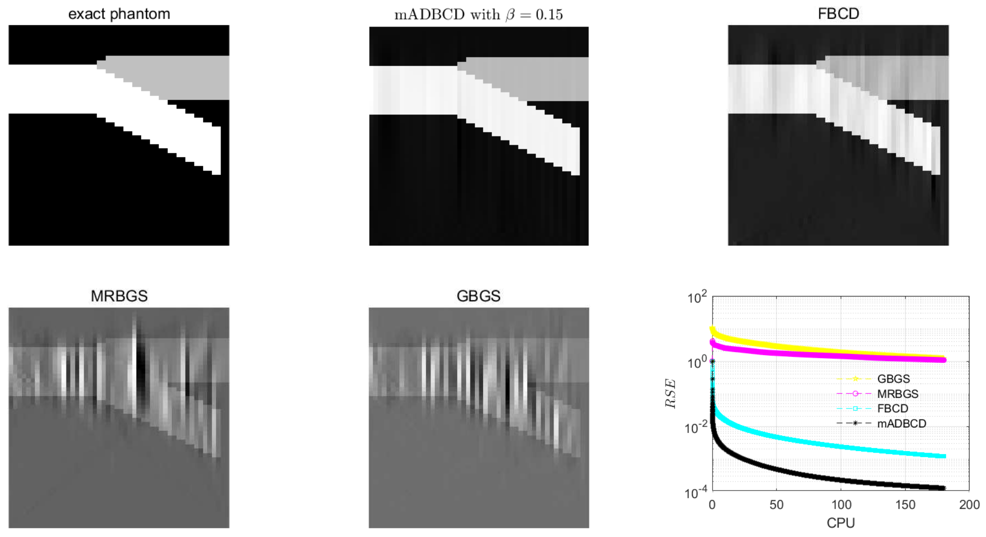

Fig. 4.17. Performance of GBGS, MRBGS, FBCD and mADBCD methods for `seismictomo`($n = 50$, $s = 80, p = 120$) test problem.

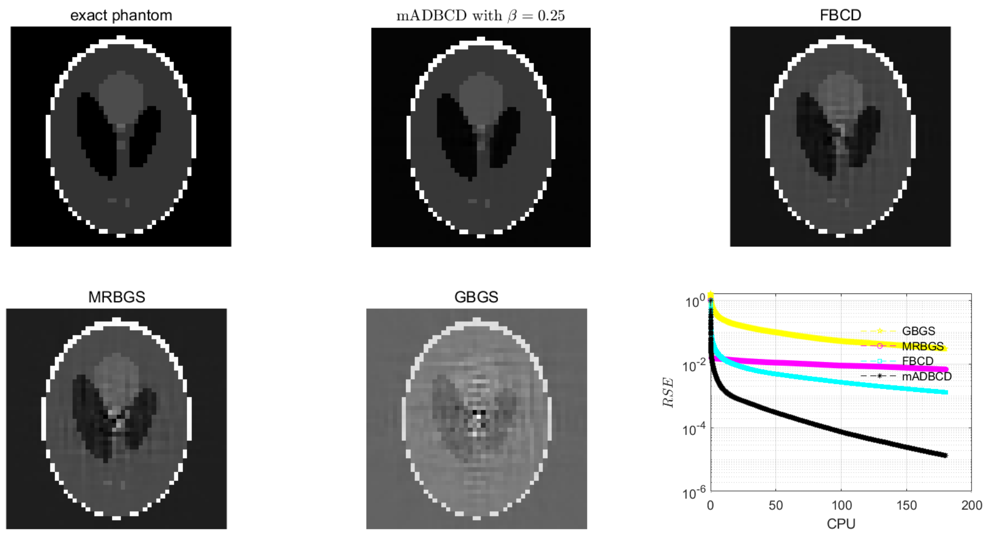

Fig. 4.18. Performance of GBGS, MRBGS, FBCD and mADBCD methods for `fanlineartomo`($N = 50$, $\theta = 0:1:300°, P = 50$) test problem.

respectively. As observed, mADBCD achieves the highest reconstruction quality within the same CPU time, followed by FBCD, while GBGS performs the worst.

5. Conclusion

In this paper, we propose an adaptive block coordinate descent method with momentum for solving linear least-squares problems. Similar to GBGS, MRBGS, and FBCD [4,14,15], the proposed mADBCD method updates all selected coordinates simultaneously in each iteration, in contrast to RCD, GRCD, and RGRCD [2,13,29], which update only a single coordinate at a time. A notable advantage of mADBCD lies in the simplicity of its iteration scheme: the update follows a momentum-augmented block coordinate descent strategy that involves only basic matrix-vector operations, resulting in a clear computational structure and ease of implementation. This feature clearly distinguishes mADBCD from the LSQR method, whose iteration scheme is designed based on a bidiagonalization process and involves operations such as subspace projections.

Under the full column-rank assumption, we establish the convergence of mADBCD to the unique solution, and show that the convergence rate depends on the minimum singular value of the full matrix, the maximum singular value of the selected submatrices, and the momentum parameter β. Numerical experiments demonstrate that mADBCD outperforms existing block methods in terms of efficiency. In particular, for highly sparse problems, the CPU time of mADBCD is nearly comparable to that of LSQR. This result suggests the potential of block coordinate descent methods, such as mADBCD, to surpass LSQR in solving sparse least-squares problems. Moreover, the experiments reveal that the performance of mADBCD is sensitive to the choice of the momentum parameter β. According to Theorem 3.1, the optimal selection of β depends intricately on the properties of the matrix and the sampling strategy, making it theoretically challenging to determine an optimal range. Therefore, developing adaptive strategies for selecting effective momentum parameters remains a promising direction for future research.

Acknowledgments. The authors would like to express their heartfelt gratitude to the anonymous referees for many constructive suggestions and comments that led to a great improvement of the paper.

X.P. Guo's research is partly supported by the National Key Research and Development Program (Grant No. 2022YFA1004403), by the National Natural Science Foundation of China (Grant No. 12471377), by the Science and Technology Commission of Shanghai Municipality (Grant No. 22DZ2229014), by the Opening Foundation of Shanxi Key Laboratory for Intelligent Optimization Computing and Blockchain Technology (Grant No. IOCBT2024Z01). M.Y. Deng's research is supported by the Anhui Provincial Natural Science Foundation (Grant No. 2508085J025).

Contributions. All authors contributed equally to this work.

References

[1] M. Arioli and I.S. Duff, Preconditioning linear least-squares problems by identifying a basis matrix, *SIAM J. Sci. Comput.*, **37**:5 (2015), S544–S561.

[2] Z.Z. Bai and W.T. Wu, On greedy randomized coordinate descent methods for solving large linear least-squares problems, *Numer. Linear Algebra Appl.*, **26**:4 (2019), e2237.

[3] Å. Björck, *Numerical Methods for Least Squares Problems*, SIAM, 2024.

[4] J.Q. Chen and Z.D. Huang, A fast block coordinate descent method for solving linear least-squares problems, *East Asian J. Appl. Math.*, **12** (2022), 406–420.

[5] T.A. Davis and Y. Hu, The university of Florida sparse matrix collection, *ACM Trans. Math. Softw.*, **38**:1 (2011), 1–25.

[6] K. Du, Tight upper bounds for the convergence of the randomized extended Kaczmarz and Gauss-Seidel algorithms, *Numer. Linear Algebra Appl.*, **26**:3 (2019), e2233.

[7] M. Elad, B. Matalon, and M. Zibulevsky, Coordinate and subspace optimization methods for linear least squares with non-quadratic regularization, *Appl. Comput. Harmon. Anal.*, **23**:3 (2007), 346–367.

[8] D. Han and J. Xie, On pseudoinverse-free randomized methods for linear systems: Unified framework and acceleration, *Optim. Methods Softw.*, (2025), 2581592.

[9] P.C. Hansen and J.S. Jørgensen, Air tools II: Algebraic iterative reconstruction methods, improved implementation, *Numer. Algorithms*, **79**:1 (2018), 107–137.

[10] A. Hefny, D. Needell, and A. Ramdas, Rows versus columns: Randomized Kaczmarz or Gauss-Seidel for ridge regression, *SIAM J. Sci. Comput.*, **39**:5 (2017), S528–S542.

[11] N.J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, 2002.

[12] A. Hoyos-Idrobo, P. Weiss, A. Massire, A. Amadon, and N. Boulant, On variant strategies to solve the magnitude least squares optimization problem in parallel transmission pulse design and under strict sar and power constraints, *IEEE Trans. Med. Imaging*, **33**:3 (2013), 739–748.

[13] D. Leventhal and A.S. Lewis, Randomized methods for linear constraints: Convergence rates and conditioning, *Math. Oper. Res.*, **35**:3 (2010), 641–654.

[14] H. Li and Y. Zhang, Greedy block Gauss-Seidel methods for solving large linear least squares problem, *J. Tongji Univ. Nat. Sci.*, **49**:11 (2021), 1514.

[15] Q. Lin, Z. Lu, and L. Xiao, An accelerated randomized proximal coordinate gradient method and its application to regularized empirical risk minimization, *SIAM J. Optim.*, **25**:4 (2015), 2244–2273.

[16] Y. Liu, X.L. Jiang, and C.Q. Gu, On maximum residual block and two-step Gauss-Seidel algorithms for linear least-squares problems, *Calcolo*, **58** (2021), 13.

[17] A. Ma, D. Needell, and A. Ramdas, Convergence properties of the randomized extended Gauss-Seidel and Kaczmarz methods, *SIAM J. Matrix Anal. Appl.*, **36**:4 (2015), 1590–1604.

[18] I. Necoara, Y. Nesterov, and F. Glineur, Random block coordinate descent methods for linearly constrained optimization over networks, *J. Optim. Theory Appl.*, **173** (2017), 227–254.

[19] D. Needell and J.A. Tropp, Paved with good intentions: Analysis of a randomized block Kaczmarz method, *Linear Algebra Appl.*, **441** (2014), 199–221.

[20] D. Needell, R. Zhao, and A. Zouzias, Randomized block Kaczmarz method with projection for solving least squares, *Linear Algebra Appl.*, 4**84** (2015), 322–343.

[21] Y. Nesterov, Efficiency of coordinate descent methods on huge-scale optimization problems, *SIAM J. Optim.*, **22**:2 (2012), 341–362.

[22] C.C. Paige and M.A. Saunders, LSQR: An algorithm for sparse linear equations and sparse least squares, *ACM Trans. Math. Softw.*, **8**:1 (1982), 43–71.

[23] B.T. Polyak, Some methods of speeding up the convergence of iteration methods, *Comput. Math. Math. Phys.*, **4**:5 (1964), 1–17.

[24] A. Ruhe, Numerical aspects of Gram-Schmidt orthogonalization of vectors, *Linear Algebra Appl.*, **52** (1983), 591–601.

[25] J.A. Scott and M. Tuuma, Sparse stretching for solving sparse-dense linear least-squares problems, *SIAM J. Sci. Comput.*, **41**:3 (2019), A1604–A1625.

[26] L.Z. Tan and X.P. Guo, On multi-step greedy randomized coordinate descent method for solving large linear least-squares problems, *Comput. Appl. Math.*, **42**:1 (2023), 37.

[27] C.L. Wang, D.S. Wu, and K.J. Yang, New decentralized positioning schemes for wireless sensor networks based on recursive least-squares optimization, *IEEE Wirel. Commun. Lett.*, **3**:1 (2013), 78–81.

[28] W. Wu, Convergence of the randomized block Gauss-Seidel method, *SIAM Undergrad. Res. Online*, **11** (2018), 369–382.

[29] J. Zhang and J. Guo, On relaxed greedy randomized coordinate descent methods for solving large linear least-squares problems, *Appl. Numer. Math.*, **157** (2020), 372–384.

[30] Y. Zhang and H. Li, A novel greedy gauss-seidel method for solving large linear least squares problem, *arXiv:2004.03692*, 2020.