East Asian Journal on Applied Mathematics
doi: 10.4208/eajam.2024-176.150325

Vol. **xx**, No. x, pp. 1-20
xxx 2025

A Greedy Randomized Average Block Projection Method for Linear Feasibility Problems

Lin Zhu[1], Yuan Lei[2] and Jiaxin Xie[3,*]

[1]*General Education Center, Changsha Social Work College, Changsha 410004, China.*
[2]*School of Mathematics, Hunan University, Changsha 410082, China.*
[3]*LMIB of the Ministry of Education, School of Mathematical Sciences, Beihang University, Beijing 100191, China.*

Received 16 October 2024; Accepted (in revised version) 15 March 2025.

Abstract. The randomized projection (RP) method is a simple iterative scheme for solving linear feasibility problems and has gained popularity due to its speed and low memory requirement. This paper develops an accelerated variant of the standard RP method by using two ingredients: the greedy probability criterion and the average block approach, and obtains a greedy randomized average block projection (GRABP) method for solving large-scale systems of linear inequalities. We demonstrate that the GRABP method achieves deterministic linear convergence with various extrapolated step sizes. Numerical experiments on both randomly generated and real-world data show the advantage of GRABP over several state-of-the-art solvers, such as the RP method, the sampling Kaczmarz Motzkin (SKM) method, the generalized SKM method, and the Nesterov acceleration of SKM method.

AMS subject classifications: 65K05, 90C15, 65F10

Key words: Linear feasibility, randomized projection, average block, greedy probability criterion, convergence property.

1. Introduction

1.1. Model and notation

We consider the problem of solving large-scale systems of linear inequalities

$$Ax \le b, \tag{1.1}$$

where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. We confine the scope of this work to the regime of $m \gg n$, where iterative methods are more competitive for such problems. We denote the feasible

*Corresponding author. *Email addresses:* `linzhu_cs@outlook.com` (L. Zhu), `yleimath@hnu.edu.cn` (Y. Lei), `xiejx@buaa.edu.cn` (J.X. Xie) region of (1.1) by $S = \{x \in \mathbb{R}^n \mid Ax \leq b\}$. Throughout this paper, we assume that the coefficient matrix A has no zero rows and $S \neq \emptyset$.

For a given matrix G, we use $\|G\|_2$, $\|G\|_F$, and $G^\dagger$ to denote the spectral norm, the Frobenius norm, and the Moore-Penrose pseudoinverse, respectively. We use $\sigma_{\min}(G)$ to denote the smallest nonzero singular value of the matrix G. For an integer $m \geq 1$, let $[m] := \{1, \ldots, m\}$. For any vector $x \in \mathbb{R}^n$, we use $x_i, x^\top$, $\|x\|_2$, and $\|x\|_p$ to denote the i-th entry, the transpose, the Euclidean norm and the p-norm of x, respectively. For any $u \in \mathbb{R}$ and $v \in \mathbb{R}^n$, we define $(u)_+ = \max\{0, u\}$ and $(v)_+ = ((v_1)_+, \ldots, (v_n)_+)^\top$. We refer to $\{\mathcal{I}_1, \mathcal{I}_2, \cdots, \mathcal{I}_t\}$ as a partition of $[m]$ if $\mathcal{I}_i \bigcap \mathcal{I}_j = \emptyset$ for $i \neq j$ and $\bigcup_{i=1}^t \mathcal{I}_i = [m]$. For a given index set $\mathcal{I}_i$, we use $G_{\mathcal{I}_i,:}$ to denote the row submatrix of the matrix G indexed by $\mathcal{I}_i$ and $u_{\mathcal{I}_i}$ denote the subvector of the vector u with components listed in $\mathcal{I}_i$. We use $P_S(u)$ to represent the orthogonal projection of u onto the feasible region S. For any random variables ξ, we use $\mathbb{E}[\xi]$ to denote the expectation of ξ.

1.2. The randomized Kaczmarz method

The Kaczmarz method [17], also known as the algebraic reconstruction technique (ART) [8,12], is a widely used algorithm for solving linear systems $Ax = b$. Starting from $x^0 \in \mathbb{R}^n$, the canonical Kaczmarz method constructs x^{k+1} by

$$x^{k+1} = x^k - \frac{A_{i,:}x^k - b_i}{\|A_{i,:}\|_2^2} A_{i,:}^\top,$$

where i is selected from $[m]$ cyclically. In fact, the current iterate is projected orthogonally onto the selected hyperplane $\{x \mid A_{i,:}x = b_i\}$ at each iteration. The iteration sequence $\{x^k\}_{k \geq 0}$ converges to $x_*^0 := A^\dagger b + (I - A^\dagger A)x^0$. However, the rate of convergence is hard to obtain. In the seminal paper [31], Strohmer and Vershynin analyzed the randomized variant of the Kaczmarz method. In particular, they proved that if the i-th row of A is selected with probability proportional to $\|A_{i,:}\|_2^2$, then the method converges linearly in expectation.

Leventhal and Lewis [19] extended the randomized Kaczmarz (RK) method to solve the linear feasibility problem (1.1). At each iteration k, if the inequality is already satisfied for the selected row i, then set $x_{k+1} = x_k$. If the inequality is not satisfied, the previous iterate only projects onto the solution hyperplane $\{x \mid A_{i,:}x = b_i\}$. The update rule for this algorithm is thus

$$x^{k+1} = x^k - \frac{(A_{i,:}x^k - b_i)_+}{\|A_{i,:}\|_2^2} (A_{i,:})^\top. \tag{1.2}$$

One can see that x^{k+1} in (1.2) is indeed the projection of x^k onto the set $\{x \mid A_{i,:}x \leq b_i\}$. Leventhal and Lewis proved that such RP method converges to a feasibility solution linearly in expectation — cf. [19, Theorem 4.3].

Combining the ideas of Kaczmarz and Motzkin methods [1,24], Loera *et al.* [6] recently proposed an SKM method for solving the linear feasibility problem (1.1). Later, Morshed *et al.* [23] developed a generalized framework, namely the generalized SKM (GSKM) method that extends the SKM algorithm and proves the existence of a family of SKM-type methods. In addition, they also proposed a Nesterov-type acceleration scheme in the SKM method called probably accelerated SKM (PASKM), which provides a bridge between Nesterov-type acceleration of machine learning to sampling Kaczmarz methods for solving linear feasibility problems.

1.3. The greedy probability criterion

The greedy probability criterion was originally proposed by Bai and Wu [2] for effectively selecting the working row from the matrix A, and a greedy RK (GRK) method which is faster than the RK method in terms of the number of iterations and computing time is introduced. Indeed, at the k-th iteration, GRK determines a subset $\mathscr{U}_k$ of $[m]$ such that the magnitude of the residual $A_{i,:}x^k - b_i$ exceeds a threshold — i.e.

$$\mathscr{U}_k = \left\{ i_k \;\middle|\; |A_{i_k,:}x^k - b_{i_k}|^2 \geq \varepsilon_k \|Ax^k - b\|_2^2 \|A_{i_k,:}\|_2^2 \right\},$$

where

$$\varepsilon_k = \frac{1}{2}\left(\frac{1}{\|Ax^k - b\|_2^2} \max_{1\leq i\leq m} \left\{ \frac{|A_{i,:}x^k - b_i|^2}{\|A_{i,:}\|_2^2} \right\} + \frac{1}{\|A\|_F^2} \right).$$

A modified residual vector $\tilde{r}^k$ is defined by

$$\tilde{r}_i^k = \begin{cases} A_{i,:}x^k - b_i, & \text{if} \quad i \in \mathscr{U}_k, \\ 0, & \text{otherwise.} \end{cases}$$

GRK selects the index $i_k \in \mathscr{U}_k$ of the working row with probability

$$\Pr(i_k) = \frac{|\tilde{r}_{i_k}^k|^2}{\|\tilde{r}^k\|_2^2}.$$

Finally, GRK orthogonally projects the current iterate x^k onto the i_k-th hyperplane $\{x \mid A_{i_k,:}x = b_{i_k}\}$ to obtain the next iterate x^{k+1}. By using the above greedy idea, small entries of the residual vector $Ax^k - b$ may not be selected, which guarantees the progress of each iteration of GRK and a faster convergence rate of GRK may be expected than that of RK. The idea of greedy has been used in many works, see for example the literatures [4, 9, 22, 32, 33, 36] and the references therein.

1.4. The block Kaczmarz method

The block Kaczmarz method first partitions the rows $[m]$ into t blocks, denoted $\mathscr{I}_1, \cdots, \mathscr{I}_t$. Instead of selecting one row per-iteration as done with the simple Kaczmarz method, the block Kaczmarz algorithm chooses a block uniformly at random at each iteration. Needell and Tropp [27] proposed a randomized block Kaczmarz (RBK), where at each iteration, the previous iterate x_k is projected onto the solution space to $A_{\mathscr{I}_{i_k},:}x = b_{\mathscr{I}_{i_k}}$.

However, each iterate of this RBK method needs to apply the pseudoinverse of the chosen submatrix to a vector and it is expensive.

Recently, Necoara [25] developed a randomized average block Kaczmarz (RABK) algorithm for linear systems which takes a convex combination of several RK updates as a new direction with some stepsize. Assuming that the subset $\mathscr{I}_{i_k}$ has been selected at the k-th iteration, RABK generates the k-th estimate x^{k+1} via

$$x^{k+1}=x^k-\alpha_k\left(\sum_{j\in\mathscr{I}_{i_k}}\omega_j^k\frac{A_{j,:}x^k-b_j}{\|A_{j,:}\|_2^2}(A_{j,:})^\top\right), \tag{1.3}$$

where the weights $\omega_j^k\in[0,1]$ such that $\sum_{j\in\mathscr{I}_{i_k}}\omega_j^k=1$ and the stepsize $\alpha_k\in(0,2)$. The convergence analysis reveals that RABK is extremely effective when it is given a good sampling of the rows into well-conditioned blocks. Specifically, if $\omega_j^k=\|A_{j,:}\|_2^2/\|A_{\mathscr{I}_{i_k},:}\|_F^2$ with $j\in\mathscr{I}_{i_k}$, then (1.3) becomes

$$x^{k+1}=x^k-\alpha_k\frac{(A_{\mathscr{I}_{i_k},:})^\top(A_{\mathscr{I}_{i_k},:}x^k-b_{\mathscr{I}_{i_k}})}{\|A_{\mathscr{I}_{i_k},:}\|_F^2}. \tag{1.4}$$

Recently, the iteration scheme (1.4) has been used in many works. In [20], Miao and Wu proposed a greedy RABK method for solving linear systems. Necoara [26] used the idea of blocks of sets to develop accelerated RP methods for convex feasibility problems. For another block version of the RK method, we refer to the literatures [7,10,11,21,37] and the references therein.

1.5. Our contribution

This paper extends the ideas of the greedy probability criterion and the average block approach [2,20] to solve linear feasibility problems, obtaining a GRABP method. Recall that $\{\mathscr{I}_1,\mathscr{I}_2,\cdots,\mathscr{I}_t\}$ is a partition of the row index set $[m]$ of the matrix A. At each step, we greedily choose a nonempty index set $\mathscr{U}_k$ using an adaptive thresholding rule so that for any $i\in\mathscr{U}_k$, the norm of the residual $(A_{\mathscr{I}_i,:}x^k-b_{\mathscr{I}_i})_+$ should be larger than a prescribed threshold. After selecting an index $i_k\in\mathscr{U}_k$ with a certain probability criterion, we project the current iteration vector onto each feasible region $\{x \mid A_{i_k,:}x\le b_{i_k}\}$ with $i_k\in\mathscr{U}_k$, average them, and apply extrapolated step sizes to construct the GRABP method. Relying on a lemma due to Hoffman [14,19], two kinds of extrapolated stepsizes for the GRABP method are analyzed. Furthermore, linear convergence of the proposed GRABP is based on the quantity $\|x^k-P_S(x^k)\|_2^2$, rather than its expectation $\mathbb{E}[\|x^k-P_S(x^k)\|_2^2]$, which is commonly used in the literature [2,20]. We refer such convergence as deterministic. The numerical results show the advantage of the GRABP method over several state-of-the-art solvers, such as the RP method, the SKM method, the GSKM method, and the PASKM method.

The organization of this paper is as follows. We introduce a GRABP method for solving the linear feasibility problems in Section 2 and analyse the convergence of the method in

Section 3. Numerical experimental results are presented in Section 4. Finally, we end this paper with concluding remarks in Section 5.

2. A Greedy Randomized Average Block Projection Method

In this section, we introduce a GRABP method for solving the linear feasibility problem (1.1). The method is formally described in Algorithm 2.1.

Algorithm 2.1 The GRABP Method

```
Require: $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $k = 0$, and an initial $x^0 \in \mathbb{R}^n$. Let $\{\mathscr{I}_1, \mathscr{I}_2, \cdots, \mathscr{I}_t\}$ be a partition of $[m]$.
1: Compute
     $$\epsilon_k = \frac{1}{2}\left(\frac{1}{\|(Ax^k - b)_+\|_2^2} \max_{1\le i\le t} \frac{\|(A_{\mathscr{I}_i,:}x^k - b_{\mathscr{I}_i})_+\|_2^2}{\|A_{\mathscr{I}_i,:}\|_F^2} + \frac{1}{\|A\|_F^2}\right).$$
2: Determine the index set
     $$\mathscr{U}_k = \left\{ i \mid \|(A_{\mathscr{I}_i,:}x^k - b_{\mathscr{I}_i})_+\|_2^2 \ge \epsilon_k \|(Ax^k - b)_+\|_2^2 \|A_{\mathscr{I}_i,:}\|_F^2 \right\}. \qquad (2.1)$$
3: Select $i_k \in \mathscr{U}_k$ according to the probability
     $$\Pr(i) = p_{k,i}, \quad i = 1, 2, \dots, t$$
   with $p_{k,i} = 0$ if $i \notin \mathscr{U}_k$, $p_{k,i} \ge 0$ if $i \in \mathscr{U}_k$, and $\sum_{i\in\mathscr{U}_k} p_{k,i} = 1$.
4: Update
     $$x^{k+1} = x^k - \alpha_k \frac{(A_{\mathscr{I}_{i_k},:})^\top (A_{\mathscr{I}_{i_k},:}x^k - b_{\mathscr{I}_{i_k}})_+}{\|A_{\mathscr{I}_{i_k},:}\|_F^2}.$$
5: If the stopping rule is satisfied, stop and go to output. Otherwise, set $k = k + 1$ and return to step 1.
Ensure: The approximate solution $x^k$.
```

Remark 2.1. Similar to [3, 38], we can introduce an arbitrary relaxation parameter $\theta \in [0, 1]$ into ϵ_k in Algorithm 2.1, i.e.

$$\epsilon_k = \frac{\theta}{\|(Ax^k - b)_+\|_2^2} \max_{1\le i\le t} \frac{\|(A_{\mathscr{I}_i,:}x^k - b_{\mathscr{I}_i})_+\|_2^2}{\|A_{\mathscr{I}_i,:}\|_F^2} + (1-\theta)\frac{1}{\|A\|_F^2}.$$

In this case, setting $\theta = 1/2$, we can obtain the GRABP method. In fact, the parameter θ affects the index set $\mathscr{U}_k$ to some extent, but the algorithm maintains linear convergence regardless of the value chosen for the parameter θ. This paper focuses on the case where $\theta = 1/2$.

We next present some specific details of Algorithm 2.1.

Randomized row partition. In the setup of Algorithm 2.1, we need to partition the row index set of $[m]$ of the coefficient matrix A into $\{\mathscr{I}_1, \mathscr{I}_2, \cdots, \mathscr{I}_t\}$. The row partition of the matrix has been extensively discussed in the literatures [25, 26, 34, 35]. In this paper, we use a simple partition sampling — i.e.

$$\begin{aligned}
&\mathscr{I}_i = \big\{\varpi(k) : k = (i-1)\beta + 1, (i-1)\beta + 2, \dots, i\beta\big\}, \quad i = 1, 2, \dots, t-1,\\
&\mathscr{I}_t = \big\{\varpi(k) : k = (t-1)\beta + 1, (t-1)\beta + 2, \dots, m\big\}, \quad |\mathscr{I}_t| \le \beta,
\end{aligned} \tag{2.2}$$

where ϖ is a uniform random permutation on $[m]$ and β is the block size.

Probability strategy. There are many choices for the probability strategy in step 3 of Algorithm 2.1. In this work, we adopt the approach proposed in [20]. Let $p > 0$, $\mu > 0$ and for $k = 0, 1, 2, \dots,$

$$\tilde{r}^k_{\mathscr{I}_i} = \begin{cases} (A_{\mathscr{I}_i,:}x^k - b_{\mathscr{I}_i})_+, & \text{if} \quad i \in \mathscr{U}_k,\\ 0, & \text{otherwise.} \end{cases}$$

Then one can use the following two different probability strategies:

$$p_{k,i} = \begin{cases} \dfrac{\|\tilde{r}^k_{\mathscr{I}_i}\|_p^p}{\sum_{i\in\mathscr{U}_k} \|\tilde{r}^k_{\mathscr{I}_i}\|_p^p}, & \text{if} \quad i \in \mathscr{U}_k,\\ 0, & \text{otherwise,} \end{cases} \tag{2.3}$$

and

$$p_{k,i} = \begin{cases} \dfrac{\|\tilde{r}^k_{\mathscr{I}_i}\|_2^\mu}{\sum_{i\in\mathscr{U}_k} \|\tilde{r}^k_{\mathscr{I}_i}\|_2^\mu}, & \text{if} \quad i \in \mathscr{U}_k,\\ 0, & \text{otherwise.} \end{cases} \tag{2.4}$$

Obviously, both (2.3) and (2.4) satisfy $p_{k,i} = 0$ if $i \notin \mathscr{U}_k$, $p_{k,i} \ge 0$ if $i \in \mathscr{U}_k$, and $\sum_{i\in\mathscr{U}_k} p_{k,i}$ $= 1$. When $p = u = 2$, the above two probability strategies are the same.

Stepsize choice. It is well-known that the stepsize affects the convergence of the algorithm. We consider two choices for the stepsize α_k as in [20]. Let $w \in (0, 2)$. One is a constant stepsize

$$\alpha_k = \frac{w}{\zeta} \quad \text{with} \quad \zeta := \max_{1\le i\le t} \frac{\sigma^2_{\max}(A_{\mathscr{I}_i,:})}{\|A_{\mathscr{I}_i,:}\|_F^2}. \tag{2.5}$$

Another is adaptive stepsize

$$\alpha_k = w\frac{\|(A_{\mathscr{I}_{i_k},:}x^k - b_{\mathscr{I}_{i_k}})_+\|_2^2 \|A_{\mathscr{I}_{i_k},:}\|_F^2}{\|(A_{\mathscr{I}_{i_k},:})^\top (A_{\mathscr{I}_{i_k},:}x^k - b_{\mathscr{I}_{i_k}})_+\|_2^2}. \tag{2.6}$$

Since at the k-th iterate of the GRABP method, we always have

$$\left\|\left(A_{\mathscr{I}_{i_k},:}x^k - b_{\mathscr{I}_{i_k}}\right)_+\right\|_2^2 \ne 0,$$

and, consequently,

$$\left\|\left(A_{\mathscr{I}_{i_k},:}\right)^\top\left(A_{\mathscr{I}_{i_k},:}x^k-b_{\mathscr{I}_{i_k}}\right)_+\right\|_2^2\neq 0. \tag{2.7}$$

Indeed, if (2.7) does not hold — i.e.

$$\left(A_{\mathscr{I}_{i_k},:}\right)^\top\left(A_{\mathscr{I}_{i_k},:}x^k-b_{\mathscr{I}_{i_k}}\right)_+=0,$$

then

$$\left\langle A_{\mathscr{I}_{i_k},:}(x^k-P_S(x^k)),\left(A_{\mathscr{I}_{i_k},:}x^k-b_{\mathscr{I}_{i_k}}\right)_+\right\rangle=0.$$

Noting that $P_S(x^k)\in S$, we have $A_{\mathscr{I}_{i_k},:}P_S(x^k)\leq b_{\mathscr{I}_{i_k}}$. So

$$\begin{aligned}0&=\left\langle A_{\mathscr{I}_{i_k},:}(x^k-P_S(x^k)),\left(A_{\mathscr{I}_{i_k},:}x^k-b_{\mathscr{I}_{i_k}}\right)_+\right\rangle\\&\geq\left\langle A_{\mathscr{I}_{i_k},:}x^k-b_{\mathscr{I}_{i_k}},\left(A_{\mathscr{I}_{i_k},:}x^k-b_{\mathscr{I}_{i_k}}\right)_+\right\rangle\\&=\left\|\left(A_{\mathscr{I}_{i_k},:}x^k-b_{\mathscr{I}_{i_k}}\right)_+\right\|_2^2,\end{aligned}$$

which contradicts the relation $\|(A_{\mathscr{I}_{i_k},:}x^k-b_{\mathscr{I}_{i_k}})_+\|_2^2\neq 0$. So the adaptive step size in (2.6) is well defined. For convenience, we refer to the GRABP method with constant stepsize as GRABP-c. The GRABP method with adaptive stepsize, we refer to as GRABP-a. In Section 3, we will discuss the convergence properties of the GRABP-c and GRABP-a methods, respectively.

Finally, let us briefly state that the index set $\mathscr{U}_k$ in (2.1) is well defined — i.e. the index set $\mathscr{U}_k$ is nonempty. In fact, for any $k\geq 0$,

$$\begin{aligned}&\max_{1\leq i\leq t}\frac{\|(A_{\mathscr{I}_i,:}x^k-b_{\mathscr{I}_i})_+\|_2^2}{\|A_{\mathscr{I}_i,:}\|_F^2}\\&\geq\sum_{i=1}^{t}\frac{\|A_{\mathscr{I}_i,:}\|_F^2}{\|A\|_F^2}\frac{\|(A_{\mathscr{I}_i,:}x^k-b_{\mathscr{I}_i})_+\|_2^2}{\|A_{\mathscr{I}_i,:}\|_F^2}\\&=\frac{\|(Ax^k-b)_+\|_2^2}{\|A\|_F^2},\end{aligned}\tag{2.8}$$

which implies

$$\begin{aligned}&\max_{1\leq i\leq t}\frac{\|(A_{\mathscr{I}_i,:}x^k-b_{\mathscr{I}_i})_+\|_2^2}{\|A_{\mathscr{I}_i,:}\|_F^2}\\&\geq\frac{1}{2}\max_{1\leq i\leq t}\frac{\|(A_{\mathscr{I}_i,:}x^k-b_{\mathscr{I}_i})_+\|_2^2}{\|A_{\mathscr{I}_i,:}\|_F^2}+\frac{1}{2}\frac{\|(Ax^k-b)_+\|_2^2}{\|A\|_F^2}\\&=\epsilon_k\|(Ax^k-b)_+\|_2^2.\end{aligned}$$

Hence, the index $I_k^{\max}$ always belongs to $\mathscr{U}_k$ and the index set $\mathscr{U}_k$ is always nonempty, where $I_k^{\max}\in\arg\max_{1\leq i\leq t}\|(A_{\mathscr{I}_i,:}x^k-b_{\mathscr{I}_i})_+\|_2^2/\|A_{\mathscr{I}_i,:}\|_F^2$.

3. Deterministic Linear Convergence

In this section, we discuss the convergence of Algorithm 2.1. Let us first introduce a crucial lemma.

Lemma 3.1 (Hoffman [14]). *Let $x \in \mathbb{R}^n$ and S be the feasible region of the linear feasibility problem* (1.1). *Then, there exists a constant $L > 0$ such that*

$$\|x - P_S(x)\|_2^2 \le L^2 \|(Ax - b)_+\|_2^2,$$

where $P_S(x)$ is the orthogonal projection of x onto the feasible region S.

Lemma 3.1 is a well-known result of Hoffman on systems of linear inequalities. The constant L is called the Hoffman constant. Recently, Pena *et al.* [29, Proposition 2] provided the following characterization of L:

$$L = \max_{J \in \mathscr{S}(A)} \frac{1}{\min_{v \in \mathbb{R}_+^{|J|}, \|v\|_2 = 1} \|A_{J,:}^\top v\|_2},$$

where $\mathscr{S}(A)$ is the collection of subsets $J \subseteq [m]$ such that $A_{J,:}x < 0$ is feasible. For a detailed discussion on the Hoffman constants for systems of linear constraints, we refer readers to [29]. For the GRABP method employing either a constant stepsize (2.5) or an adaptive stepsize (2.6), we establish the following convergence result.

Theorem 3.1. *Suppose that the linear feasibility problem* (1.1) *is consistent — i.e. the feasible region S is nonempty and let $w \in (0,2)$. Let $\{x^k\}_{k \ge 0}$ be the sequence generated by Algorithm* 2.1 *using either a constant stepsize* (2.5) *or an adaptive stepsize* (2.6). *Then we have*

$$\left\|x^k - P_S(x^k)\right\|_2^2 \le \left(1 - \frac{2w - w^2}{\zeta L^2 \|A\|_F^2}\right)^k \left\|x^0 - P_S(x^0)\right\|_2^2,$$

where $P_S(x^k)$ is the orthogonal projection of x^k onto the feasible region S.

Proof. Straightforward calculations yield

$$\begin{aligned} \left\|x^{k+1} - P_S(x^{k+1})\right\|_2^2 &\le \left\|x^{k+1} - P_S(x^k)\right\|_2^2 \\ &= \left\| x^k - \alpha \frac{(A_{\mathscr{I}_{i_k},:})^\top (A_{\mathscr{I}_{i_k},:}x^k - b_{\mathscr{I}_{i_k}})_+}{\|A_{\mathscr{I}_{i_k},:}\|_F^2} - P_S(x^k) \right\|_2^2 \\ &= \left\|x^k - P_S(x^k)\right\|_2^2 + \alpha^2 \frac{\|(A_{\mathscr{I}_{i_k},:})^\top (A_{\mathscr{I}_{i_k},:}x^k - b_{\mathscr{I}_{i_k}})_+\|_2^2}{\|A_{\mathscr{I}_{i_k},:}\|_F^4} \\ &\quad - \frac{2\alpha}{\|A_{\mathscr{I}_{i_k},:}\|_F^2} \left\langle \left(A_{\mathscr{I}_{i_k},:}\right)^\top \left(A_{\mathscr{I}_{i_k},:}x^k - b_{\mathscr{I}_{i_k}}\right)_+, x^k - P_S(x^k) \right\rangle. \end{aligned} \tag{3.1}$$

Since $P_S(x^k) \in S$, we have $A_{\mathscr{I}_{i_k},:}P_S(x^k) \leq b_{\mathscr{I}_{i_k}}$. Therefore,

$$\begin{aligned}
&\left\langle \left(A_{\mathscr{I}_{i_k},:}\right)^\top \left(A_{\mathscr{I}_{i_k},:}x^k - b_{\mathscr{I}_{i_k}}\right)_+, x^k - P_S(x^k)\right\rangle \\
&= \left\langle \left(A_{\mathscr{I}_{i_k},:}x^k - b_{\mathscr{I}_{i_k}}\right)_+, A_{\mathscr{I}_{i_k},:}\left(x^k - P_S(x^k)\right)\right\rangle \\
&\geq \left\langle \left(A_{\mathscr{I}_{i_k},:}x^k - b_{\mathscr{I}_{i_k}}\right)_+, A_{\mathscr{I}_{i_k},:}x^k - b_{\mathscr{I}_{i_k}}\right\rangle \\
&= \left\|\left(A_{\mathscr{I}_{i_k},:}x^k - b_{\mathscr{I}_{i_k}}\right)_+\right\|_2^2.
\end{aligned}$$

Then,

$$\begin{aligned}
\left\|x^{k+1} - P_S(x^{k+1})\right\|_2^2 \leq \left\|x^k - P_S(x^k)\right\|_2^2 + \alpha_k^2 \frac{\|(A_{\mathscr{I}_{i_k},:})^\top (A_{\mathscr{I}_{i_k},:}x^k - b_{\mathscr{I}_{i_k}})_+\|_2^2}{\|A_{\mathscr{I}_{i_k},:}\|_F^4} \\
- \frac{2\alpha_k \|(A_{\mathscr{I}_{i_k},:}x^k - b_{\mathscr{I}_{i_k}})_+\|_2^2}{\|A_{\mathscr{I}_{i_k},:}\|_F^2}. \qquad (3.2)
\end{aligned}$$

First, we consider the case where the Algorithm 2.1 employs the constant stepsize (2.5). It follows from (2.5) that

$$\begin{aligned}
&\frac{\|(A_{\mathscr{I}_{i_k},:})^\top (A_{\mathscr{I}_{i_k},:}x^k - b_{\mathscr{I}_{i_k}})_+\|_2^2}{\|A_{\mathscr{I}_{i_k},:}\|_F^4} \\
&\leq \frac{\sigma_{\max}^2(A_{\mathscr{I}_{i_k},:})}{\|A_{\mathscr{I}_{i_k},:}\|_F^2} \frac{\|(A_{\mathscr{I}_{i_k},:}x^k - b_{\mathscr{I}_{i_k}})_+\|_2^2}{\|A_{\mathscr{I}_{i_k},:}\|_F^2} \\
&\leq \zeta \frac{\|(A_{\mathscr{I}_{i_k},:}x^k - b_{\mathscr{I}_{i_k}})_+\|_2^2}{\|A_{\mathscr{I}_{i_k},:}\|_F^2}. \qquad (3.3)
\end{aligned}$$

Substituting (3.3) into (3.2), we obtain

$$\left\|x^{k+1} - P_S(x^{k+1})\right\|_2^2 \leq \left\|x^k - P_S(x^k)\right\|_2^2 - \left(2\alpha_k - \alpha_k^2\zeta\right) \frac{\|(A_{\mathscr{I}_{i_k},:}x^k - b_{\mathscr{I}_{i_k}})_+\|_2^2}{\|A_{\mathscr{I}_{i_k},:}\|_F^2}.$$

According to (2.8) and the definition of $\mathscr{U}_k$, for any $\mathscr{I}_{i_k} \in \mathscr{U}_k$, we have

$$\begin{aligned}
\frac{\|(A_{\mathscr{I}_{i_k},:}x^k - b_{\mathscr{I}_{i_k}})_+\|_2^2}{\|A_{\mathscr{I}_{i_k},:}\|_F^2} &\geq \frac{1}{2} \max_{1 \leq i \leq t} \frac{\|(A_{\mathscr{I}_i,:}x^k - b_{\mathscr{I}_i})_+\|_2^2}{\|A_{\mathscr{I}_i,:}\|_F^2} + \frac{1}{2} \frac{\|(Ax^k - b)_+\|_2^2}{\|A\|_F^2} \\
&\geq \frac{\|(Ax^k - b)_+\|_2^2}{\|A\|_F^2}. \qquad (3.4)
\end{aligned}$$

Hence, we can get

$$\begin{aligned}
\left\|x^{k+1}-P_S(x^{k+1})\right\|_2^2 &\le \left\|x^k-P_S(x^k)\right\|_2^2-\left(2\alpha_k-\alpha_k^2\zeta\right)\frac{\|(Ax^k-b)_+\|_2^2}{\|A\|_F^2}\\
&\le\left(1-\frac{2\alpha_k-\alpha_k^2\zeta}{L^2\|A\|_F^2}\right)\left\|x^k-P_S(x^k)\right\|_2^2\\
&=\left(1-\frac{2w-w^2}{\zeta L^2\|A\|_F^2}\right)^k\left\|x^0-P_S(x^0)\right\|_2^2,
\end{aligned}$$

where the second inequality follows from Lemma 3.1 and the last equality follows from the definition of the constant stepsize α_k in (2.5). By induction on the iteration index k, we can obtain the desired result.

Next, we consider the case where Algorithm 2.1 employs an adaptive stepsize (2.6). Substituting such stepsize α_k into (3.1) gives

$$\begin{aligned}
\left\|x^{k+1}-P_S(x^{k+1})\right\|_2^2 \le \left\|x^k-P_S(x^k)\right\|_2^2-(2w-w^2)&\frac{\|(A_{\mathcal{I}_{i_k},:}x^k-b_{\mathcal{I}_{i_k}})_+\|_2^2\|A_{\mathcal{I}_{i_k},:}\|_F^2}{\|(A_{\mathcal{I}_{i_k},:})^\top(A_{\mathcal{I}_{i_k},:}x^k-b_{\mathcal{I}_{i_k}})_+\|_2^2}\\
&\times\frac{\|(A_{\mathcal{I}_{i_k},:}x^k-b_{\mathcal{I}_{i_k}})_+\|_2^2}{\|A_{\mathcal{I}_{i_k},:}\|_F^2}.
\end{aligned}$$

Since $2w-w^2>0$ and

$$\left\|\left(A_{\mathcal{I}_{i_k},:}\right)^\top\left(A_{\mathcal{I}_{i_k},:}x^k-b_{\mathcal{I}_{i_k}}\right)_+\right\|_2^2\le\sigma_{\max}^2\left(A_{\mathcal{I}_{i_k},:}\right)\left\|\left(A_{\mathcal{I}_{i_k},:}x^k-b_{\mathcal{I}_{i_k}}\right)_+\right\|_2^2,$$

we arrive at the estimate

$$\begin{aligned}
\left\|x^{k+1}-P_S(x^{k+1})\right\|_2^2 &\le \left\|x^k-P_S(x^k)\right\|_2^2-\frac{2w-w^2}{\zeta}\frac{\|(A_{\mathcal{I}_{i_k},:}x^k-b_{\mathcal{I}_{i_k}})_+\|_2^2}{\|A_{\mathcal{I}_{i_k},:}\|_F^2}\\
&\le\left(1-\frac{2w-w^2}{\zeta L^2\|A\|_F^2}\right)\left\|x^k-P_S(x^k)\right\|_2^2,
\end{aligned}$$

where the last inequality follows from (3.4) and Lemma 3.1. □

Remark 3.1. It can be seen from Theorem 3.1 that the convergence factor of the GRABP method is $1-(2w-w^2)/(\zeta L^2\|A\|_F^2)$. It attains the minimum value $1-1/(\zeta L^2\|A\|_F^2)$ when $w=1$. However, in our numerical experiments, we observed that $w>1$ often yield better performance. This observation is consistent with the phenomenon that the performance of optimization algorithms can be practically accelerated under over-relaxation conditions [10,25].

Remark 3.2. It may look confusing that despite being randomized, our algorithm exhibits deterministic linear convergence. This contrasts to much of the literature on randomized iterative methods [6, 19, 23], which typically considers the linear convergence of the expected norm of the error $\mathbb{E}[\|x^k - P_S(x^k)\|_2^2]$. This is because the inequality in (3.4) always holds regardless of the probability employed. In fact, deterministic linear convergence of $\|x^k - P_S(x^k)\|_2^2$ can lead to a lower iteration complexity compared to the linear convergence of $\mathbb{E}[\|x^k - P_S(x^k)\|_2^2]$. One may check [32, Section 2.2] for more discussions.

4. Experimental Results

In this section, we study the computational behavior of the GRABP algorithm (Algorithms GRABP-c and GRABP-a). We also compare our algorithms with some of the state-of-the-art methods, namely, the RP method, the SKM method, the GSKM method, and the PASKM method. All experiments are carried out using MATLAB on a laptop computer (AMD Ryzen 5 3500U @2.10 GHz).

4.1. Numerical setup

To evaluate computational performance, we conduct numerical experiments on a diverse set of instances, including both randomly generated and real-world test problems. The real-world instances are sourced from the SuiteSparse Matrix Collection [5] and the sparse Netlib LP instances [28]. For randomly generated coefficient matrix A, we consider two types: dense and sparse matrices. The dense matrix is generated by the MATLAB function `randn`. The sparse matrix is generated by the MATLAB function `sprandn`, with a density of $1/2\log(mn)$ for the non-zero elements. To ensure that the system (1.1) is consistent — i.e. $S \neq \emptyset$, we randomly generate vectors $x_1 \in \mathbb{R}^n$, $x_2 \in \mathbb{R}^n$, $x_3 \in \mathbb{R}^m$ and set the right-hand side as $b = 0.5Ax_1 + 0.5Ax_2 + x_3$. Both x_1 and x_2 are generated randomly by the MATLAB function `randn`. The vector x_3 is a randomly generated vector with elements in the range $[0.1, 1]$.

We initialize with $x^0 = 0 \in \mathbb{R}^n$. The experiments conclude when the relative residual error (RRE) satisfies

$$\text{RRE} := \frac{\|(Ax^k - b)_+\|_2}{\|b\|_2} < 10^{-6}.$$

We use the indices $\mathscr{I}_i$ from the random partition $\{\mathscr{I}_1, \mathscr{I}_2, \cdots, \mathscr{I}_t\}$ as defined in (2.2). The probability criterion $p_{k,i}$ in step 4 of the GRABP method is chosen as (2.3) with $p = 2$. The SKM, GSKM, and PASKM algorithms involve selecting various parameters. We have chosen a set of parameters with improved performance based on the literature [23]. Specifically, the parameters for the PASKM algorithm are selected according to the PASKM-2 algorithm (see [23] for details). All the results are averaged over 10 trials.

Note that the most computationally expensive step in the k-th iteration of Algorithm 2.1 is the computation of the residual $r^k = Ax^k - b$. Next, we demonstrate that if the matrix $B := AA^\top$ can be precomputed and stored during initialization, Algorithm 2.1 can achieve faster practical performance. Recall that Algorithm 2.1 updates x^{k+1} as

$$x^{k+1} = \frac{x^k - \alpha_k (A_{\mathscr{I}_{i_k},:})^\top (A_{\mathscr{I}_{i_k},:} x^k - b_{\mathscr{I}_{i_k}})_+}{\|A_{\mathscr{I}_{i_k},:}\|_F^2}.$$

Thus, the residual r^{k+1} can be expressed as

$$\begin{aligned} r^{k+1} &= Ax^{k+1} - b \\ &= Ax^k - b - \alpha_k \frac{A(A_{\mathscr{I}_{i_k},:})^\top (A_{\mathscr{I}_{i_k},:} x^k - b_{\mathscr{I}_{i_k}})_+}{\|A_{\mathscr{I}_{i_k},:}\|_F^2} \\ &= r^k - \alpha_k \frac{A(A_{\mathscr{I}_{i_k},:})^\top (A_{\mathscr{I}_{i_k},:} x^k - b_{\mathscr{I}_{i_k}})_+}{\|A_{\mathscr{I}_{i_k},:}\|_F^2}. \end{aligned}$$

Since $A(A_{\mathscr{I}_{i_k},:})^\top = B_{:,\mathscr{I}_{i_k}}$, storing $B = AA^\top$ at initialization can enhance the efficiency of Algorithm 2.1. This strategy is also used by the GRK method [2, 4] and the weighted RK method [30].

4.2. Choice of parameters for GRABP algorithm

The optimal block size selection for the partition sampling (2.2) has been analyzed by Necoara for solving linear systems under the assumption that the matrix A is normalized — i.e. $\|A_{i,:}\|_2 = 1$ for all $i \in [m]$. Specifically, Necoara estimated the term ξ in (2.5) and demonstrated that the optimal block size is approximately $p^* \backsimeq m/\|A\|_2^2$, cf. [26, Section 4.1] and [25, Section 4.3].

We first examine the impact of the block size β on the convergence behavior of Algorithm 2.1 with constant stepsizes (2.5) (GRABP-c) and adaptive stepsizes (2.6) (GRABP-a) for general matrices, particularly focusing on randomly generated matrices. The matrix $AA^\top$ is precomputed and stored during initialization and we set $w = 1$. The performance of the algorithms is evaluated using two metrics: the computational time (CPU) and the number of full iterations ($k \cdot (p/m)$). This ensures that all algorithms perform an equivalent number of operations per full pass through the rows of A. The results are displayed in Fig. 1.

Fig. 1 shows that the increase of the value of β results in a larger number of full iterations. This suggests that smaller values of β improve the performance of the GRABP method in terms of the number of full iterations. Note that the convergence upper bound in Theorem 3.1 depends on the parameter

$$\zeta = \max_{1 \le i \le t} \frac{\sigma_{\max}^2 (A_{\mathscr{I}_i,:})}{\|A_{\mathscr{I}_i,:}\|_F^2}.$$

It can be observed that $\zeta \le 1$, and when $t = m$ (i.e. $\beta = 1$), ζ can attain the value 1.

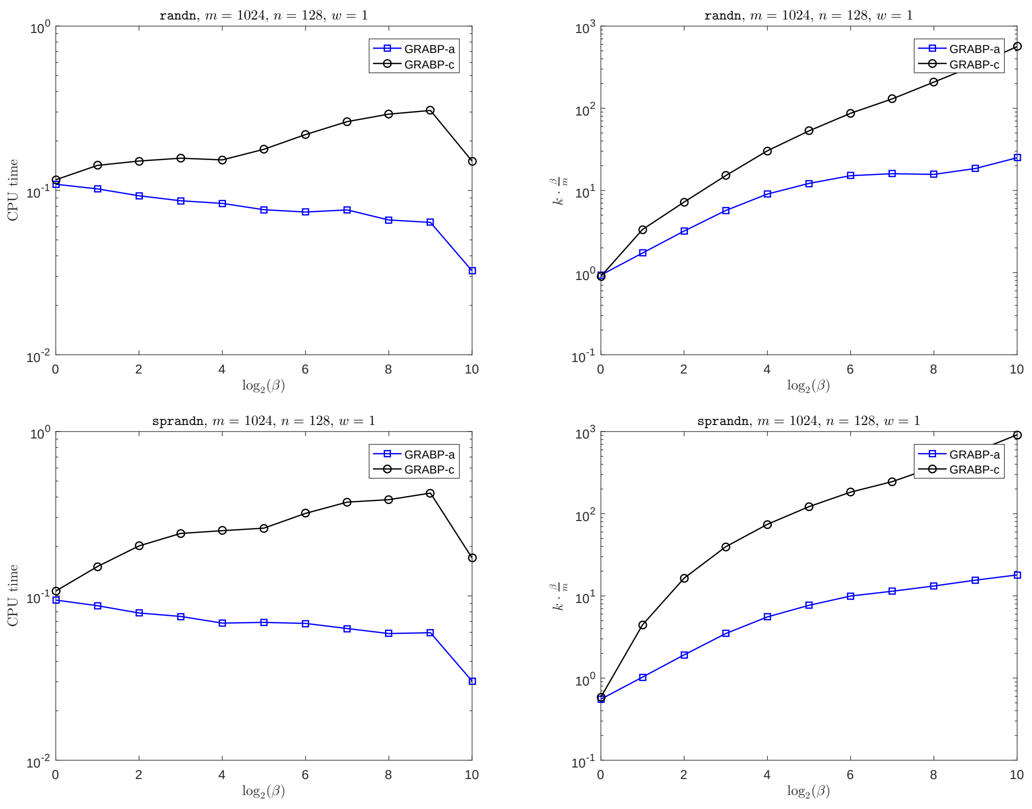

Figure 1: Figures depict the evolution of the CPU time (left) and the number of full iterations (right) with respect to the block size β. The title of each plot indicates the values of m, n, w, and the type of coefficient matrices.

Furthermore, as t decreases (i.e. β increases), the value of ζ tends to decrease. Hence, the algorithm can achieve better convergence upper bound when $\beta = 1$. The numerical observations above are consistent with the convergence analysis. However, variations in CPU time for different values of β are relatively minor, with some instances showing an initial increase followed by a slight decrease. This behavior can be attributed to MATLAB's automatic multithreading when computing matrix-vector products, which constitutes the computational bottleneck in block sampling-based methods. As a result, despite the higher number of full iterations required by the GRABP method, its wall-clock time performance remains competitive.

Figs. 2 and 3 illustrate the impact of the stepsize w on the performance of the algorithm. It can be observed that larger stepsizes, such as $w = 1.6$, lead to better performance of the algorithm. Furthermore, the performance of GRABP-a is often superior to that of GRABP-c.

Finally, we note that during our test, it has been consistently observed that the GRABP-a method is slight better the GRABP-c method. Consequently, our subsequent tests will solely focus on the GRABP-a method. Additionally, we will set $w = 1.6$ and $\beta = 20$, as this choice requires relatively less CPU time.

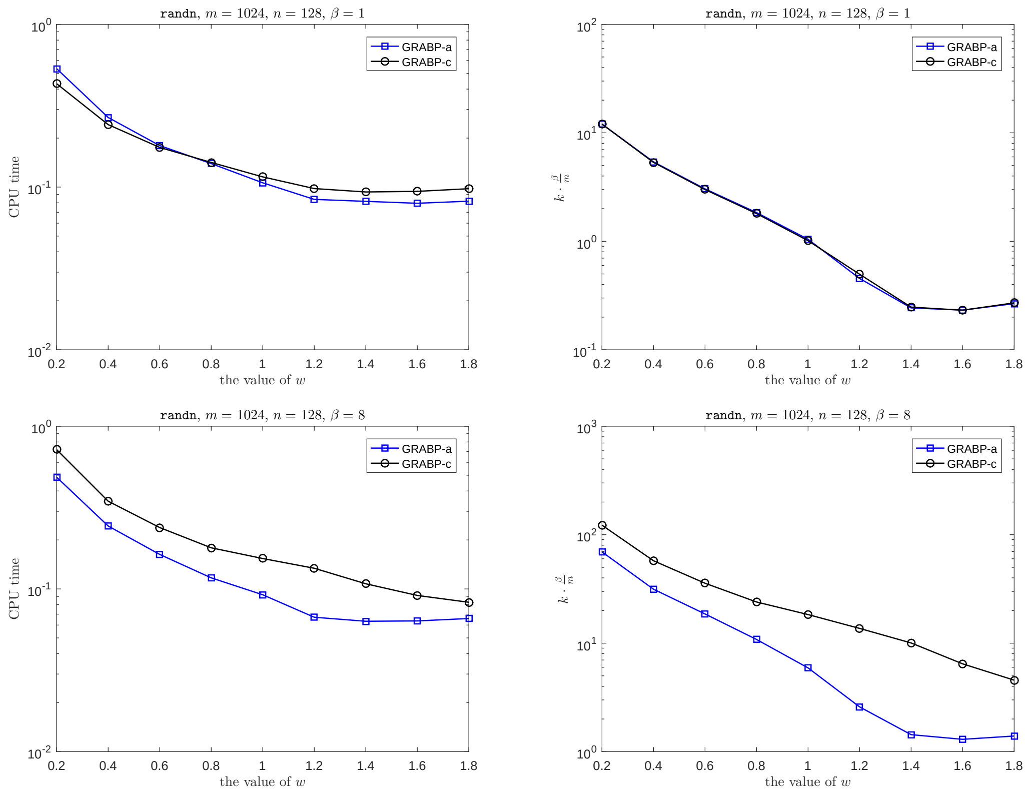

Figure 2: Figures depict the evolution of the CPU time (left) and the number of full iterations (right) with respect to the values of step-size w or α. The title of each plot indicates the values of m, n, β, and the type of coefficient matrices.

4.3. Comparison to the other methods: Randomly generated instances

We now compare the GRABP-a method to other related methods for solving linear feasibility problem, including the RP method, the SKM method, the GSKM method, and the PASKM method. As mentioned above, we set $w = 1.6$ and $\beta = 20$ for the GRABP-a method. We denote the GRABP-a method without pre-stored matrix $AA^\top$ as GRABP-a1 and with pre-stored matrix $AA^\top$ as GRABP-a2.

Tables 1-4 present the results of our numerical experiments. In Tables 1 and 2, we test two sets of coefficient matrices with a fixed number of rows but an increasing number of columns. Tables 3 and 4, on the other hand, demonstrate the performance of the algorithms for coefficient matrices of varying sizes. From the results, we observe that the GRABP-a method is competitive compared to the other methods. Moreover, the GRABP-a2 method exhibits superior performance compared to the GRABP-a1 method in most cases.

4.4. Comparison with other methods: Real-world test instances

In this section, we consider the two following types of real-world test instances: SuiteSparse Matrix Collection and sparse Netlib LP instances.

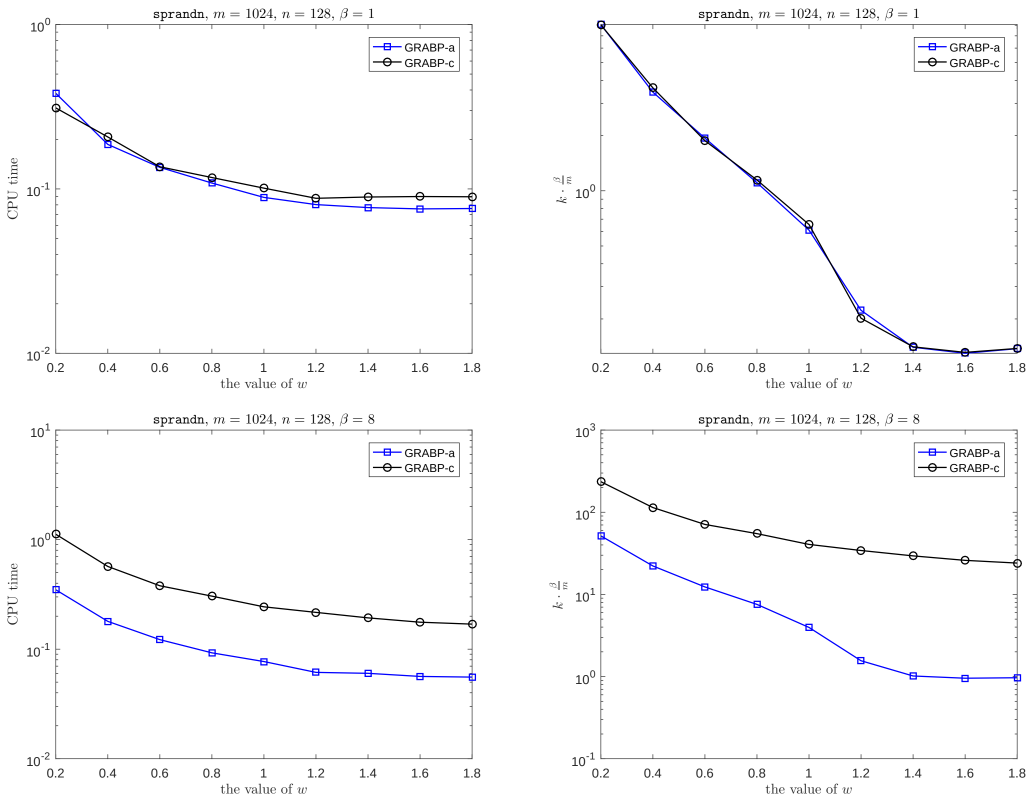

Figure 3: Figures depict the evolution of the CPU time (left) and the number of full iterations (right) with respect to the values of step-size w or α. The title of each plot indicates the values of m, n, β, and the type of coefficient matrices.

Table 1: Numerical results for m-by-n random dense matrices A with $m = 5000$ and different n.

	n	100	200	300	400	500	600
RP	IT	43963	68947.4	84456.2	108291.2	137631	154000.1
	CPU	2.5254e+00	4.1026e+00	5.2832e+00	7.3424e+00	8.9144e+00	1.0353e+01
SKM	IT	851.6	1794.8	2677.8	4153.9	5636.8	8151.5
	CPU	7.0240e-01	1.4169e+00	2.2188e+00	3.6475e+00	5.2375e+00	8.0360e+00
GSKM	IT	846.3	1469.1	2167.2	3400.7	4614.6	6393.3
	CPU	5.8890e-01	1.4398e+00	2.7218e+00	5.0842e+00	8.0868e+00	1.2934e+01
PASKM	IT	437.2	753.7	1025.5	1292.2	1542.2	1831.3
	CPU	**3.1560e-01**	7.7260e-01	1.3429e+00	2.0318e+00	2.8530e+00	3.9030e+00
GRABP-a1	IT	**87.4**	**181.4**	278.4	**366.6**	468.9	579.1
	CPU	9.2770e-01	1.2723e+00	1.6125e+00	1.8892e+00	2.2025e+00	2.5523e+00
GRABP-a2	IT	89.4	185.1	**274.7**	371	**452.3**	**576.6**
	CPU	4.3130e-01	**7.1660e-01**	**1.0206e+00**	**1.3464e+00**	**1.7822e+00**	**2.2011e+00**

Table 2: Numerical results for m-by-n random sparse matrices A with $m = 5000$ and different n.

	n	100	200	300	400	500	600
RP	IT	60534.7	98296.6	141994.6	161968.8	182128.1	243855.8
	CPU	3.3537e+00	5.5948e+00	8.2723e+00	1.0447e+01	1.1340e+01	1.5541e+01
SKM	IT	1352.6	2089.8	3157.4	3695.6	5548.9	7473
	CPU	9.4130e-01	1.5779e+00	2.5671e+00	3.2981e+00	5.2199e+00	7.5287e+00
GSKM	IT	911.2	1809.3	2609.6	3043.6	4459.1	5856.7
	CPU	6.4800e-01	1.7574e+00	3.2579e+00	4.5788e+00	7.8469e+00	1.2207e+01
PASKM	IT	447.5	1118.7	1524.8	1847.6	2804.9	3774
	CPU	**3.1900e-01**	1.1021e+00	1.9437e+00	2.9475e+00	5.0739e+00	8.0506e+00
GRABP-a1	IT	98.1	**171.9**	**247.2**	**319.9**	393.8	474.7
	CPU	8.7700e-01	1.2198e+00	1.5452e+00	1.8376e+00	2.1600e+00	2.5025e+00
GRABP-a2	IT	**96.5**	175.1	249.8	321.2	**390.7**	**463.1**
	CPU	4.3720e-01	**7.1590e-01**	**1.0076e+00**	**1.3270e+00**	**1.6700e+00**	**2.0449e+00**

Table 3: Numerical results for m-by-n random dense matrices A.

	$m \times n$	1000×100	2000×200	3000×300	4000×400	5000×500	6000×600
RP	IT	25712.2	54449.5	81759.6	102540.2	126600.4	165938.7
	CPU	3.2000e-01	1.2071e+00	2.6451e+00	6.2102e+00	8.3203e+00	1.2725e+01
SKM	IT	1054.4	2350.2	3690.3	4420.1	5696.2	7272.5
	CPU	1.1980e-01	6.3460e-01	1.7908e+00	3.2173e+00	5.3215e+00	8.5472e+00
GSKM	IT	915.6	1904.1	2875.3	3516.6	4565.7	5605.3
	CPU	1.1870e-01	7.3880e-01	2.3048e+00	4.5733e+00	8.2064e+00	1.3529e+01
PASKM	IT	259.5	552.9	848.3	1217.1	1540.6	1810.7
	CPU	4.1800e-02	2.3320e-01	7.2470e-01	1.6514e+00	2.8602e+00	4.5500e+00
GRABP-a1	IT	**86.6**	**177.7**	**274.1**	**364.5**	463.9	**564.1**
	CPU	4.4600e-02	2.0510e-01	5.9530e-01	1.2253e+00	2.2370e+00	3.6417e+00
GRABP-a2	IT	87.6	179.9	280.9	366.6	**459.9**	568.2
	CPU	**2.9400e-02**	**1.4830e-01**	**4.3820e-01**	**9.8520e-01**	**1.7660e+00**	**2.9581e+00**

Table 4: Numerical results for m-by-n random sparse matrices A.

	$m \times n$	1000×100	2000×200	3000×300	4000×400	5000×500	6000×600
RP	IT	53539	89179.6	113720.9	153180.1	176028.7	204318.4
	CPU	6.2330e-01	1.8068e+00	3.3828e+00	8.6569e+00	1.0910e+01	1.4931e+01
SKM	IT	916.8	1677.5	2980.9	4161.8	4829.9	6448.8
	CPU	1.0670e-01	4.4550e-01	1.3800e+00	2.9238e+00	4.4743e+00	7.4542e+00
GSKM	IT	748.4	1347	2448.7	3362.4	3901.8	5184
	CPU	9.2600e-02	4.4560e-01	1.8720e+00	4.2479e+00	6.7534e+00	1.2242e+01
PASKM	IT	518.6	855.6	1594.4	2221.4	2447.5	3344.8
	CPU	6.8300e-02	3.6480e-01	1.2499e+00	2.9289e+00	4.3315e+00	8.0947e+00
GRABP-a1	IT	**88.1**	**150.6**	240.8	306.7	**385.6**	**470.4**
	CPU	3.9900e-02	2.0470e-01	5.5120e-01	1.2089e+00	2.1364e+00	3.6596e+00
GRABP-a2	IT	91.7	154.2	**237.3**	**304.6**	392	473.1
	CPU	**2.8500e-02**	**1.3040e-01**	**4.0960e-01**	**9.2100e-01**	**1.7181e+00**	**2.8719e+00**

4.4.1. SuiteSparse Matrix Collection

In Table 6, the coefficient matrix A is chosen from the SuiteSparse Matrix Collection. In Table 5, we list the size, density, condition number, and the squared Euclidean norm. Note that the density of a matrix is defined by

$$\text{density} = \frac{\text{number of nonzeros of an } m\text{-by-}n \text{ matrix}}{mn}.$$

In addition, the right-hand side b is generated randomly as $b = 0.5Ax_1 + 0.5Ax_2 + x_3$. From Table 6, we can also observe that the GRABP-a, PASKM, GSKM, and SKM methods outperform the RP method in terms of both the iteration count and the CPU time. In addition, the GRABP-a2 method demonstrates superior performance compared to the other methods in most cases.

Table 5: Properties of m-by-n matrices A from the SuiteSparse Matrix Collection.

Name	m	n	density	cond(A)	$\|A\|_2^2$
ash958	958	292	0.68%	3.2014	17.9630
illc1033	1033	320	1.43%	1.8888e+04	4.5983
well1033	1033	320	1.43%	166.1333	3.2635
ch8_8_b1	1568	64	3.13%	3.3502e+15	56
illc1850	1850	712	0.66%	1.404e+03	4.5086
Franzl	2240	768	0.30%	8.0481e+15	17.4641

Table 6: Numerical results for matrices A from the SuiteSparse Matrix Collection.

	Name	ash958	illc1033	well1033	ch8-8-b1	illc1850	Franzl
RP	IT	4717.2	5736.8	9214	16535.8	12272.8	81189.8
	CPU	9.0900e-02	1.0750e-01	1.5260e-01	2.6770e-01	4.2110e-01	2.6125e+00
SKM	IT	411.8	201.4	347.8	292.9	479.8	3505.3
	CPU	8.0700e-02	6.0000e-02	7.9000e-02	5.8200e-02	4.3580e-01	2.7327e+00
GSKM	IT	309.7	148.7	251	238.7	360	2682.1
	CPU	6.6500e-02	**4.1400e-02**	6.0100e-02	**3.6200e-02**	4.4960e-01	4.0129e+00
PASKM	IT	357.5	211.5	347.6	319.8	483.3	3494.1
	CPU	9.3500e-02	6.7200e-02	9.8100e-02	5.3400e-02	7.3490e-01	5.4663e+00
GRABP-a1	IT	34.2	**29.4**	41.8	28.7	60.9	177
	CPU	6.2800e-02	8.2300e-02	7.6500e-02	6.4300e-02	4.1510e-01	6.6800e-01
GRABP-a2	IT	35.4	30.3	**41.8**	28.8	**60.7**	**176.7**
	CPU	**4.3200e-02**	5.1400e-02	**5.4500e-02**	3.9600e-02	**2.8960e-01**	**5.2800e-01**

4.4.2. Netlib LP instances

In this subsection, we compare the performance of the algorithms for solving Netlib LP test instances. We follow the standard framework used by De Loera [6] and Morshed [23]

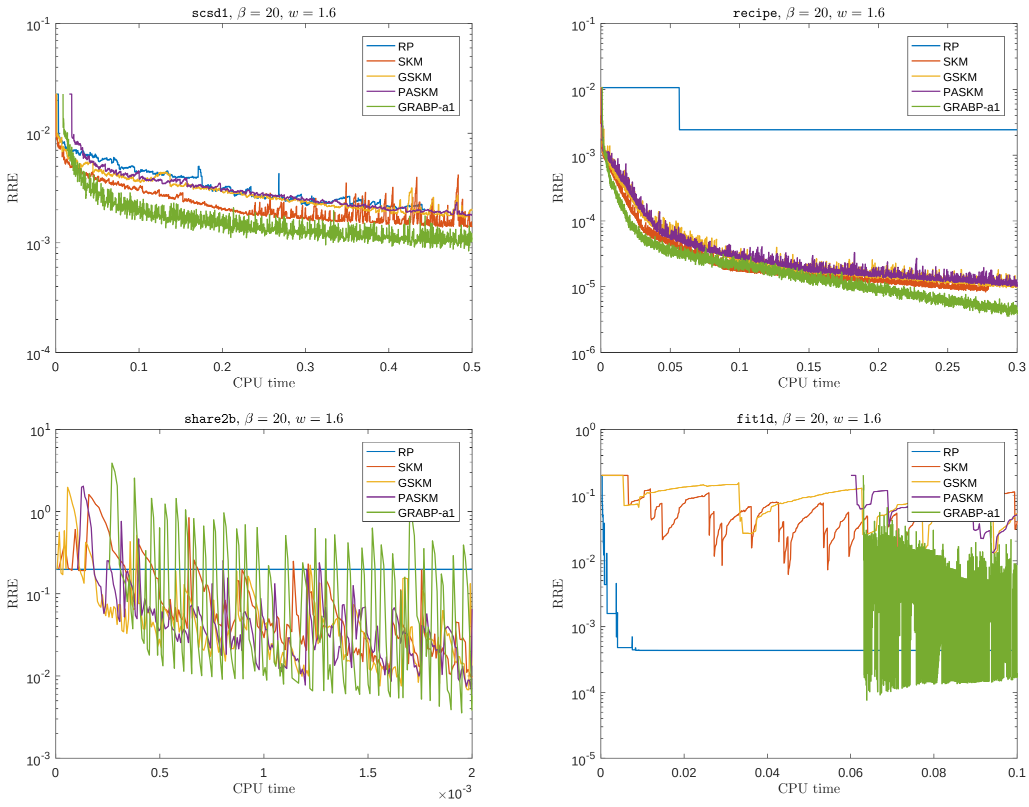

Figure 4: Performance of RP, SKM, GSKM, PASKM, and GRABP-a1 for linear feasibility with coefficient matrices from Netlib LP [28]. Figures depict the evolution of RRE with respect to the CPU time (in seconds). The title of each plot indicates the names of the data and the values of β, w. We stop the algorithms if the number of iterations exceeds a certain limit.

in their work for linear feasibility problems. The problem instances are transformed form standard LP problems (i.e. min $c^\top x$ subject to $Ax = b$, $l \le x \le u$ with optimum value p^*) to an equivalent linear feasibility formulation (i.e. $\mathbf{A}x \le \mathbf{b}$, where $\mathbf{A} = [A^\top \; -A^\top \; I \; -I \; c]^\top$ and $\mathbf{b} = [b^\top \; -b^\top \; u^\top \; -l^\top \; p^*]^\top$). In this test, we only present the results for GRABP-a1, as GRABP-a2 requires more CPU time. Fig. 4 illustrates our findings, showing that the GRABP-a1 method delivers superior performance.

5. Conclusions

This paper introduced a GRABP method, which inherits ideas from the greedy probability criterion and the average block method, for solving linear feasibility problems. It was proved that the GRABP method converges linearly with two kinds of choices of extrapolation steps. Numerical results show that the GRABP method works better than several state-of-the-art methods.

Finally, it should be pointed out that the RP method and its variants only ensure that the iteration sequence $\{x^k\}_{k\geq 0}$ converges to a certain feasible point in S. Precisely, Theorem 3.1 only guarantee that the distance between x^k and S converges to zero. However, in practice one may want to find solutions with a certain structure in S, e.g. the least norm solution. The Hildreth's method [13, 15, 16, 18] is also a row action method for solving linear feasibility problems, but with one more benefit: finding the closest point in the solution set to a given point x^0, i.e. its iteration sequence $\{x^k\}_{k\geq 0}$ converges to $P_S(x^0)$. This topic is practically valuable and theoretically meaningful, and will be investigated in detail and discussed in depth in the future.

Acknowledgments

We would like to thank the anonymous reviewers for their comments, which helped improve this paper substantially.

J.X. is supported by NSFC (Grant Nos. 12571315, 12001026) and by the Fundamental Research Funds for the Central Universities.

References

[1] S. Agmon, *The relaxation method for linear inequalities*, Canad. J. Math. **6**, 382–392 (1954).

[2] Z.Z. Bai and W.T. Wu, *On greedy randomized Kaczmarz method for solving large sparse linear systems*, SIAM J. Sci. Comput. **40**, A592–A606 (2018).

[3] Z.Z. Bai and W.T. Wu, *On relaxed greedy randomized Kaczmarz methods for solving large sparse linear systems*, Appl. Math. Lett. **83**, 21–26 (2018).

[4] Z.Z. Bai and W.T. Wu, *On greedy randomized augmented Kaczmarz method for solving large sparse inconsistent linear systems*, SIAM J. Sci. Comput. **43**, A3892–A3911 (2021).

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

[6] J.A. De Loera, J. Haddock and D. Needell, *A sampling Kaczmarz-Motzkin algorithm for linear feasibility*, SIAM J. Sci. Comput. **39**, S66–S87 (2017).

[7] K. Du, W.T. Si and X.H. Sun, *Randomized extended average block Kaczmarz for solving least squares*, SIAM J. Sci. Comput. **42**, A3541–A3559 (2020).

[8] R. Gordon, R. Bender and G.T. Herman, *Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography*, J. Theoret. Biol. **29**, 471–481 (1970).

[9] R.M. Gower, D. Molitor, J. Moorman and D. Needell, *On adaptive sketch-and-project for solving linear systems*, SIAM J. Matrix Anal. Appl. **42**, 954–989 (2021).

[10] D. Han, Y.S. Su and J.X. Xie, *Randomized Douglas-Rachford methods for linear systems: Improved accuracy and efficiency*, SIAM J. Optim. **34**, 1045–1070 (2024).

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

[12] G.T. Herman and L.B. Meyer, *Algebraic reconstruction techniques can be made computationally efficient (positron emission tomography application)*, IEEE Trans. Medical Imaging **12**, 600–609 (1993).

[13] C. Hildreth, *A quadratic programming procedure*, Naval Res Logist Quart. **4**, 79–85 (1957).

[14] A.J. Hoffman, *On approximate solutions of systems of linear inequalities*, J. Research Nat. Bur. Standards **49**, 263–265 (1952).

[15] A.N. Iusem and A.R. De Pierro, *On the convergence properties of Hildreth's quadratic programming algorithm*, Math. Program. **47**, 37–51 (1990).

[16] N. Jamil, X.M. Chen and A. Cloninger, *Hildreth's algorithm with applications to soft constraints for user interface layout*, J. Comput. Appl. Math. **288**, 193–202 (2015).

[17] S. Kaczmarz, *Angenäherte auflösung von systemen linearer gleichungen*, Bull. Int. Acad. Pol. Sci. Lett A **35**, 355–357 (1937).

[18] A. Lent and Y. Censor, *Extensions of Hildreth's row-action method for quadratic programming*, SIAM J. Control Optim. **18**, 444–454 (1980).

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

[20] C.Q. Miao and W.T Wu, *On greedy randomized average block Kaczmarz method for solving large linear systems*, J. Comput. Appl. Math. **413**, 114372 (2022).

[21] J.D. Moorman, T.K. Tu, D. Molitor and D. Needell, *Randomized Kaczmarz with averaging*, BIT Numer. Math. **61**, 337–359 (2021).

[22] M.S. Morshed, S. Ahmad and M. Noor-E-Alam, *Stochastic steepest descent methods for linear systems: Greedy sampling & momentum*, arXiv: 2012.13087, (2020).

[23] M.S. Morshed, M.S. Islam and M. Noor-E-Alam, *Sampling Kaczmarz-Motzkin method for linear feasibility problems: Generalization and acceleration*, Math. Program. **194**, 719–779 (2022).

[24] T.M. Motzkin and I.J. Schoenberg, *The relaxation method for linear inequalities*, Canad. J. Math. **6**, 393–404 (1954).

[25] I. Necoara, *Faster randomized block Kaczmarz algorithms*, SIAM J. Matrix Anal. Appl. **40**, 1425–1452 (2019).

[26] I. Necoara, *Stochastic block projection algorithms with extrapolation for convex feasibility problems*, Optimization Methods and Software **1**, 1–31 (2022).

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

[28] Netlib, *netlib/LP/DATA*. `https://www.netlib.org/lp/data/`

[29] J. Pena, J.C. Vera and L.F. Zuluaga, *New characterizations of Hoffman constants for systems of linear constraints*, Math. Program. **187**, 79–109 (2021).

[30] S. Stefan, *A weighted randomized Kaczmarz method for solving linear systems*, Math. Comp. **90**, 2815–2826 (2021).

[31] T. Strohmer and R. Vershynin, *A randomized Kaczmarz algorithm with exponential convergence*, J. Fourier Anal. Appl. **15**, 262–278 (2009).

[32] Y.S. Su, D. Han, Y. Zeng and J.X. Xie, *On the convergence analysis of the greedy randomized Kaczmarz method*, arXiv:2307.01988, (2023).

[33] Y.S. Su, D. Han, Y. Zeng and J.X. Xie, *On greedy multi-step inertial randomized Kaczmarz method for solving linear systems*, Calcolo **61**, 68 (2024).

[34] J.A. Tropp, *Column subset selection, matrix factorization, and eigenvalue optimization*, in: Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, 978–986 (2009).

[35] J.X. Xie and Z.Q. Xu, *Subset selection for matrices with fixed blocks*, Israel J. Math. **245**, 1–26 (2021).

[36] Z.Z. Yuan, L. Zhang, H.X. Wang and H. Zhang, *Adaptively sketched Bregman projection methods for linear systems*, Inverse Problems **38**, 065005 (2022).

[37] Y. Zeng, D. Han, Y.S. Su and J.X. Xie, *On adaptive stochastic heavy ball momentum for solving linear systems*, SIAM J. Matrix Anal. Appl. **45**, 1259–1286 (2024).

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