Electronic Research Archive

http://www.aimspress.com/journal/era

ERA, 30(2): 755–779.
DOI: 10.3934/era.2022040
Received: 20 December 2021
Revised: 11 February 2022
Accepted: 20 February 2022
Published: 28 February 2022

Research article

A two-step randomized Gauss-Seidel method for solving large-scale linear least squares problems

Yimou Liao[1], **Tianxiu Lu**[1,2,*] **and Feng Yin**[1]

[1] School of Mathematics and Statistics, Sichuan University of Science and Engineering, Zigong 643000, China

[2] Key Laboratory of Higher Education of Sichuan Province for Enterprise Informationalization and Internet of Things, Bridge Non-destruction Detecting and Engineering Computing Key Laboratory, Zigong 643000, China

* **Correspondence:** Email: lubeeltx@163.com.

Abstract: A two-step randomized Gauss-Seidel (TRGS) method is presented for large linear least squares problem with tall and narrow coefficient matrix. The TRGS method projects the approximate solution onto the solution space by given two random columns and is proved to be convergent when the coefficient matrix is of full rank. Several numerical examples show the effectiveness of the TRGS method among all methods compared.

Keywords: linear least-squares problem; two-step iterative method; convergence property; Gauss-Seidel

1. Introduction

We consider the approximate solutions of a large linear least squares problem

$$\min_{x\in\mathbb{R}^n} ||b - Ax||_2^2, \tag{1.1}$$

where $A \in \mathbb{R}^{m\times n}$, $m > n$, is of full clolumn rank. A and $b \in \mathbb{R}^m$ are known, $x \in \mathbb{R}^n$ is unknown to be determined. Under the condition that the linear system is consistent or inconsistent, people are interested in finding the unique least squares solution $x_* = A^\dagger b$, where $A^\dagger$ is the Moore-Penrose pseudoinverse of the matrix A ($A^\dagger = (A^TA)^{-1}A^T$, where A^T denotes the transpose of the matrix A). See also references [1–6]. As we know, the least squares problem arises widely in many fields such as tomography [7–9], protein structure [10], machine learning [11], biological feature selection [12], and so on [13–15]. For solving (1.1), there are many direct methods, such as QR decomposition and singular value decomposition (SVD) [1, 16]. However, for large-scale system matrices, these methods are too expensive because they consume a lot of memory space. So, some iterative methods are applied to solve large-scale linear least squares problems.

As one of the famous iterative algorithms, the Gauss-Seidel method [17] selects a coordinate d_k and a step $\alpha_k = arg \min_{\alpha\in\mathbb{R}} f(x_k + \alpha d_k)$ in each iteration. When $\alpha_k = A_{j_k}^T(b - Ax_k)/\left\|A_{j_k}\right\|_2^2$ and $d_k = e_{j_k}$ are accurately given, it generates the following iterative process:

$$x_{k+1} = x_k + \frac{A_{j_k}^T(b - Ax_k)}{\|A_{j_k}\|_2^2} e_{j_k}, \quad k = 0, 1, 2, \cdots, \tag{1.2}$$

where $j_k = (k \ mod \ n) + 1$, $(\cdot)^T$ represents the transpose of a matrix or a vector, and e_{j_k} represents the coordinate column vector, whose j_k-th entry is 1 and zero otherwise. Inspired by the randomized Kaczmarz (RK) linear convergence characteristics of Strohmer and Vershynin [18], Leventhal and Lewis [19] obtained a randomized Gauss-Seidel (RGS) method, which is also known as randomized coordinate descent (RCD) method to solve (1.1). The RGS selects the update column index j_k according to the appropriate probability. Theoretical analysis shows that RGS converges linearly to the unique least squares solution $x_* = A^\dagger b$. Many variants of RGS are receiving extensive attention recently due to its good performance. For example, the versions of block [20, 21], random greedy [22–24]. Other versions, see literatures [4, 17, 25, 26] and reference therein.

In 2018, Wu [27] obtained a randomized block Gauss-Seidel (RBGS) method, which can significantly improve the convergence speed. However, the good column pavings for the RBGS is difficult to find when the column norms of the coefficient matrix fluctuate in a large range. In this paper, we propose a two-step randomimzed Gauss-Seidel method (TRGS), which does not need any columns pavings. The convergence of the TRGS algorithm is proved for (1.1) with the coefficient matrix of full rank.

This paper is organized as follows. In Section 2, some symbols and a lemma related to RGS are introduced. The convergence of RGS2 is analyzed. In Section 3, a convergent upper bound of TRGS is obtained theoretically. Several numerical experiments are reported to verify the feasibility of our proposed algorithms in Section 4. The conclusions of this paper are given in Section 5 and the proof of Theorem 2 is shown in appendix.

2. The simplified two-step randomized Gauss-Seidel method

We first introduce the notations and definitions as follows. For the matrix $Q \in \mathbb{R}^{m\times n}$, Q_i represents the ith column of the Q, $\lambda_{max}(Q^TQ)$ and $\lambda_{min}(Q^TQ)$ represent the maximum and minimum positive eigenvalues of the Q^TQ, respectively, $(\cdot)^T$ represents the transpose of a matrix or a vector, the Q_{τ_k} represents a submatrix of the Q (where τ_k is a set of column indexes), $\|Q\|_2$ and $\|Q\|_F$ represent the spectral norm and Frobenius norm of the Q, respectively, and $\mathcal{R}(Q)$ is the image space of matrix Q. For a vector $p \in \mathbb{R}^n$ or $p \in \mathbb{R}^m$, p_i is the ith component of p. For constant $c \in \mathbb{R}$, $[c]$ refers to the set consising of all positive integers not exceeding c. For the matrix in (1.1), assuming that every two columns are independent and identically distributed, the correlation coefficient parameters of the matrix A are defined as follows

$$\delta = \min_{s\neq t} \frac{|A_s^TA_t|}{\|A_s\|_2\|A_t\|_2} \quad and \quad \Delta = \max_{s\neq t} \frac{|A_s^TA_t|}{\|A_s\|_2\|A_t\|_2}, \quad s, t \in \{1, 2, \cdots, n\}.$$

Then, we have $0 \le \delta \le \Delta < 1$.

The randomized Gauss-Seidel method proposed by Leventhal and Lewis [19] consists of two parts. The RGS determines a column index j_k according to the probability $Pr(column = j_k) = \frac{\|A_{j_k}\|_2^2}{\|A\|_F^2}$ and then updates x_{k+1} by (1.2).

The following results in [19] summarized an upper bound for the error of the solution in expectation on the convergence of RGS algorithm.

Lemma 1. *Assume the least squares problems* (1.1) *has tall and narrow coefficient matrix* $A \in \mathbb{R}^{m\times n}$ *with full rank. Let* $x_* = A^\dagger b$ *be a solution of* (1.1)*. Given an initial guess* $x_0 \in \mathbb{R}^n$*, then the sequence* $\{x_k\}_0^\infty$ *generated by RGS algorithm converges linearly to the* x_* *in expectation. Moreover, it satisfies*

$$E\|x_k - x_*\|_{A^TA}^2 \le \left(1 - \frac{\lambda_{min}(A^TA)}{\|A\|_F^2}\right)^k \|x_0 - x_*\|_{A^TA}^2, \quad k = 1, 2, \cdots .$$

Algorithm 1 RGS2 method

Input: $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m$ and $x_0 = \mathbf{0} \in \mathbb{R}^n$

Output: the approximation solution x_k of (1.1)

1. **for** $k = 0, 1, \ldots$ do until termination criterion is satisfied
2. Set $r_k = b - Ax_k$ and $s_k = A^T r_k$.
3. Select j_{k_1} and j_{k_2} with probability by (2.1).
4. Compute $s_k^{j_{k_1}} = A_{j_{k_1}}^T r_k$.
5. Update

$$y_k = x_k + \frac{s_k^{j_{k_1}}}{\|A_{j_{k_1}}\|_2^2} e_{j_{k_1}} \text{ and } x_{k+1} = y_k + \frac{A_{j_{k_2}}^T (b - Ay_k)}{\|A_{j_{k_2}}\|_2^2} e_{j_{k_2}}.$$

6. **end for**

Now, we propose a simplified two-step randomized Gauss-Seidel method (RGS2) to solve (1.1). Algorithm 1 lists the RGS2 method, which mainly consists of two stages. The first stage is to select two different working columns by

$$Pr(\text{column} = j_{k_1}) = \frac{\|A_{j_{k_1}}\|_2^2}{\|A\|_F^2}, \quad Pr(\text{column} = j_{k_2}) = \frac{\|A_{j_{k_2}}\|_2^2}{\|A\|_F^2 - \|A_{j_{k_1}}\|_2^2}, \tag{2.1}$$

where $j_{k_1} \in \{1, 2, \cdots, n\}$, $j_{k_2} \in \{1, 2, \cdots, n\}/\{j_{k_1}\}$. The second stage is to use (1.2) twice to update x_k.

If the coefficient matrix A in (1.1) is tall and narrow with full column rank, the following result gives the convergence of the RGS2 algorithm.

Theorem 1. *Assume the least squares problem* (1.1) *has tall and full-rank coefficient matrix* $A \in \mathbb{R}^{m\times n}$*. Let* $x_* = A^\dagger b$ *be a solution of* (1.1)*. Given an initial guess* $x_0 \in \mathbb{R}^n$*, then the iterative sequence* $\{x_k\}_0^\infty$ *generated by RGS2 algorithm converges linearly to the* x_* *in expectation. Moreover, it satisfies*

$$E_k\|A(\hat{x}_{k+1} - x_*)\|_2^2 \le \left[(1 - \frac{\lambda_{min}(A^TA)}{\tau_{max}})(1 - \frac{\lambda_{min}(A^TA)}{\|A\|_F^2})\right] \cdot \|A(x_k - x_*)\|_2^2$$

and

$$E\|x_k - x_*\|_{A^TA}^2 \le \left[(1 - \frac{\lambda_{min}(A^TA)}{\tau_{max}})(1 - \frac{\lambda_{min}(A^TA)}{\|A\|_F^2})\right]^k \cdot \|x_0 - x_*\|_{A^TA}^2, \tag{2.2}$$

where $\tau_{max} = \|A\|_F^2 - \min\limits_{p\in[n]} \|A_p\|_2^2$.

Proof. Let $\hat{y}_k$ and $\hat{x}_{k+1}$ be the first and the second iterative solutions of x_k obtained by single-step continuous execution of Algorithm 1, respectively. The update process is divided into two steps as follows

$$\hat{y}_k = x_k + \frac{A_{j_{k_1}}^T(b - Ax_k)}{\|A_{j_{k_1}}\|_2^2} e_{j_{k_1}} \quad \text{and} \quad \hat{x}_{k+1} = \hat{y}_k + \frac{A_{j_{k_2}}^T(b - A\hat{y}_k)}{\|A_{j_{k_2}}\|_2^2} e_{j_{k_2}}.$$

Let $P_{j_{k_i}} = I_m - \frac{A_{j_{k_i}} A_{j_{k_i}}^T}{\|A_{j_{k_i}}\|_2^2}$ satisfy $P_{j_{k_i}}^2 = P_{j_{k_i}}, P_{j_{k_i}}^T = P_{j_{k_i}}$, where $i = 1, 2$. Then, $P_{j_{k_i}}$ is the projection matrix, and

$$\begin{aligned}
A(\hat{x}_{k+1} - x_*) &= A(\hat{y}_k - x_*) + \frac{A_{j_{k_2}} A_{j_{k_2}}^T (b - A\hat{y}_k)}{\|A_{j_{k_2}}\|_2^2} = (I_m - \frac{A_{j_{k_2}} A_{j_{k_2}}^T}{\|A_{j_{k_2}}\|_2^2}) A(\hat{y}_k - x_*) \\
&= (I_m - \frac{A_{j_{k_2}} A_{j_{k_2}}^T}{\|A_{j_{k_2}}\|_2^2})(I_m - \frac{A_{j_{k_1}} A_{j_{k_1}}^T}{\|A_{j_{k_1}}\|_2^2}) A(x_k - x_*) \\
&= P_{j_{k_2}} P_{j_{k_1}} A(x_k - x_*),
\end{aligned}$$

where the second equality follows from the normal equation $A^T A x_* = A^T b$, whose j_{k_2}-th equation gives $A_{j_{k2}}^T b = A_{j_{k_2}}^T A x_*$. Taking the expectation on the equality above, and by the expectation conditional upon j_{k_1} (we fix the choice of j_{k_1} and averagever the random index j_{k_2}), one can get

$$\begin{aligned}
E_k\|A(\hat{x}_{k+1} - x_*)\|_2^2 &= E_k\|P_{j_{k_2}} P_{j_{k_1}} A(x_k - x_*)\|_2^2 = E_k\big[(A(x_k - x_*))^T P_{j_{k_1}} P_{j_{k_2}} P_{j_{k_1}} A(x_k - x_*)\big] \\
&= \sum_{j_{k_1}=1}^{n} \frac{\|A_{j_{k_1}}\|_2^2}{\|A\|_F^2} \sum_{\substack{j_{k_2}=1, \\ j_{k_2}\neq j_{k_1}}}^{n} \frac{\|A_{j_{k_2}}\|_2^2}{\|A\|_F^2 - \|A_{j_{k_1}}\|_2^2} (A(x_k - x_*))^T P_{j_{k_1}} P_{j_{k_2}} P_{j_{k_1}} A(x_k - x_*) \\
&= \sum_{j_{k_1}=1}^{n} \frac{\|A_{j_{k_1}}\|_2^2}{\|A\|_F^2} (A(x_k - x_*))^T P_{j_{k_1}} \Big(I_m - \frac{AA^T - A_{j_{k_1}} A_{j_{k_1}}^T}{\|A\|_F^2 - \|A_{j_{k_1}}\|_2^2}\Big) P_{j_{k_1}} A(x_k - x_*) \\
&= \sum_{j_{k_1}=1}^{n} \frac{\|A_{j_{k_1}}\|_2^2}{\|A\|_F^2} (A(x_k - x_*))^T P_{j_{k_1}} \Big(I_m - \frac{AA^T}{\|A\|_F^2 - \|A_{j_{k_1}}\|_2^2}\Big) P_{j_{k_1}} A(x_k - x_*),
\end{aligned}$$

in which the last equation holds because

$$\frac{A_{j_{k_1}} A_{j_{k_1}}^T}{\|A\|_F^2 - \|A_{j_{k_1}}\|_2^2} P_{j_{k_1}} = \frac{A_{j_{k_1}} A_{j_{k_1}}^T}{\|A\|_F^2 - \|A_{j_{k_1}}\|_2^2}\Big(I_m - \frac{A_{j_{k_1}} A_{j_{k_1}}^T}{\|A_{j_{k_1}}\|_2^2}\Big) = \frac{A_{j_{k_1}} A_{j_{k_1}}^T}{\|A\|_F^2 - \|A_{j_{k_1}}\|_2^2} - \frac{A_{j_{k_1}} (A_{j_{k_1}}^T A_{j_{k_1}}) A_{j_{k_1}}^T}{(\|A\|_F^2 - \|A_{j_{k_1}}\|_2^2)\|A_{j_{k_1}}\|_2^2} = 0.$$

Note that

$$\|A^T u\|_2^2 \geq \lambda_{min}(A^T A)\|u\|_2^2 \quad \text{and} \quad \tau_{max} = \max_{p\in[n]}\{\|A\|_F^2 - \|A_p\|_2^2\},$$

then $\|I_m - \frac{AA^T}{\|A\|_F^2 - \|A_{j_{k_1}}\|_2^2}\|_2 \leq 1 - \frac{\lambda_{\min}(A^T A)}{\tau_{\max}}$ and

$$E_k\|A(\hat{x}_{k+1} - x_*)\|_2^2 \leq (1 - \frac{\lambda_{min}(A^T A)}{\tau_{max}}) \sum_{j_{k_1}=1}^{n} \frac{\|A_{j_{k_1}}\|_2^2}{\|A\|_F^2} (A(x_k - x_*))^T (I_m - \frac{A_{j_{k_1}} A_{j_{k_1}}^T}{\|A\|_F^2}) \cdot A(x_k - x_*)$$

$$\leq (1-\frac{\lambda_{min}(A^TA)}{\tau_{max}})(1-\frac{\lambda_{min}(A^TA)}{\|A\|_F^2})\|A(x_k-x_*)\|_2^2.$$

Therefore,

$$E_k\|\hat{x}_{k+1}-x_*\|_{A^TA}^2 \leq (1-\frac{\lambda_{min}(A^TA)}{\tau_{max}})(1-\frac{\lambda_{min}(A^TA)}{\|A\|_F^2})\|x_k-x_*\|_{A^TA}^2. \tag{2.3}$$

(2.2) is obtained from the recurrence relation of (2.3) and the full expectation formula. This completed the proof.

3. Two-step randomized Gauss-Seidel method

The minimum positive eigenvalue of $\lambda_{min}(A^TA)$ will become very small if the matrix A has high correlation parameters [28]. A weak bound of the convergence in Theorem 1 will appear. Under this condition, the angle of the unit coordinate direction as the search direction for two consecutive stages may be too small, which is the main reason for the slow convergence of RGS. Inspired by the work of Needell [28] and Wu [29], we iteratively update the solution by continuously seeking two more extensive directions, that is, determine the column pairs (r, s) in advance. So a two-step randomized Gauss-Seidel algorithm is obtained. In a two-step randomized algorithm, which mainly consists of three steps as follows: First, we randomly select the two columns $r, s \in [n]$ according to the probability criterion, then estimate the optimal parameter λ_{opt} for the first iteration with $y_k = x_k + \lambda_{opt}\frac{A_r^T r_k}{\|A_r\|_2^2}e_r$, and finally perform the second iteration to update the iterative solution by $x_{k+1} = y_k + \frac{A_s^T(b-Ay_k)}{\|A_s\|_2^2}e_s$.

We consider the convergence of the TRGS algorithm. We first need the following result presented in [28].

Lemma 2. *For any $\epsilon \in \mathbb{R}$, $\phi, \psi \in \mathbb{R}^m$, the minimizer of $\|\epsilon\phi+\psi\|_2^2$ is $\epsilon_{opt} = -\frac{<\phi,\psi>}{\|\phi\|_2^2}$.*

Now, we note that

$$Ay_k = Ax_k + \lambda_k\frac{A_rA_r^Tr_k}{\|A_r\|_2^2} \quad and \quad Ax_{k+1} = Ay_k + \frac{A_sA_s^T(b-Ay_k)}{\|A_s\|_2^2},$$

then

$$\begin{aligned} Ax_{k+1} &= Ax_k + \lambda_k\frac{A_rA_r^Tr_k}{\|A_r\|_2^2} + \frac{A_sA_s^T\left(b-\left(Ax_k+\lambda_k\frac{A_rA_r^Tr_k}{\|A_r\|_2^2}\right)\right)}{\|A_s\|_2^2} \\ &= \lambda_k\left[\left(\frac{A_r}{\|A_r\|_2}-\frac{\mu_kA_s}{\|A_s\|_2}\right)\frac{A_r^Tr_k}{\|A_r\|_2}\right] + Ax_k + \frac{A_sA_s^Tr_k}{\|A_s\|_2^2}, \end{aligned} \tag{3.1}$$

where $\mu_k = \frac{(A_r)^T(A_s)}{\|A_r\|_2\|A_s\|_2}$. Obviously,

$$\|Ax_{k+1}-b\|_2^2 = \left\|\lambda_k\left[(\frac{A_r}{\|A_r\|_2}-\frac{\mu_kA_s}{\|A_s\|_2})\frac{A_r^Tr_k}{\|A_r\|_2}\right] + Ax_k - b + \frac{A_sA_s^Tr_k}{\|A_s\|_2^2}\right\|_2^2.$$

The selection of optimal parameter λ_{opt} aims to minimize $\|Ax_{k+1} - b\|_2^2$. By Lemma 2, one can get the optimal value of λ_k, that is,

$$\lambda_{opt} = -\frac{\left(\frac{A_r^T}{\|A_r\|_2} - \frac{\mu_k A_s^T}{\|A_s\|_2}\right)\left(Ax_k - b + \frac{A_s A_s^T r_k}{\|A_s\|_2^2}\right)}{\frac{A_r^T r_k}{\|A_r\|_2}\|\frac{A_r}{\|A_r\|_2} - \frac{\mu_k A_s}{\|A_s\|_2}\|_2^2}.$$

Substituting λ_{opt} into (3.1), we obtain

$$\begin{aligned} Ax_{k+1} = &\left(\frac{\frac{A_r^T b}{\|A_r\|_2} - \mu_k \frac{A_s^T b}{\|A_s\|_2}}{\|\frac{A_r}{\|A_r\|_2} - \mu_k \frac{A_s}{\|A_s\|_2}\|_2^2}\right)\left(\frac{A_r}{\|A_r\|_2} - \mu_k \frac{A_s}{\|A_s\|_2}\right) - \left(\frac{(\frac{A_r^T}{\|A_r\|_2} - \mu_k \frac{A_s^T}{\|A_s\|_2})(Ax_k + \frac{A_s A_s^T r_k}{\|A_s\|_2^2})}{\|\frac{A_r}{\|A_r\|_2} - \mu_k \frac{A_s}{\|A_s\|_2}\|_2^2}\right)\left(\frac{A_r}{\|A_r\|_2} - \mu_k \frac{A_s}{\|A_s\|_2}\right) \\ &+ Ax_k + \frac{A_s A_s^T r_k}{\|A_s\|_2^2}. \end{aligned}$$

So,

$$\begin{aligned} x_{k+1} = &\left(\frac{\frac{A_r^T b}{\|A_r\|_2} - \mu_k \frac{A_s^T b}{\|A_s\|_2}}{\|\frac{A_r}{\|A_r\|_2} - \mu_k \frac{A_s}{\|A_s\|_2}\|_2^2}\right)\left(\frac{e_r}{\|A_r\|_2} - \mu_k \frac{e_s}{\|A_s\|_2}\right) - \left(\frac{(\frac{A_r^T}{\|A_r\|_2} - \mu_k \frac{A_s^T}{\|A_s\|_2})(A(x_k + \frac{A_s^T r_k}{\|A_s\|_2^2} e_s))}{\|\frac{A_r}{\|A_r\|_2} - \mu_k \frac{A_s}{\|A_s\|_2}\|_2^2}\right)\left(\frac{e_r}{\|A_r\|_2} - \mu_k \frac{e_s}{\|A_s\|_2}\right) \\ &+ x_k + \frac{A_s^T r_k}{\|A_s\|_2^2} e_s. \end{aligned}$$

Set $r = j_{k_1}$ and $s = j_{k_2}$ with the optimal parameter λ_{opt}, this process can be described as

$$y_k = x_k + \frac{A_{j_{k_2}}^T (b - Ax_k)}{\|A_{j_{k_2}}\|_2^2} e_{j_{k_2}} \quad \text{and} \quad x_{k+1} = y_k + \left(\frac{\beta_k - u_k^T A y_k}{\|u_k\|_2^2}\right) v_k, \tag{3.2}$$

where

$$\mu_k = \frac{(A_{j_{k_1}})^T (A_{j_{k_2}})}{\|A_{j_{k_1}}\|_2 \|A_{j_{k_2}}\|_2} \quad \text{and} \quad \beta_k = \frac{A_{j_{k_1}}^T b}{\|A_{j_{k_1}}\|_2} - \mu_k \frac{A_{j_{k_2}}^T b}{\|A_{j_{k_2}}\|_2},$$

$$u_k = \frac{A_{j_{k_1}}}{\|A_{j_{k_1}}\|_2} - \mu_k \frac{A_{j_{k_2}}}{\|A_{j_{k_2}}\|_2} \quad \text{and} \quad v_k = \frac{e_{j_{k_1}}}{\|A_{j_{k_1}}\|_2} - \mu_k \frac{e_{j_{k_2}}}{\|A_{j_{k_2}}\|_2}.$$

In order to simplify the computation, we rewrite the iterative process in Algorithm 2.

The following Lemmas 3 and 4 will be used to prove the convergence of TRGS algorithm.

Lemma 3. *For the column vector u_k defined above, $\|u_k\|_2^2 = 1 - \mu_k^2$.*

Proof. By the definition of μ_k in (3.1) and u_k in (3.2), one can obtain that

$$\begin{aligned} \|u_k\|_2^2 = u_k^T u_k &= (\frac{A_{j_{k_1}}^T}{\|A_{j_{k_1}}\|_2} - \mu_k \frac{A_{j_{k_2}}^T}{\|A_{j_{k_2}}\|_2})(\frac{A_{j_{k_1}}}{\|A_{j_{k_1}}\|_2} - \mu_k \frac{A_{j_{k_2}}}{\|A_{j_{k_2}}\|_2}) \\ &= 1 - 2\mu_k^2 + \mu_k^2 = 1 - \mu_k^2. \end{aligned}$$

Algorithm 2 TRGS method

Input: $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m$ and$x_0 \in \mathbb{R}^n$

Output: the approximation solution x_k of (1.1)

1. **for** $k = 0, 1, \ldots$ do until termination criterion is satisfied
2. Set $r_k = b - Ax_k$
3. Select j_{k_1} and j_{k_2} with probability by (2.1)
4. Compute

$$\mu_k = \frac{A_{j_{k_1}}^T A_{j_{k_2}}}{\|A_{j_{k_1}}\|_2\|A_{j_{k_2}}\|_2}, \ r_{k_1} = \frac{A_{j_{k_1}}^T r_k}{\|A_{j_{k_1}}\|_2} \text{ and } r_{k_2} = \frac{A_{j_{k_2}}^T r_k}{\|A_{j_{k_2}}\|_2}$$

5. Updata

$$x_{k+1} = x_k + \frac{r_{k_1} - \mu_k r_{k_2}}{(1-|\mu_k|^2)\|A_{j_{k_1}}\|_2} e_{j_{k_1}} + \frac{r_{k_2} - \mu_k r_{k_1}}{(1-|\mu_k|^2)\|A_{j_{k_2}}\|_2} e_{j_{k_2}}$$

6. **end for**

Lemma 4. *Define $\alpha_{s,t}$ and $\beta_{s,t}$ as*

$$\alpha_{s,t} = \frac{\mu_k^2}{\|u_k\|_2} = \frac{\frac{|A_s^T A_t|^2}{\|A_s\|_2^2\|A_t\|_2^2}}{\sqrt{1 - \frac{|A_s^T A_t|^2}{\|A_s\|_2^2\|A_t\|_2^2}}} \quad \text{and} \quad \beta_{s,t} = \frac{\mu_k}{\|u_k\|_2} = \frac{\frac{A_s^T A_t}{\|A_s\|_2\|A_t\|_2}}{\sqrt{1 - \frac{|A_s^T A_t|^2}{\|A_s\|_2^2\|A_t\|_2^2}}},$$

then, the existence of $\gamma \in \mathbb{R}$ subject to $(|\alpha_{s,t}| - |\beta_{s,t}|)^2 \geq \gamma$.

Proof. It is known that $\delta = \min\limits_{s\neq t} \frac{|A_s^T A_t|}{\|A_s\|_2\|A_t\|_2}$ and $\Delta = \max\limits_{s\neq t} \frac{|A_s^T A_t|}{\|A_s\|_2\|A_t\|_2}$, Then,

$$(|\alpha_{s,t}| - |\beta_{s,t}|)^2 = \frac{\frac{|A_s^T A_t|^2}{\|A_s\|_2^2\|A_t\|_2^2}\left(\frac{|A_s^T A_t|}{\|A_s\|_2\|A_t\|_2} - 1\right)^2}{\left(1 - \frac{|A_s^T A_t|}{\|A_s\|_2\|A_t\|_2}\right)\left(1 + \frac{|A_s^T A_t|}{\|A_s\|_2\|A_t\|_2}\right)} \geq \min\left\{\frac{\delta^2(1-\delta)}{1+\delta}, \frac{\Delta^2(1-\Delta)}{1+\Delta}\right\} = \gamma.$$

When the coefficient matrix A in (1.1) is tall and narrow with full column rank, the following result gives the convergence of the TRGS algorithm.

Theorem 2. *Assume that the tall and narrow coefficient matrix $A \in R^{m\times n}$ has full column rank. Let $x_* = A^\dagger b$ be a solution of* (1.1)*. Given an initial guess $x_0 \in \mathbb{R}^n$, then the sequence $\{x_k\}_0^\infty$ generated by TRGS algorithm converges linearly to the x_* in expectation. Moreover, it satisfies*

$$E_k\|x_{k+1} - x_*\|_{A^T A}^2 \leq \left[(1 - \frac{\lambda_{min}(A^T A)}{\tau_{max}})(1 - \frac{\lambda_{min}(A^T A)}{\|A\|_F^2}) - \frac{\lambda_{min}(A^T A)\gamma\tau_{min}}{\|A\|_F^2\tau_{max}}\right] \cdot \|x_k - x_*\|_{A^T A}^2$$

and

$$E\|x_k - x_*\|_{A^T A}^2 \leq \left[(1 - \frac{\lambda_{min}(A^T A)}{\tau_{max}})(1 - \frac{\lambda_{min}(A^T A)}{\|A\|_F^2}) - \frac{\lambda_{min}(A^T A)\gamma\tau_{min}}{\|A\|_F^2\tau_{max}}\right]^k \cdot \|x_0 - x_*\|_{A^T A}^2,$$

where $\tau_{max} = \max\limits_{p\in[n]}\{\|A\|_F^2 - \|A_p\|_2^2\}$, $\tau_{min} = \min\limits_{q\in[n]}\{\|A\|_F^2 - \|A_q\|_2^2\}$ and $\gamma = \min\left\{\frac{\delta^2(1-\delta)}{1+\delta}, \frac{\Delta^2(1-\Delta)}{1+\Delta}\right\}$.

Proof. See Appendix 1.

Remark 1. *We remark that the upper bound of the convergence of RGS method from Lemma 1 is*

$$\Psi_{RGS} = 1 - \frac{\lambda_{min}(A^T A)}{\|A\|_F^2}.$$

From Theorem 1, we remark the upper bound of the convergence of RGS2 method is

$$\Psi_{RGS2} = \Big(1 - \frac{\lambda_{min}(A^T A)}{\tau_{max}}\Big)\Big(1 - \frac{\lambda_{min}(A^T A)}{\|A\|_F^2}\Big).$$

By Theorem 2, we can obtain an upper bound of the convergence of TRGS method

$$\Psi_{TRGS} = \Big(1 - \frac{\lambda_{min}(A^T A)}{\tau_{max}}\Big)\Big(1 - \frac{\lambda_{min}(A^T A)}{\|A\|_F^2}\Big) - \frac{\lambda_{min}(A^T A)\gamma\tau_{min}}{\|A\|_F^2\tau_{max}}.$$

At each iteration, the TRGS method uses two columns of the matrix, while RGS utilizes only one. To be fair, we compare the upper bound on the convergence factor of one iteration of the TRGS method with that of two iterations, instead of one iteration, of the RGS method. Note that $0 \leq \gamma < 1$ and $\tau_{max} \leq \|A\|_F^2$, we have $0 \leq \gamma\tau_{min}/\tau_{max} < 1$ then $\Psi_{TRGS} \leq \Psi_{RGS2} \leq \Psi_{RGS}^2 < \Psi_{RGS}$. Especially, when the column correlation coefficient δ or $\Delta = 0$, then $\gamma = 0$. The RGS2 and TRGS methods have the same convergence factor in the upper bound.

Remark 2. *If* $\|A_j\|_2^2,\ j = 1, \cdots, n$, *are precomputed, we discuss the computational cost of RGS, RGS2 and TRGS in each iteration step. The RGS costs* $2m + 2n + 1$ *flopping operations (flops), the RGS2 method needs* $4m + 4n + 2$ *flops, while the TRGS method requires* $6m + 4n + 11$ *flops.*

4. Numerical examples

In this section, we give several examples to show the efficiency of our TRGS method. We compare TRGS with RGS2 and RGS. In addition, randomized Kaczmarz (RK) in [18], randomized extended Kaczmarz (REK) in [2], partially randomized extended Kaczmarz (PREK) in [30], generalized two-subspace randomized Kaczmarz (GTRK) and two-subspace randomized extended Kaczmarz (TREK) in [29] , as other iterative methods, are considered for solving consistent or inconsistent linear systems. All experiments are carried out with the Matlab 2020b on a computer with 3.00 GHz processing unit and 16 GB RAM.

We measure the efficiency of TRGS and other methods by *the relative solution error*

$$\text{RSE} := \frac{\|x_k - x_*\|_2^2}{\|x_*\|_2^2}.$$

The initial vector is set as $x_0 = (0, 0, \cdots, 0)^T$ for all methods. When the set maximum number of iterations $k_{\max} = 10^6$ or RSE $< 10^{-6}$, we terminate the iteration process of each method. The '-' means that the number of iteration steps of the algorithm reaches $k_{\max}$.

Example 4.1 In this example, for conherent matrix $A \in \mathbb{R}^{2000\times 100}$ in least-squares problem (1.1). The entries of the A are the independent identically distributed uniform random variables in the interval $(t, 1)$, where the $t \in [0, 1]$. We remark the average column correlation index as follows

$$\bar{\mu}_k = \frac{2}{n^2 + n} \sum_{q=1;q>p}^{n} \frac{|A_p^T A_q|}{\|A_p\|_2\|A_q\|_2}, \quad p, q \in \{1, 2, \cdots, n\}.$$

When t increases from 0 to 1, the change of the $\bar{\mu}_k$ with t and the relationship between Ψ_{RGS},Ψ^2_{RGS}, Ψ_{RGS2} and Ψ_{TRGS} versus t are plotted in Figure 1.

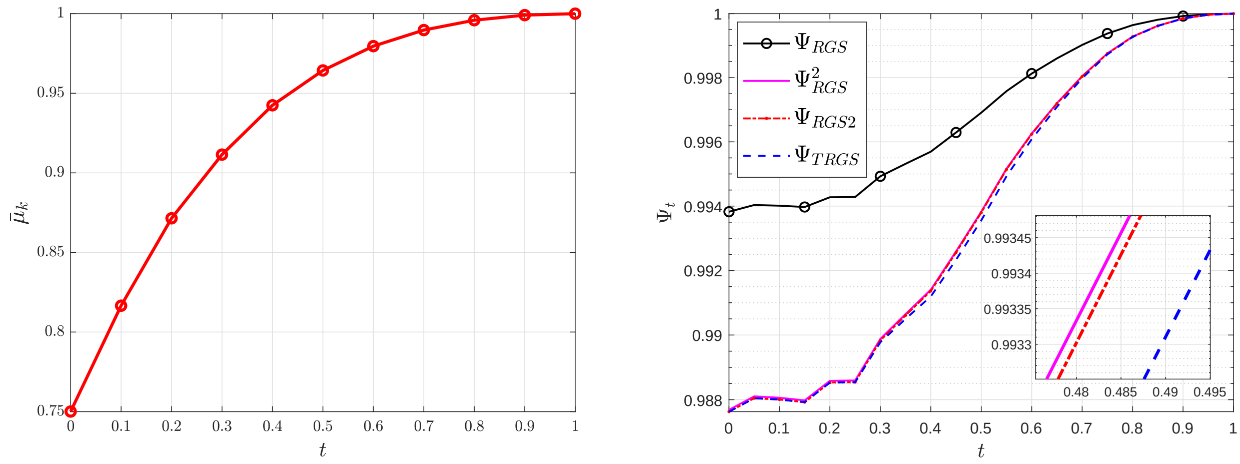


Figure 1. Relationship between t and $\bar{\mu}_k$ (left). Comparison of Ψ_{RGS}, Ψ^2_{RGS},Ψ_{RGS2} and Ψ_{TRGS} with t (right).

As can be seen from Figure 1, the left subfigure shows that as t approaches 1, the average column correlation index $\bar{\mu}_k$ is highly correlated. From the right subfigure, the convergence upper bound of RGS2 and TRGS are always lower than RGS, and further we find that the theoretical upper bound of convergence of the TRGS will not exceed the RGS2.

Example 4.2 We use the RGS2 and TRGS methods to test the consistent least squares problem (1.1) with different size and compare them with the RK, RGS and the GTRK methods. The size of A is $m \times n$ with $m = 10^3 * k$ $(k = 1, 2, \cdots, 5)$ and $n = 50$. The entries of the matrix A are the independent identically distributed uniform random variables in the interval $(t, 1)$. The vector $b = Ax_*$, where x_* is generated randomly with the MATLAB function *randn*.

Table 1. IT, CPU of all methods for the consistent system with $n = 50$ and different m.

name	t	$m \times n$	1000×50	2000×50	3000×50	4000×50	5000×50
	0.1	*Cond(A)*	18.90	17.69	16.83	16.94	16.50
RK		IT	2742	2532	2461	2274	2282
		CPU(s)	6.233e-02	8.149e-02	1.041e-01	1.650e-01	1.938e-01
RGS		IT	2765	2252	2538	2259	2399
		CPU(s)	6.663e-02	7.995e-02	1.170e-01	1.297e-01	1.574e-01
RGS2		IT	1390	1132	1083	1201	1087
		CPU(s)	6.309e-02	7.624e-02	9.961e-02	1.351e-01	1.406e-01
GTRK		IT	625	587	577	568	567
		CPU(s)	2.931e-02	4.188e-02	5.464e-02	8.949e-02	1.063e-01
TRGS		IT	483	539	533	486	466
		CPU(s)	1.809e-02	2.678e-02	3.354e-02	3.921e-02	4.725e-02

In Tables 1–3, we list the IT and the CPU(s) for the RK, RGS, RGS2, GTRK, and TRGS methods with $t = 0.1, 0.5$ and 0.8, and the Euclidean condition number *Cond(A)* of the matrix is reported in each table. Figure 2 shows the plots of m versus IT (left), and m versus CPU (right) of Algorithm 2 applied to solve (1.1) with different coefficient matrix A listed in Tables 1–3, respectively.

Table 2. IT, CPU of all methods for the consistent system with $n = 50$ and different m.

name	t	$m \times n$	1000×50	2000×50	3000×50	4000×50	5000×50
	0.5	*Cond(A)*	47.91	43.13	41.39	41.19	40.64
RK		IT	13796	11153	11488	10563	10431
		CPU(s)	3.017e-01	3.586e-01	5.073e-01	7.467e-01	9.254e-01
RGS		IT	14074	11362	11162	10375	10714
		CPU(s)	3.292e-01	3.956e-01	5.033e-01	5.988e-01	7.219e-01
RGS2		IT	6791	5733	5622	5474	5337
		CPU(s)	3.050e-01	3.880e-01	5.036e-01	6.234e-01	7.082e-01
GTRK		IT	726	611	703	713	687
		CPU(s)	3.501e-02	4.567e-02	7.049e-02	1.151e-01	1.303e-01
TRGS		IT	636	592	677	611	642
		CPU(s)	2.53e-02	2.939e-02	4.328e-02	4.938e-02	6.810e-02

Table 3. IT, CPU of all methods for m-by-n consistent system with $n = 50$ and different m.

name	t	$m \times n$	1000×50	2000×50	3000×50	4000×50	5000×50
	0.8	*Cond(A)*	141.61	128.37	124.61	122.65	121.99
RK		IT	100928	96807	87951	90652	89026
		CPU(s)	2.187e+00	3.089e+00	3.992e+00	6.870e+00	7.643e+00
RGS		IT	116846	103915	89490	89978	87764
		CPU(s)	2.690e+00	3.755e+00	4.087e+00	5.181e+00	5.902e+00
RGS2		IT	60650	50882	46398	44669	44784
		CPU(s)	2.654e+00	3.428e+00	4.155e+00	5.091e+00	5.922e+00
GTRK		IT	738	717	697	736	709
		CPU(s)	3.463e-02	5.124e-02	6.969e-02	1.203e-01	1.356e-01
TRGS		IT	696	665	658	658	683
		CPU(s)	2.585e-02	3.253e-02	4.066e-02	5.334e-02	8.527e-02

From Tables 1–3, we can see that the TRGS method is better than other algorithms based on IT and CPU(s). We find that GTRK and TRGS are basically stable in both IT and CPU(s), while RK, RGS and RGS2 methods need more iterations and CPU time.

From Figure 2, it is not difficult to see that the curve of TRGS is much lower than that of RGS2 in terms of the IT and the CPU(s). In addition, RGS2 is sensitive to t, while TRGS is not affected by it. For example, in Figure 2, fix $m = 3000$, when $t = 0.1,\ 0.5$ and 0.8, the IT of TRGS are basically steady at the level about 10^7, while the IT of RGS2 are steady at the level 10^8, 10^{10} and 10^{12}, respectively. Similar results also appear in the CPU(s).

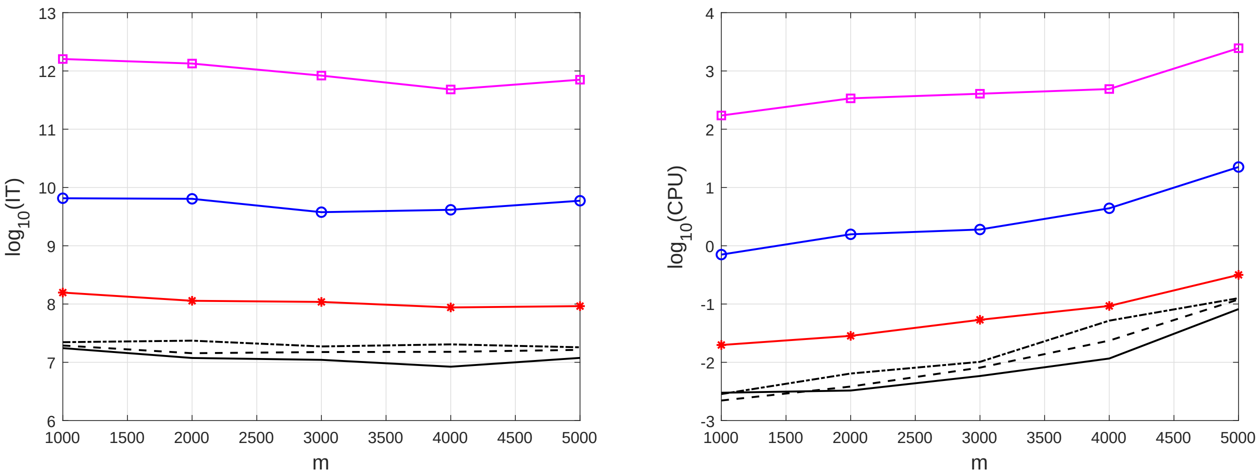


Figure 2. Pictures of $log_{10}(IT)$ (left) and $log_{10}(CPU)$ (right) versus for RGS2 and TRGS for consistent system when $n = 100$. RGS2 for $t = 0.1$:"- * -", RGS2 for $t = 0.5$: "- o -", RGS2 for $t = 0.8$:"−□−", TRGS for $t = 0.1$: "-", TRGS for $t = 0.5$: "- -" and TRGS for $t = 0.8$: "-.-.-".

Table 4. IT, CPU of all methods for the inconsistent system with $n = 100$ and different m.

name	t	$m \times n$	1000×100	2000×100	3000×100	4000×100	5000×100
	0.1	*Cond(A)*	30.04	30.00	25.73	24.98	24.74
REK		IT	8246	7360	6383	6319	6135
		CPU(s)	3.954e-01	5.279e-01	6.529e-01	1.080e-01	1.579e-01
RGS		IT	6676	5821	5306	4710	4705
		CPU(s)	2.334e-01	3.305e-01	4.174e-01	5.378e-01	8.937e-01
PREK		IT	-	-	-	904195	857303
		CPU(s)	-	-	-	1.385e+02	2.075e+02
RGS2		IT	3268	2914	2581	2383	2187
		CPU(s)	2.224e-01	3.257e-01	3.954e-01	5.237e-01	7.610e-01
TREK		IT	1641	1534	1439	1401	1486
		CPU(s)	1.120e-01	1.410e-01	1.662e-01	2.582e-01	3.536e-01
TRGS		IT	1402	1253	1201	1148	1118
		CPU(s)	6.810e-02	8.920e-02	1.176e-01	1.738e-01	2.846e-01

Table 5. IT, CPU of all methods for the inconsistent system with n = 100 and different m.

name	t	$m \times n$	1000×100	2000×100	3000×100	4000×100	5000×100
	0.5	*Cond(A)*	74.36	66.12	63.36	60.93	60.49
REK		IT	45941	33128	31918	33464	29270
		CPU(s)	2.207e+00	2.187e+00	2.993e+00	5.266e+00	7.963e+00
RGS		IT	38943	24655	23342	24497	22516
		CPU(s)	1.346e+00	1.332e+00	1.761e+00	2.605e+00	4.211e+00
PREK		IT	-	-	-	-	-
		CPU(s)	-	-	-	-	-
RGS2		IT	18370	12110	11231	12188	11255
		CPU(s)	1.230e+00	1.272e+00	1.668e+00	2.567e+00	3.854e+00
TREK		IT	1896	1636	1714	1677	1617
		CPU(s)	1.282e-01	1.501e-01	2.360e-01	2.885e-01	4.155e-01
TRGS		IT	1607	1367	1381	1217	1396
		CPU(s)	7.700e-02	1.022e-01	1.370e-01	1.647e-01	3.837e-01

Table 6. IT, CPU of all methods for the inconsistent system with n = 100 and different m.

name	t	$m \times n$	1000×100	2000×100	3000×100	4000×100	5000×100
	0.8	*Cond(A)*	223.66	197.26	188.00	183.47	178.91
REK		IT	405922	302370	277601	298566	268318
		CPU(s)	1.818e+01	2.056e+01	2.700e+01	5.005e+01	7.072e+01
RGS		IT	283262	226035	203500	208911	196200
		CPU(s)	9.472e+00	1.211e+01	1.571e+01	2.304e+01	3.554e+01
PREK		IT	-	-	-	-	-
		CPU(s)	-	-	-	-	-
RGS2		IT	144031	111424	103317	103665	100780
		CPU(s)	9.268e+00	1.187e+01	1.565e+01	2.218e+01	3.378e+01
TREK		IT	1888	1684	1613	1830	1804
		CPU(s)	1.206e-01	1.477e-01	1.874e-01	3.337e-01	4.297e-01
TRGS		IT	1725	1341	1207	1462	1340
		CPU(s)	8.063e-02	9.848e-02	1.122e-01	2.184e-01	3.487e-01

Example 4.3 In this example, we apply the RGS2 and TRGS methods to solve the inconsistent least squares problem (1.1) and compare them with the REK, RGS, PREK and TREK methods. The entries of the matrix A are the independent identically distributed uniform random variables in the interval

$(t, 1)$, and the vector $b = Ax_* + r$, where x_* is one of the solutions of (1.1), which is generated randomly with the MATLAB function *randn*, and r is a nonzero vector in the null space of A^T, which is generated by the MATLAB function *null*. The size of A is $m \times n$ with $m = 10^3 * k$ $(k = 1, 2, \cdots, 5)$, and $n = 100$.

Tables 4–6 list the iteration number (IT) and the CPU time when all methods stop and we also set $t = 0.1, 0.5, 0.8$ in each case. Figure 3 shows the plots of m versus IT (left) and m versus CPU (right) of TRGS and RGS2 applied to solve all linear systems (1.1) in Tables 4–6.

From Tables 4–6, we can see that the TRGS method is better than other algorithms based on IT and CPU(s). We also find that TREK and TRGS are basically stable in both IT and CPU(s), while the REK, RGS, PREK and RGS2 need more iterations and CPU time.

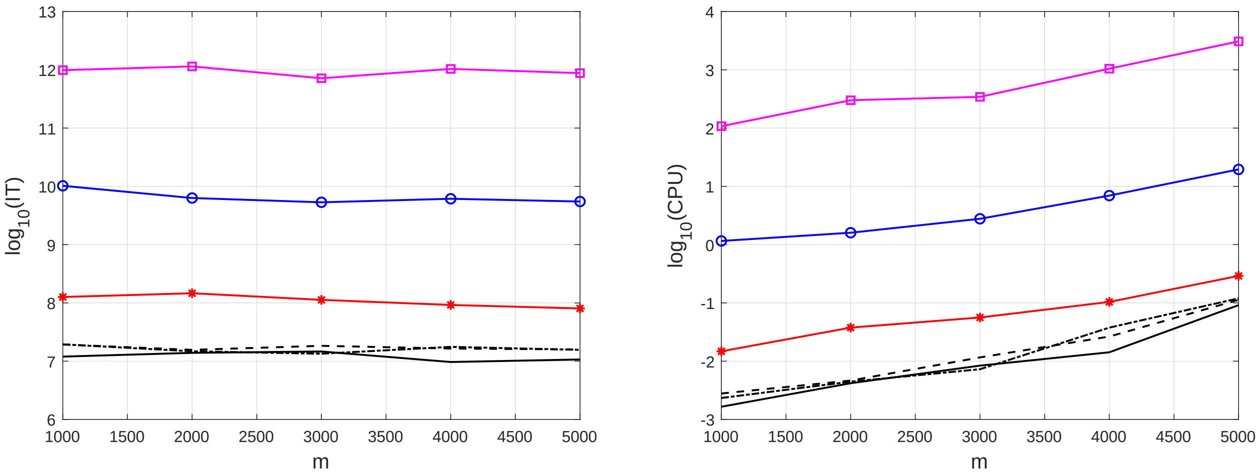


Figure 3. Pictures of $log_{10}(IT)$ (left) and $log_{10}(CPU)$ (right) versus for RGS2 and TRGS for inconsistent system when $n = 100$. RGS2 for $t = 0.1$:"- * -", RGS2 for $t = 0.5$: "- o -", RGS2 for $t = 0.8$:"–□–", TRGS for $t = 0.1$: "-", TRGS for$t = 0.5$: "- -" and TRGS for $t = 0.8$: "-.-.-".

From Figure 3, the curves of TRGS and RGS2 show that RGS2 needs more iterations and CPU time to reach the stopping criterion. In addition, RGS2 is sensitive to t, while TRGS is not. For example, in Figure 3, fix $m = 3000$, when $t = 0.1,\ 0.5$ and 0.8, the IT of TRGS are basically steady at the level about 10^7, while the IT of RGS2 are steady at the level about 10^8, 10^{10} and 10^{12} respectively. Similar results also appear in the CPU(s).

Example 4.4 In this example, we apply the TRGS method to solve the least squares problem (1.1) with the sparse coefficient matrix A taken from the Florida sparse matrix collection in [31]. Especially, we select the tall and narrow sparse matrix A with full column rank. Table 7 summarizes different sparse systems with *density* and condition number $Cond(A)$, where the *density* of a matrix A means the ratio of the number of the nonzero elements of A to the total number of the elements of A. Algorithm 2 is compared with the RK, RGS, GTRK, REK, PREK, TREK and RGS2 methods.

When the sparse least squares problem (1.1) is set to be consistent, the vector $b = Ax_*$, where x_*, one of the solutions of learst squares problem (1.1), is generated randomly with the MATLAB function *randn*. Table 8 lists the iteration number (IT) and the CPU time when RK, RGS, RGS2, GTRK and TRGS methods stop. Figure 4 shows the plot of RSE versus IT (left) and RSE versus CPU (right) of

Table 7. The properties of different matrices from the Florida sparse matrix collection in [31].

name	*abtaha2*	*divorce*	*Cites*	*bibd-81-3*^{T}	*WorldCities*
$m \times n$	37932×331	50×9	55×46	85320×3240	315×100
density	1.09%	50.00%	53.04%	0.09%	23.87%
Cond(A)	12.22	19.39	207.15	1.75	66.00

Algorithm 2 applied to solve (1.1) with the sparse coefficient matrix A named "*divorce*".

Table 8. IT, CPU of all methods for m-by-n consistent system.

name		*abtaha2*	*divorce*	*Cites*	*bibd-81-3*^{T}	*WorldCities*
RK	IT	150180	2244	320707	45633	37064
	CPU(s)	9.028e+01	3.642e-02	5.686e+00	6.894e+01	9.448e-01
RGS	IT	137190	2978	286699	35515	39585
	CPU(s)	2.913e+01	3.566e-02	3.779e+00	1.550e+01	8.348e-01
RGS2	IT	76949	1340	145180	17935	20637
	CPU(s)	3.215e+01	2.958e-02	3.633e+00	1.477e+01	8.382e-01
GTRK	IT	78808	877	75523	22727	13788
	CPU(s)	7.695e+01	3.540e-02	3.232e+00	6.220e+01	8.000e-01
TRGS	IT	64956	139	56142	18351	11306
	CPU(s)	1.618e+01	4.470e-03	1.811e+00	9.819e+00	4.862e-01

When the sparse least squares problem (1.1) is set to be inconsistent, the $b = Ax_* + r$, where the r is a nonzero vector in the null space of A^T. Due to the large dimension of A, the r cannot be generated by the Matlab function *null*, but it can be generated by the projection vector $\check{r}$. The $\check{r}$ is generated by the MATLAB function *randn* and projected onto the null space of A^T. That is to say, $r = \check{r} - AA^\dagger\check{r}$, where $A^\dagger\check{r}$ is obtained by the Matlab function *lsqminnorm*. Table 9 lists the IT and the CPU(s) when REK, RGS, PREK, RGS2, TREK and TRGS methods stop. Figure 5 shows the plot of RSE versus IT (left) and RSE versus CPU (right) of Algorithm 2 applied to solve (1.1) with the sparse coefficient matrix A named "*divorce*".

It can be seen from Table 8 that for sparse matrices listed in Table 7, TRGS achieves fast convergence with less IT and CPU(s) than that of RK, RGS, GTRK and RGS2 do. Similar results are shown in Table 9. Furthermore, from Figure 4, TRGS reaches the stop criteria with less iterations (left) and CPU time (right) than other methods for the *"divorce"* consistent sparse least squares problem (1.1). Similar results also appear in Figure 5.

Example 4.5 This example uses Algorithm 2 to restore a computer tomography (CT) image.

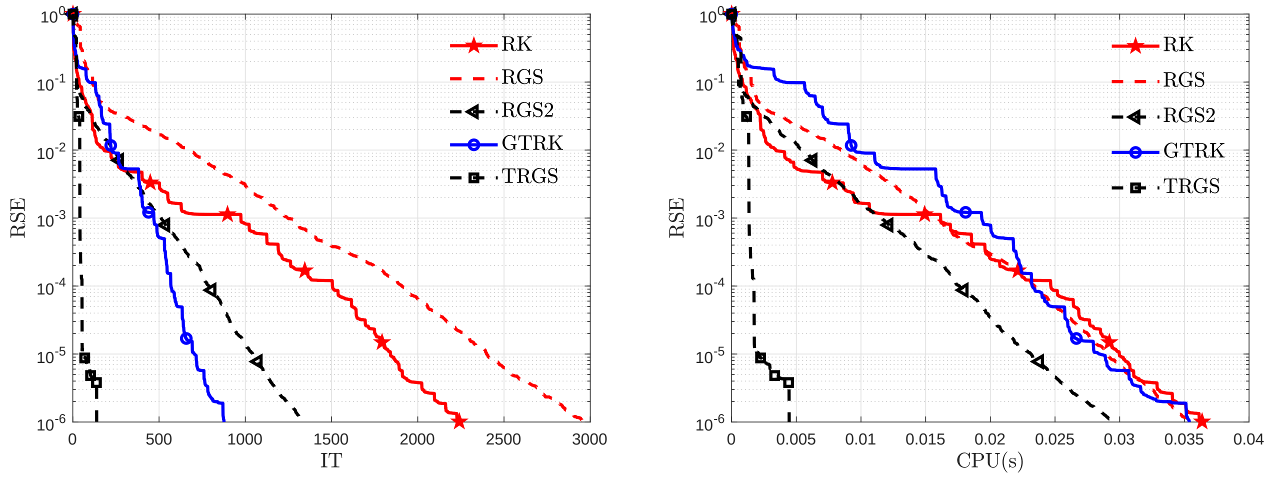


Figure 4. The RK, RGS, RGS2, GTRK and TRGS methods for solving linear consistent systems named *divorce*. Left: the relationship between IT and RSE; Right: the relationship between CPU (s) and RSE.

Table 9. IT, CPU of all methods for *m*-by-*n* inconsistent system.

name		*abtaha2*	*divorce*	*Cites*	*bibd-81-3^T*	*WorldCities*
REK	IT	194480	4450	403533	52384	58586
	CPU(s)	1.258e+02	1.636e-01	1.496e+01	9.001e+01	2.946e+00
RGS	IT	152939	3552	307945	35266	39632
	CPU(s)	3.198e+01	4.360e-02	4.202e+00	1.552e+01	8.494e-01
PREK	IT	166019	3890	347192	47165	65670
	CPU(s)	1.048e+02	9.710e-02	9.445e+00	7.933e+01	2.480e+00
RGS2	IT	53481	1548	156941	18004	20030
	CPU(s)	2.210e+01	3.790e-02	4.086e+00	1.522e+01	8.309e-01
TREK	IT	88463	1075	103915	26109	20569
	CPU(s)	8.522e+01	9.452e-02	9.175e+00	7.641e+01	2.209e+00
TRGS	IT	77953	94	98361	18306	15423
	CPU(s)	1.957e+01	4.500e-03	3.260e+00	9.950e+00	6.633e-01

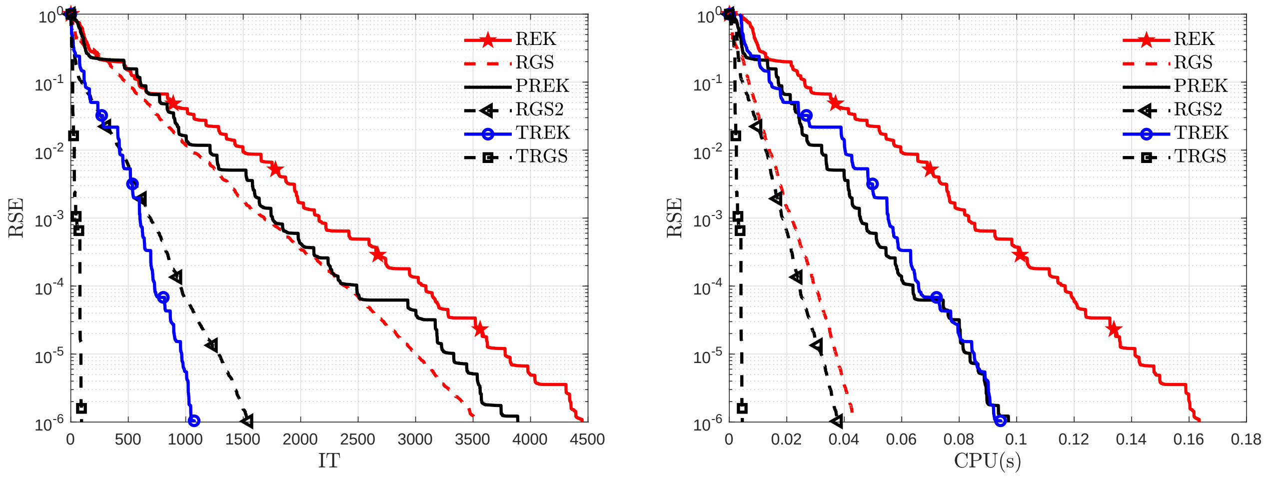


Figure 5. The REK, RGS, PREK, RGS2, TREK and TRGS methods for solving linear inconsistent systems named *divorce*. Left: the relationship between IT and RSE; Right: the relationship between CPU (s) and RSE.

We use the MATLAB function $paralleltomo(N, \theta, p)$ from Algebraic Iterative Reconstruction (ART) package in [31] to generate a large sparse matrix A and the exact solution x_*, where $N = 35$, $\theta = 0^\circ : 1.5^\circ : 178^\circ$ and $p = 50$, then the size of A is 5950×1225 and the condition number $Cond(A) = 352.32$. We compute $\hat{b}$ by $\hat{b} = Ax_*$ and $b = \hat{b} + r$, where the noise r is from the null space of the coefficient matrix A^T, i.e., $A^T r = 0$. We set ordinary Gaussian white noise with noise levels $\delta = 0.01$ and the maximum iterative number is $5 * 10^6$. The TRGS is used to recover x_* from b and compared with the REK, RGS, PREK, RGS2 and TREK methods.

Figure 6 shows the recovered images by REK, RGS, PREK, RGS2, TREK and TRGS together with the original image and noised image with $\delta = 0.01$. Figure 7 shows the convergence of RSE versus IT (left) and RSE versus CPU(s) (right) of TRGS compared with other methods when $\delta = 0.01$.

It is shown from Figure 6 that all methods obtain well restored image, and Figure 7 shows that TRGS converges much faster than REK, RGS, PREK, RGS2 and TREK do when $\delta = 0.01$.

Example 4.6 In this example, we use Algorithm 2 to solve the famous *Phillips* ill-posed problem in [32], which comes from the Fredholm integral equation of first kind

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

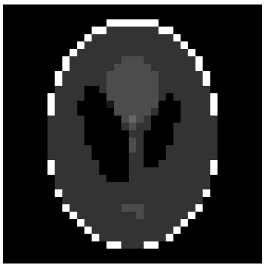
on the square $[-6, 6] \times [-6, 6]$, where the kernel function is presented by $K(s,t) = \phi(s-t)$ with

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

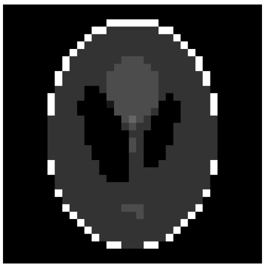
and the right-hand side

$$f(s) = (6 - |s|)(1 + \frac{1}{2}cos(\frac{s\pi}{3})) + \frac{9}{2\pi}sin(\frac{|s|\pi}{3}).$$

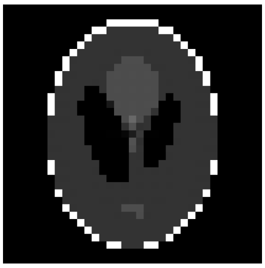
The above problem is discretized to obtain the linear systems (1.1), where the coefficient matrix $A \in \mathbb{R}^{n\times n}$, exact solution vector $x_* \in \mathbb{R}^n$ and column vector $b \in \mathbb{R}^n$ are all generated by *MATLAB* function



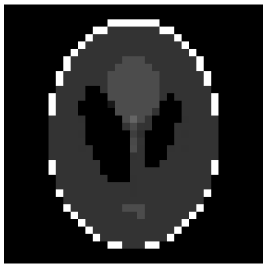
(a) the original image



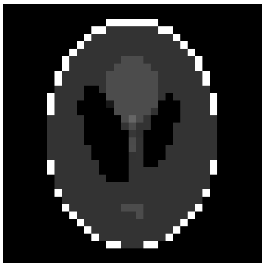
(b) The noised image



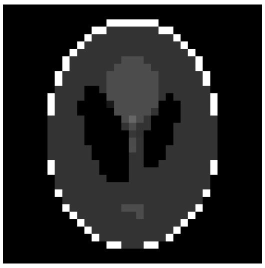
(c) REK



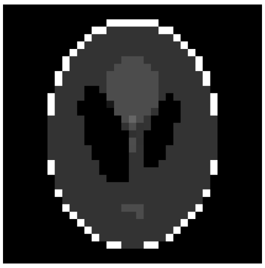
(d) RGS



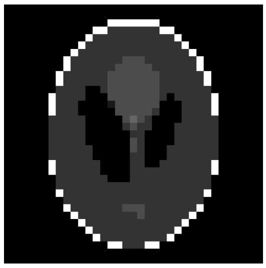
(e) PREK



(f) RGS2

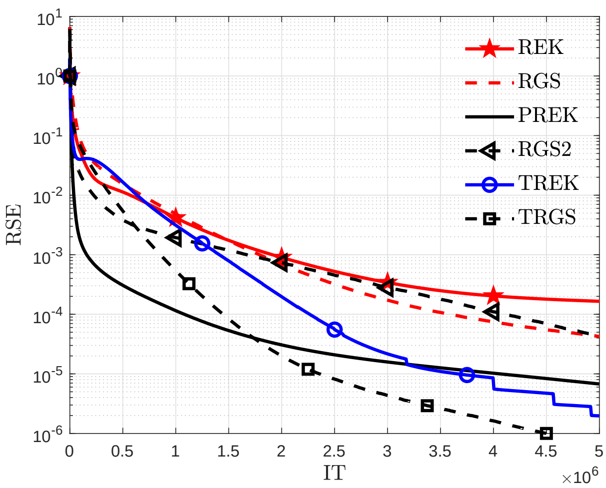


(g) TREK

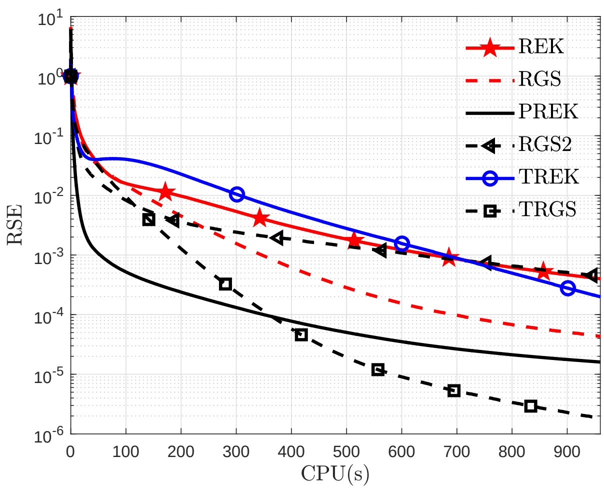


(h) TRGS

Figure 6. The original ”*phantom*” image (a), the noised image (b), the recovered images by REK (c), RGS (d), PREK (e), RGS2 (f), TREK (g) and TRGS (h).

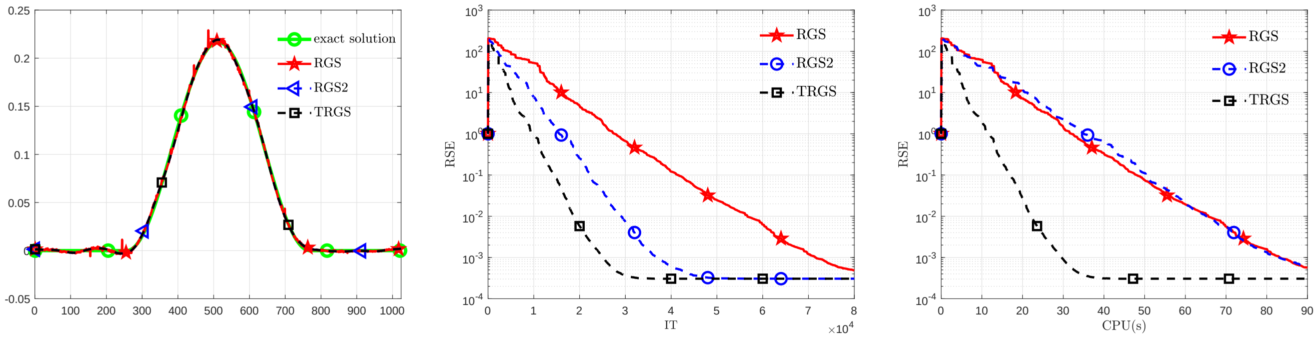


(a)



(b)

Figure 7. Convergence of RSE versus IT (a) and RSE versus CPU (b) of TRGS compared with that of RGS2 and TREK for restoring the noised ”*phantom*” image with $\delta = 0.01$.



(a) (b) (c)

Figure 8. The performance of RGS, RGS2 and TRGS for the nosied *Phillips* test problem.

phillips(n) in [32]. As we know, if the system matrix A of (1.1) is ill-conditioned, one way to solve this problem is to use the Tikhonov regularization, that is

$$\min_{x\in\mathbb{R}^n}\left\{\|Ax-b\|_2^2+\lambda\|x\|_2^2\right\}, \text{ with } \lambda>0.$$

We set $n = 1024$, $\lambda = 0.01$ and Gaussian white noise level denoted by $\delta = \|r\|_2/\|b\|_2 = 1\%$ to obtain $b = Ax_* + r$. Here the condition number $Cond(A) = 4.1618e + 10$ and the rank is 1024, which means (1.1) is a several linear ill-posed problem. It is worth noting that the system matrix satisfies the conditions of Lemma 1, Theorems 1 and 2, so the above methods will converge. In Figure 8, the (a) displays the approximation solution derived by RGS, RGS2 and TRGS together with the exact solution for the *Phillips* test problem when $\delta = 0.01$. The (b) and (c) show the convergence of RSE versus IT and RSE versus CPU(s) of TRGS compared with the RGS and RGS2 methods when $\delta = 0.01$.

We can see from Figure 8 that all the recovered solution by RGS, RGS2 and TRGS are close to the exact solution in (a), (b) and (c) show that TRGS converges much faster than the other two methods.

5. Conclusions

A two-step randomized Guass-Seidel (TRGS) method for solving the linear least squares problems with tall and narrow coefficient matrix was presented. And the convergence analysis is provided when the coefficient matrix of (1.1) is of full column rank. This method does not need any columns pavings. Numerical examples for different cases show the superiority of the current method (TRGS) in this paper.

Acknowledgments

This work was supported by the Project of Department of Science and Technology of Sichuan Provincial (No. 2021ZYD0005), the Opening Project of Key Laboratory of Higher Education of Sichuan Province for Enterprise Informationalization and Internet of Things (No. 2020WZJ01), the Opening Project of Bridge Non-destruction Detecting and Engineering Computing Key Laboratory of Sichuan (Nos. 2021QYJ07, 2020QZJ03), the Scientific Research Project of Sichuan University of Science and Engineering (No. 2020RC24), and the Graduate student Innovation Fund (No. y2021101).

Conflict of interest

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

References

1. A. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996. https://doi.org/10.1137/1.9781611971484
2. A. Zouzias, N. M. Freris, Randomized extended Kaczmarz for solving least squares, *SIAM J. Matrix Anal. Appl.*, **34** (2013), 773–793. https://doi.org/10.1137/120889897
3. R. M. Gower, P. Richtarik, Randomized iterative methods for linear systems, *SIAM J. Matrix Anal. Appl.*, **36** (2015), 1660-1690. https://doi.org/10.1137/15M1025487
4. A. Ma, D. Needell, A. Ramdas, Convergence properties of the randomized extended Gauss-Seidel and Kaczmarz methods, *SIAM J. Matrix Anal. Appl.*, **36** (2015), 1590–1604. https://doi.org/10.1137/15M1014425
5. Z. Z. Bai, W. T. Wu, On greedy randomized Kaczmarz method for solving large sparse linear systems, *SIAM J. Sci. Comput.*, **40** (2018), A592–A606. https://doi.org/10.1137/17M1137747
6. A. Ma, D. Needell, A. Ramdas, Iterative methods for solving factorized linear systems, *SIAM J. Matrix Anal. Appl.*, **39** (2018), 104–122. https://doi.org/10.1137/17M1115678
7. C. Byrne, A unified treatment of some iterative algorithms in signal processing and image reconstruction, *Inverse Prob.*, **20** (2004), 103–120. https://doi.org/10.1088/0266-5611/20/1/006
8. C. A. Bouman, K. Sauer, A unified approach to statistical tomography using coordinate descent optimization, *IEEE Trans. Image Process.*, **5** (1996), 480–492. https://doi.org/10.1109/83.491321
9. J. C. Ye, K. J. Webb, C. A. Bouman, R. P. Millane, Optical diffusion tomography by iterative-coordinate-descent optimization in a Bayesian framework, *J. Opt. Soc. Am. A.*, **16** (1999), 2400–2412. https://doi.org/10.1364/JOSAA.16.002400
10. A. A. Canutescu, R. L. Dunbrack, Cyclic coordinate descent: A robotics algorithm for protein loop closure, *Protein Sci.*, **12** (2003), 963–972. https://doi.org/10.1110/ps.0242703
11. K. W. Chang, C. J. Hsieh, C. J. Lin, Coordinate descent method for large-scale L2-loss linear support vector machines, *J. Mach. Learn. Res.*, **9** (2008), 1369–1398. https://doi.org/10.1145/1390681.1442778
12. P. Breheny, J. Huang, Coordinate descent algorithms for nonconvex penalized regression with applications to biological feature selection, *Ann. Appl. Stat.*, **5** (2011), 232–253. https://doi.org/10.1214/10-AOAS388
13. M. Elad, B. Matalon, M. Zibulevsky, Coordinate and subspace optimization methods for linear least squares with non-quadratic regularization, *Appl. Comput. Harmonic Anal.*, **23** (2007), 346–367. https://doi.org/10.1016/j.acha.2007.02.002
14. C. Wang, D. Wu, K. Yang, New decentralized positioning schemes for wireless sensor networks based on recursive least-squares optimization, *IEEE Wirel. Commun. Lett.*, **3** (2014), 78–81. https://doi.org/10.1109/WCL.2013.111713.130734

15. J. A. Scott, M. Tuma, Sparse stretching for solving sparse-dense linear least-squares problems, *SIAM J. Sci. Comput.*, **41** (2019), A1604–A1625. https://doi.org/10.1137/18M1181353
16. N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, 2002. https://doi.org/10.2307/2669725
17. Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, 2003. https://doi.org/10.1137/1.9780898718003.ch4
18. T. Strohmer, R. Vershynin, A randomized Kaczmarz algorithm with exponential convergence, *J. Fourier Anal. Appl.*, **15** (2009), 262–278. https://doi.org/10.1007/s00041-008-9030-4
19. D. Leventhal, A. S. Lewis, Randomized methods for linear constraints: Convergence rates and conditioning, *Math. Oper. Res.*, **35** (2010), 641–654. https://arxiv.org/abs/0806.3015
20. Z. S. Lu, L. Xiao, On the complexity analysis of randomized block-coordinate descent methods, *J. Math. Program*, **152** (2015), 615–642. https://doi.org/10.1007/s10107-014-0800-2
21. Y. Liu, X. L. Jiang, C. Q. Gu, On maximum residual block and two-step Gauss-Seidel algorithms for linear least-squares problems, *Calcolo*, **58** (2021), 1–32. https://doi.org/10.1007/s10092-021-00404-x
22. Z. Z. Bai, W. T. Wu, On greedy randomized coordinate descent methods for solving large linear least-squares problems, *Numer. Linear Algebra Appl.*, **26** (2019), 22–37. https://doi.org/10.1002/nla.2237
23. J. H. Zhang, J. H. Guo, On relaxed greedy randomized coordinate descent methods for solving large linear least-squares problems, *J. Appl. Numer. Math.*, **157** (2020), 372–384. https://doi.org/10.1016/j.apnum.2020.06.014
24. Z. Z. Bai, L. Wang, W. T. Wu, On convergence rate of the randomized Gauss-Seidel method, *Linear Algebra Appl.*, **611** (2021), 237–252. https://doi.org/10.1016/j.laa.2020.10.028
25. J. Nutini, M. Schmidt, I. H. Laradji, M. Friedlander, H. Koepke, Coordinate descent converges faster with the Gauss-Southwell rule than random selection, *Int. J. Technol. Manage.*, **43** (2015), 1632-1641. https://doi.org/10.1504/IJTM.2008.019410
26. K. Du, Tight upper bounds for the convergence of the randomized extended Kaczmarz and Gauss-Seidel algorithms, *Numer. Linear Algebra Appl.*, **26** (2019), 22–33. https://doi.org/10.1002/nla.2233
27. W. Wu, *Paving the Randomized Gauss-Seidel Method*, BSc Thesis, Scripps College, Claremont, California, 2017. https://scholarship.claremont.edu/scripps_theses/1074
28. D. Needell, R. Ward, Two-subspace projection method for coherent overdetermined systems, *J. J. Fourier Anal. Appl.*, **19** (2013), 256–269. https://doi.org/10.1007/s00041-012-9248-z
29. W. T. Wu, On two-subspace randomized extended Kaczmarz method for solving large linear least-squares problems, *Numer. Algor.*, **89** (2022), 1–31. https://doi.org/10.1007/s11075-021-01104-x
30. Z. Z. Bai, W. T. Wu, On partially randomized extended Kaczmarz method for solving large sparse overdetermined inconsistent linear systems, *Linear Algebra Appl.*, **578** (2019), 225–250. https://doi.org/10.1016/j.laa.2019.05.005
31. T. A. Davis, Y. Hu, The university of florida sparse matrix collection, *ACM Trans. Math. Software*, **38** (2011), 1–25. https://doi.org/10.1145/2049662.2049663

32. P. C. Hansen, Regularization Tools: A Matlab package for analysis and solution of discrete ill-posed problems, *Numer. Algor.*, **06** (1994), 1–35. https://doi.org/10.1007/BF02149761

Appendix

Proof of Theorem 2. By the definition of TRGS, one can obtain that

$$\begin{aligned} x_{k+1} &= y_k + \left(\frac{A_{j_{k_1}}^T (b - Ay_k)}{\|A_{j_{k_1}}\|_2} - \mu_k \frac{A_{j_{k_2}}^T (b - Ay_k)}{\|A_{j_{k_2}}\|_2} \right) \frac{\frac{e_{j_{k_1}}}{\|A_{j_{k_1}}\|_2} - \mu_k \frac{e_{j_{k_1}}}{\|A_{j_{k_2}}\|_2}}{\|u_k\|_2^2} \\ &= x_k + \frac{A_{j_{k_2}}^T (b - Ax_k)}{\|A_{j_{k_2}}\|_2} e_{j_{k_2}} + \left(\frac{A_{j_{k_1}}^T (b - Ay_k)}{\|A_{j_{k_1}}\|_2} - \mu_k \frac{A_{j_{k_2}}^T (b - Ay_k)}{\|A_{j_{k_2}}\|_2} \right) \cdot \frac{\frac{e_{j_{k_1}}}{\|A_{j_{k_1}}\|_2} - \mu_k \frac{e_{j_{k_1}}}{\|A_{j_{k_2}}\|_2}}{\|u_k\|_2^2}. \end{aligned}$$

Then,

$$A(x_{k+1} - x_k) = \frac{A_{j_{k_2}}^T (b - Ax_k)}{\|A_{j_{k_2}}\|_2} A_{j_{k_2}} + \left(\frac{A_{j_{k_1}}^T (b - Ay_k)}{\|A_{j_{k_1}}\|_2} - \mu_k \frac{A_{j_{k_2}}^T (b - Ay_k)}{\|A_{j_{k_2}}\|_2} \right) \cdot \frac{u_k}{\|u_k\|_2^2}. \tag{A1}$$

The following will estimate $\frac{A_{j_{k_1}}^T (b-Ay_k)}{\|A_{j_{k_1}}\|_2}$ and $\frac{A_{j_{k_2}}^T (b-Ay_k)}{\|A_{j_{k_2}}\|_2}$,

(i)

$$\begin{aligned} A_{j_{k_1}}^T (b - Ay_k) &= A_{j_{k_1}}^T \left(b - A \left(x_k + \frac{A_{j_{k_2}}^T (b - Ax_k)}{\|A_{j_{k_2}}\|_2} e_{j_{k_2}} \right) \right) \\ &= A_{j_{k_1}}^T (b - Ax_k) - \frac{A_{j_{k_1}}^T A_{j_{k_2}} A_{j_{k_2}}^T (b - Ax_k)}{\|A_{j_{k_2}}\|_2^2} \\ &= A_{j_{k_1}}^T (b - Ax_k) - \mu_k \frac{A_{j_{k_2}}^T (b - Ax_k)}{\|A_{j_{k_2}}\|_2} \|A_{j_{k_1}}\|_2, \end{aligned}$$

then,

$$\frac{A_{j_{k_1}}^T (b - Ay_k)}{\|A_{j_{k_1}}\|_2} = \frac{A_{j_{k_1}}^T (b - Ax_k)}{\|A_{j_{k_1}}\|_2} - \mu_k \frac{A_{j_{k_2}}^T (b - Ax_k)}{\|A_{j_{k_2}}\|_2}. \tag{A2}$$

(ii)

$$\begin{aligned} \frac{A_{j_{k_2}}^T (b - Ay_k)}{\|A_{j_{k_2}}\|_2} &= \frac{A_{j_{k_2}}^T \left[b - A \left(x_k + \frac{A_{j_{k_2}}^T (b-Ax_k)}{\|A_{j_{k_2}}\|_2} e_{j_{k_2}} \right) \right]}{\|A_{j_{k_2}}\|_2} \\ &= \frac{A_{j_{k_2}}^T (b - Ax_k) - \frac{A_{j_{k_2}}^T A_{j_{k_2}} A_{j_{k_2}}^T (b-Ax_k)}{\|A_{j_{k_2}}\|_2^2}}{\|A_{j_{k_2}}\|_2} = 0 \end{aligned} \tag{A3}$$

Substitute (A2) and (A3) into (A1), then,

$$A(x_{k+1}-x_k)=\frac{A_{j_{k_2}}^T(b-Ax_k)}{\|A_{j_{k_2}}\|_2}A_{j_{k_2}}+\left(\frac{A_{j_{k_1}}^T b}{\|A_{j_{k_1}}\|_2}-\mu_k\frac{A_{j_{k_2}}^T b}{\|A_{j_{k_2}}\|_2}-u_k^TAx_k\right)\frac{u_k}{\|u_k\|_2^2}$$

Subtract Ax_* from both sides of the equation, one can get

$$A(x_{k+1}-x_*)=A(x_k-x_*)+\frac{A_{j_{k_2}}^T(b-Ax_k)}{\|A_{j_{k_2}}\|_2}A_{j_{k_2}}+\left(\frac{A_{j_{k_1}}^T b}{\|A_{j_{k_1}}\|_2}-\mu_k\frac{A_{j_{k_2}}^T b}{\|A_{j_{k_2}}\|_2}-u_k^TAx_k\right)\frac{u_k}{\|u_k\|_2^2}.$$

Since

$$A(x_k-x_*)+\frac{A_{j_{k_2}}^T(b-Ax_k)}{\|A_{j_{k_2}}\|_2}A_{j_{k_2}}=A(x_k-x_*)+\frac{A_{j_{k_2}}A_{j_{k_2}}^TA(x_*-x_k)}{\|A_{j_{k_2}}\|_2}=\left(I_m-\frac{A_{j_{k_2}}A_{j_{k_2}}^T}{\|A_{j_{k_2}}\|_2^2}\right)A(x_k-x_*)$$

and

$$\left(\frac{A_{j_{k_1}}^T b}{\|A_{j_{k_1}}\|_2}-\mu_k\frac{A_{j_{k_2}}^T b}{\|A_{j_{k_2}}\|_2}-u_k^TAx_k\right)\frac{u_k}{\|u_k\|_2^2}=\left[\left(\frac{A_{j_{k_1}}^T}{\|A_{j_{k_1}}\|_2}-\mu_k\frac{A_{j_{k_2}}^T}{\|A_{j_{k_2}}\|_2}\right)Ax_*-u_k^TAx_k\right]\frac{u_k}{\|u_k\|_2^2}=\frac{u_ku_k^TA(x_*-x_k)}{\|u_k\|_2^2},$$

then,

$$A(x_{k+1}-x_*)=\left(I_m-\frac{A_{j_{k_2}}A_{j_{k_2}}^T}{\|A_{j_{k_2}}\|_2^2}-\frac{u_ku_k^T}{\|u_k\|_2^2}\right)A(x_k-x_*) \tag{A4}$$

Let $\hat{y}_k$ and $\hat{x}_{k+1}$ be the first and second iterative solutions of x_k obtained by single-step continuous execution of RGS2 algorithm, respectively. i.e.,

$$\hat{y}_k=x_k+\frac{A_{j_{k_1}}^T(b-Ax_k)}{\|A_{j_{k_1}}\|_2^2}e_{j_{k_1}}\quad\text{and}\quad\hat{x}_{k+1}=\hat{y}_k+\frac{A_{j_{k_2}}^T(b-A\hat{y}_k)}{\|A_{j_{k_2}}\|_2^2}e_{j_{k_2}}.$$

According to Theorem 1, one has

$$\begin{aligned}
A(\hat{x}_{k+1}-x_*)&=\left(I_m-\frac{A_{j_{k_2}}A_{j_{k_2}}^T}{\|A_{j_{k_2}}\|_2^2}\right)\left(I_m-\frac{A_{j_{k_1}}A_{j_{k_1}}^T}{\|A_{j_{k_1}}\|_2^2}\right)A(x_k-x_*)\\
&=\left[I_m-\frac{A_{j_{k_2}}A_{j_{k_2}}^T}{\|A_{j_{k_2}}\|_2^2}-\left(I_m-\frac{A_{j_{k_2}}A_{j_{k_2}}^T}{\|A_{j_{k_2}}\|_2^2}\right)\frac{A_{j_{k_1}}A_{j_{k_1}}^T}{\|A_{j_{k_1}}\|_2^2}\right]A(x_k-x_*)\\
&=\left[I_m-\frac{A_{j_{k_2}}A_{j_{k_2}}^T}{\|A_{j_{k_2}}\|_2^2}-\left(\frac{A_{j_{k_1}}A_{j_{k_1}}^T}{\|A_{j_{k_1}}\|_2^2}-\frac{A_{j_{k_2}}A_{j_{k_2}}^TA_{j_{k_1}}A_{j_{k_1}}^T}{\|A_{j_{k_2}}\|_2^2\|A_{j_{k_1}}\|_2^2}\right)\right]A(x_k-x_*)\\
&=\left[I_m-\frac{A_{j_{k_2}}A_{j_{k_2}}^T}{\|A_{j_{k_2}}\|_2^2}-\left(\frac{A_{j_{k_1}}A_{j_{k_1}}^T}{\|A_{j_{k_1}}\|_2^2}-\mu_k\frac{A_{j_{k_2}}A_{j_{k_1}}^T}{\|A_{j_{k_2}}\|_2\|A_{j_{k_1}}\|_2}\right)\right]A(x_k-x_*)\\
&=\left(I_m-\frac{A_{j_{k_2}}A_{j_{k_2}}^T}{\|A_{j_{k_2}}\|_2^2}-\frac{u_kA_{j_{k_1}}^T}{\|A_{j_{k_1}}\|_2}\right)A(x_k-x_*)
\end{aligned}$$

$$= \left(I_m - \frac{A_{j_{k_2}} A_{j_{k_2}}^T}{\|A_{j_{k_2}}\|_2^2} - \frac{u_k u_k^T}{\|u_k\|_2^2} + \frac{u_k u_k^T}{\|u_k\|_2^2} - \frac{u_k A_{j_{k_1}}^T}{\|A_{j_{k_1}}\|_2}\right) A(x_k - x_*)$$

$$= \left(I_m - \frac{A_{j_{k_2}} A_{j_{k_2}}^T}{\|A_{j_{k_2}}\|_2^2} - \frac{u_k u_k^T}{\|u_k\|_2^2}\right) A(x_k - x_*) + \left(\frac{u_k u_k^T}{\|u_k\|_2^2} - \frac{u_k A_{j_{k_1}}^T}{\|A_{j_{k_1}}\|_2}\right) A(x_k - x_*)$$

$$= A(x_{k+1} - x_*) + \left(\frac{u_k u_k^T}{\|u_k\|_2^2} - \frac{u_k A_{j_{k_1}}^T}{\|A_{j_{k_1}}\|_2}\right) A(x_k - x_*),$$

the last equation is obtained from (A4), then

$$A(x_{k+1} - x_*) = A(\hat{x}_{k+1} - x_*) - \left(\frac{u_k u_k^T}{\|u_k\|_2^2} - \frac{u_k A_{j_{k_1}}^T}{\|A_{j_{k_1}}\|_2}\right) A(x_k - x_*).$$

Due to the orthogonality, one can get that,

$$\|A(x_{k+1} - x_*)\|_2^2 = \|A(\hat{x}_{k+1} - x_*)\|_2^2 - \left\| \left(\frac{u_k u_k^T}{\|u_k\|_2^2} - \frac{u_k A_{j_{k_1}}^T}{\|A_{j_{k_1}}\|_2}\right) A(x_k - x_*)\right\|_2^2$$

$$= \|A(\hat{x}_{k+1} - x_*)\|_2^2 - \left| \frac{A_{j_{k_1}}^T A(x_k - x_*)}{\|A_{j_{k_1}}\|_2} - \frac{u_k^T A(x_k - x_*)}{\|u_k\|_2^2} \right|^2 \|u_k\|_2^2.$$

So,

$$E_k \|A(x_{k+1} - x_*)\|_2^2 = E_k \|A(\hat{x}_{k+1} - x_*)\|_2^2 - E_k \left| \frac{A_{j_{k_1}}^T A(x_k - x_*)}{\|A_{j_{k_1}}\|_2} - \frac{u_k^T A(x_k - x_*)}{\|u_k\|_2^2} \right|^2 \|u_k\|_2^2. \tag{A5}$$

Next, we estimate the two conditional expectations on the right side of (A5).

(I) The first expectation

According to Theorem 1,

$$E_k \|A(\hat{x}_{k+1} - x_*)\|_2^2 \le (1 - \frac{\lambda_{min}(A^T A)}{\tau_{max}})(1 - \frac{\lambda_{min}(A^T A)}{\|A\|_F^2}) \|A(x_k - x_*)\|_2^2. \tag{A6}$$

(II) The second expectation

By Lemma 3, $\|u_k\|_2^2 - 1 = -\mu_k^2$. Then,

$$E_k \left| \frac{A_{j_{k_1}}^T A(x_k - x_*)}{\|A_{j_{k_1}}\|_2} - \frac{u_k^T A(x_k - x_*)}{\|u_k\|_2^2} \right|^2 \|u_k\|_2^2$$

$$= E_k \left| \|u_k\|_2 \frac{A_{j_{k_1}}^T A(x_k - x_*)}{\|A_{j_{k_1}}\|_2} - \frac{1}{\|u_k\|_2} \left(\frac{A_{j_{k_1}}^T}{\|A_{j_{k_1}}\|_2} - \mu_k \frac{A_{j_{k_2}}^T}{\|A_{j_{k_2}}\|_2}\right) A(x_k - x_*) \right|^2$$

$$= E_k \left| \left(\|u_k\|_2 - \frac{1}{\|u_k\|_2}\right) \frac{A_{j_{k_1}}^T A(x_k - x_*)}{\|A_{j_{k_1}}\|_2} + \frac{\mu_k}{\|u_k\|_2} \frac{A_{j_{k_2}}^T A(x_k - x_*)}{\|A_{j_{k_2}}\|_2} \right|^2$$

$$= E_k \left| \frac{\mu_k^2}{\|u_k\|_2} \frac{A_{j_{k_1}}^T A(x_k - x_*)}{\|A_{j_{k_1}}\|_2} - \frac{\mu_k}{\|u_k\|_2} \frac{A_{j_{k_2}}^T A(x_k - x_*)}{\|A_{j_{k_2}}\|_2} \right|^2$$

$$= \sum_{j_{k_1}=1}^{n} \frac{\|A_{j_{k_1}}\|_2^2}{\|A\|_F^2} \sum_{\substack{j_{k_2}=1 \\ j_{k_2}\neq j_{k_1}}}^{n} \frac{\|A_{j_{k_2}}\|_2^2}{\|A\|_F^2 - \|A_{j_{k_1}}\|_2^2} \cdot \left| \frac{\mu_k^2}{\|u_k\|_2} \frac{A_{j_{k_1}}^T A(x_k - x_*)}{\|A_{j_{k_1}}\|_2} - \frac{\mu_k}{\|u_k\|_2} \frac{A_{j_{k_2}}^T A(x_k - x_*)}{\|A_{j_{k_2}}\|_2} \right|^2$$

$$= \sum_{j_{k_1}=1}^{n} \sum_{\substack{j_{k_2}=1 \\ j_{k_2}\neq j_{k_1}}}^{n} \frac{\|A_{j_{k_1}}\|_2^2 \|A_{j_{k_2}}\|_2^2}{\|A\|_F^2 \tau_{max}} \left| \frac{\mu_k^2}{\|u_k\|_2} \frac{A_{j_{k_1}}^T A(x_k - x_*)}{\|A_{j_{k_1}}\|_2} - \frac{\mu_k}{\|u_k\|_2} \frac{A_{j_{k_2}}^T A(x_k - x_*)}{\|A_{j_{k_2}}\|_2} \right|^2.$$

By Lemma 4, it is established as follows

$$E_k \left| \frac{A_{j_{k_1}}^T A(x_k - x_*)}{\|A_{j_{k_1}}\|_2} - \frac{u_k^T A(x_k - x_*)}{\|u_k\|_2^2} \right|^2 \|u_k\|_2^2$$

$$\geq \sum_{s=1}^{n} \sum_{\substack{t=1 \\ t\neq s}}^{n} \frac{\|A_s\|_2^2 \|A_t\|_2^2}{\|A\|_F^2 \tau_{max}} \left| \alpha_{s,t} \frac{A_s^T A(x_k - x_*)}{\|A_s\|_2} - \beta_{s,t} \frac{A_t^T A(x_k - x_*)}{\|A_t\|_2} \right|^2$$

$$= \frac{1}{\|A\|_F^2 \tau_{max}} \sum_{s<t} \|A_s\|_2^2 \|A_t\|_2^2 \left(\left| \alpha_{s,t} \frac{A_s^T A(x_k - x_*)}{\|A_s\|_2} - \beta_{s,t} \frac{A_t^T A(x_k - x_*)}{\|A_t\|_2} \right|^2 + \left| \alpha_{s,t} \frac{A_t^T A(x_k - x_*)}{\|A_t\|_2} - \beta_{s,t} \frac{A_s^T A(x_k - x_*)}{\|A_s\|_2} \right|^2 \right).$$

For any $\alpha, \beta, \theta, \eta \in \mathbb{R}$, we note the fact that

$$|\alpha\theta - \beta\eta|^2 + |\alpha\eta - \beta\theta|^2 \geq (|\alpha| - |\beta|)^2 (|\theta|^2 + |\eta|^2).$$

Then,

$$E_k \left| \frac{A_{j_{k_1}}^T A(x_k - x_*)}{\|A_{j_{k_1}}\|_2} - \frac{u_k^T A(x_k - x_*)}{\|u_k\|_2^2} \right|^2 \|u_k\|_2^2$$

$$\geq \frac{1}{\|A\|_F^2 \tau_{max}} \sum_{s<t} \|A_s\|_2^2 \|A_t\|_2^2 (|\alpha_{s,t}| - |\beta_{s,t}|)^2 \left(\left| \frac{A_t^T A(x_k - x_*)}{\|A_t\|_2} \right|^2 + \left| \frac{A_s^T A(x_k - x_*)}{\|A_s\|_2} \right|^2 \right)$$

$$\geq \frac{1}{\|A\|_F^2 \tau_{max}} \sum_{s<t} (|\alpha_{s,t}| - |\beta_{s,t}|)^2 (\|A_s\|_2^2 |A_t^T A(x_k - x_*)|^2 + \|A_t\|_2^2 |A_s^T A(x_k - x_*)|^2).$$

By Lemma 4, $(|\alpha_{s,t}| - |\beta_{s,t}|)^2 \geq \gamma$. Then

$$E_k \left| \frac{A_{j_{k_1}}^T A(x_k - x_*)}{\|A_{j_{k_1}}\|_2} - \frac{u_k^T A(x_k - x_*)}{\|u_k\|_2^2} \right|^2 \|u_k\|_2^2 \geq \frac{\gamma}{\|A\|_F^2 \tau_{max}} \sum_{s<t} \left(\|A_s\|_2^2 \left| A_t^T A(x_k - x_*) \right|^2 + \|A_t\|_2^2 |A_s^T A(x_k - x_*)|^2 \right)$$

$$\geq \frac{\gamma}{\|A\|_F^2 \tau_{max}} \sum_{s=1}^{n} \sum_{\substack{t=1 \\ t\neq s}}^{n} \|A_t\|_2^2 |A_s^T A(x_k - x_*)|^2$$

$$= \frac{\gamma}{\|A\|_F^2 \tau_{max}} \sum_{s=1}^{n} ((\|A\|_F^2 - \|A_s\|_2^2) |A_s^T A(x_k - x_*)|^2)$$

$$\geq \frac{\gamma \tau_{min}}{\|A\|_F^2 \tau_{max}} \sum_{s=1}^{n} |A_s^T A(x_k - x_*)|^2$$

$$\geq \frac{\lambda_{min}(A^T A) \gamma \tau_{min}}{\|A\|_F^2 \tau_{max}} \|A(x_k - x_*)\|^2,$$

i.e.,

$$E_k\left|\frac{A_{j_{k_1}}^T A(x_k - x_*)}{\|A_{j_{k_1}}\|_2} - \frac{u_k^T A(x_k - x_*)}{\|u_k\|_2^2}\right|^2 \|u_k\|_2^2 \geq \frac{\lambda_{min}(A^T A)\gamma\tau_{min}}{\|A\|_F^2 \tau_{max}} \|A(x_k - x_*)\|_2^2 \tag{A7}$$

Substitute (A6) and (A7) into (A5), one can obtain that

$$E_k\|A(x_{k+1} - x_*)\|_2^2 \leq \left[\left(1 - \frac{\lambda_{min}(A^T A)}{\tau_{max}}\right)\left(1 - \frac{\lambda_{min}(A^T A)}{\|A\|_F^2}\right) - \frac{\lambda_{min}(A^T A)\gamma\tau_{min}}{\|A\|_F^2 \tau_{max}}\right]\|A(x_k - x_*)\|_2^2.$$

Thus,

$$E_k\|x_{k+1} - x_*\|_{A^T A}^2 \leq \left[\left(1 - \frac{\lambda_{min}(A^T A)}{\tau_{max}}\right)\left(1 - \frac{\lambda_{min}(A^T A)}{\|A\|_F^2}\right) - \frac{\lambda_{min}(A^T A)\gamma\tau_{min}}{\|A\|_F^2 \tau_{max}}\right]\|x_k - x_*\|_{A^T A}^2.$$