CONVERGENCE OF THE RANDOMIZED BLOCK GAUSS-SEIDEL METHOD

WEI MAGGIE WU

SPONSOR: DEANNA NEEDELL

ABSTRACT. The Randomized Gauss-Seidel Method (RGS) is an iterative algorithm that solves large-scale systems of linear equations $Ax = b$. This paper studies a block version of the RGS method, the Randomized Block Gauss-Seidel Method (RBGS). At each step, the algorithm greedily minimizes the objective function $L(x) = \|Ax-b\|_2^2$ with respect to a subset of coordinates. This paper describes the RBGS method, which uses a randomized control method to choose a subset of columns of A at each step. We show this method exhibits an expected linear convergence rate that can be described by the properties of the matrix A and its column submatrices. The analysis demonstrates that convergence of RBGS improves upon RGS when given an appropriate column-paving of the matrix, a partition of the columns into well-conditioned blocks. The main result yields a RBGS method that is more efficient than the classical RGS method, and bridges existing theory on the related block Kaczmarz method and the convergence analysis of classical RGS.

1. INTRODUCTION

The Randomized Gauss-Seidel Method (RGS) [LL10, MNR15] is an iterative algorithm for solving large-scale linear systems of equations. This approach has gained recent attention in applications like digital image and signal processing due to its simplicity of implementation and its ability to find the solution without needing access to the entire system at once [Xu18, HNR17, MNR15, Nat01, GBH70, CZ97]. In the RGS method, each iteration greedily minimizes the objective function $L(x) = \|Ax - b\|_2^2$ with respect to a selected coordinate, and the results converge to the least-squares solution at a linear rate. Another algorithm widely used for solving linear systems of equations is the Randomized Kaczmarz method (RK) [Kac37, SV09]. In RK, each iteration projects the estimate from one equation's solution space (hyperplane) to the other, converging to the solution of a consistent system (or close to the solution of an inconsistent system [Nee10]) at a linear rate. The block version of the RK algorithm [Elf80] was analyzed in [NT13], utilizing a *row-paving*of the matrix, which is a partition of the matrix into blocks of rows such that each block submatrix has bounded singular values. With this assumption, linear convergence with improved computational complexity was possible. While the simple Kaczmarz algorithm enforces one single constraint at each iteration, the block update enforces multiple constraints simultaneously at each iteration. The row-paving of the matrix guarantees that each subset of constraints yields a well-conditioned system.

Contribution. Inspired by the block Kaczmarz algorithm, this paper demonstrates that the block approach can also be applied to the Randomized Gauss-Seidel Method to improve the convergence. We view our work as a bridge between the convergence results for the block RK

Date: October 5, 2018.

This work was supported by NSF CAREER grant #1348721, and the Alfred P. Sloan Fellowship. Wu is at Scripps College, Claremont CA 91711. Needell is at the University of California, Los Angeles, 90095.

method [NT13] and for the classical RGS method [MNR15]. The latter work analyzes the Gauss-Seidel method in parallel to the Kaczmarz method, proving convergence bounds and comparing the two approaches in various settings. We view our work as completing the convergence framework for these approaches.

1.1. **Model and Notation.** The ℓ_p vector norm for $p \in [1,\infty]$ is denoted $\|\cdot\|_p$, and the usual inner (dot) product by $\langle\cdot,\cdot\rangle$. For matrices, $\|\cdot\|$ denotes the spectral norm and $\|\cdot\|_F$ denotes the Frobenius norm. The set of columns of A is denoted $\{A^1, A^2, ..., A^n\}$, and the set of rows is denoted $\{A_1, A_2, ..., A_m\}$. For a set of row indices τ, A_τ denotes the submatrix of A indexed by a set τ (for which we will always use Greek letters and should be clear from context). In block RK, A_τ denotes the row submatrix and in RBGS, A_τ denotes the column submatrix; the distinction should be clear from context. For a Hermitian (self-adjoint) matrix, $\lambda_{\min}$ and $\lambda_{\max}$ are the algebraic minimum and maximum eigenvalues. For an $m \times n$ matrix A, the singular values are arranged such that

$$\sigma_{\max}(A) := \sigma_1(A) \geq \sigma_2(A) \geq ... \geq \sigma_{\min\{m,n\}}(A) =: \sigma_{\min}(A). \tag{1}$$

We also define the condition number $\kappa(A) := \sigma_{\max}(A)/\sigma_{\min}(A)$ and the scaled condition number $K(A) := \|A\|_F/\sigma_{\min}(A)$. Note that $\kappa(A) \leq K(A)$. The adjoint (transpose) of a matrix or vector A is denoted A^*. The Moore-Penrose pseudoinverse of matrix A is denoted $A^\dagger$. When matrix A has linearly independent columns, the pseudoinverse $A^\dagger = (A^*A)^{-1}A^*$. Lastly, we write $e_{(j)}$ for the jth coordinate basis column vector, which has a 1 in the jth position and 0 in the rest of the entries.

Consider a system of linear equations

$$Ax = b, \tag{2}$$

where A is a real $m \times n$ matrix[1]. We consider the interesting case when the system is overdetermined ($m \geq n$) and we seek the least squares solution:

$$\operatorname*{argmin}_x \|Ax - b\|_2^2. \tag{3}$$

For convenience, we assume that A is standardized, in other words each row A_i of A has

$$\|A_i\|_2 = 1 \text{ for each } i = 1, ..., m.$$

In the following discussion, we will utilize the following convention.

Definition 1. *Define the unique minimizer of* (3)*, x_*. We also introduce the residual vector $r_* := Ax_* - b$. Subsequently, $b = Ax_* - r_*$.*

1.2. **Organization.** The next subsection discusses related work and introduces the relevant iterative methods. Section 1.4 derives our formulation of a block RGS method. Section 2 lays out the main theorem concerning the convergence rate and the expected error bound, followed by a detailed proof and an analysis of the result. Section 3 analyzes the results of the experiments comparing RBGS with different sizes of partitions applied to both consistent and inconsistent systems and synthetic and real data. Section 4 concludes the paper and indicates future directions of work.

1.3. **Related Works.**

[1]There is nothing preventing the use of complex-valued matrices, we simply use real matrices for simplicity of presentation.

1.3.1. *The Randomized Gauss-Seidel Method (RGS).* Taking A, b as input and beginning from an arbitrarily chosen x_0, the RGS Method, also known as the Randomized Coordinate Descent Method, repeats the following in each iteration.

First, a column $j \in \{1, ..., n\}$ is selected at random with probability proportional to the square of its Euclidean norm:

$$\Pr(\text{column} = j) = \frac{\|A^j\|_2^2}{\|A\|_F^2}. \tag{4}$$

We then minimize $L(x) = \frac{1}{2}\|b - Ax\|_2^2$ (equivalent to minimizing (3)) with respect to the selected coordinate to get

$$x_t := x_{t-1} + \frac{\left(A^j\right)^* (b - Ax_{t-1})}{\|A^j\|_2^2} e_{(j)}, \tag{5}$$

where $e_{(j)}$ is the coordinate vector defined above. The method continues with these updates, where in each iteration a new column index j is selected at random (typically uniformly at random, with replacement). This method is described in Algorithm 1 below. Leventhal and Lewis [LL10] show that this algorithm has an expected linear convergence rate, as given by the following theorem.

Algorithm 1 The Randomized Gauss-Seidel (RGS)

procedure (A, b, T,h) ▷ $m \times n$ matrix A, $b \in \mathbb{C}^m$, maximum iterations h
  Initialize $x_0 = 0$, $r_0 = b - Ax_0$
  for $t = 1, 2, ..., h$ **do**
    Choose j uniformly from $\{1, \ldots, n\}$.
    Set $x_t = x_{t-1} + \frac{\left(A^j\right)^* (b - Ax_{t-1})}{\|A^j\|_2^2} e_{(j)}$
    Update $r_t = b - Ax_t$.
  end for
  Output x_t
end procedure

Theorem 1 (from [LL10])**.** *Given any linear system $Ax = b$, where the matrix A is a non-zero $m \times n$ matrix, define the least-squares residual and the error by*

$$L(x) = \frac{1}{2}\|Ax - b\|_2^2 \quad \textit{and} \quad \delta(x) = L(x) - L(x_*),$$

where x_* is a least-squares solution. Then the iterates described by (5) *converge linearly in expectation to a least-squares solution for the system: for each iteration $t = 0, 1, ...,$*

$$\mathbb{E}[\delta(x_t)] \leq \left[1 - \frac{\sigma_{\min}^2(A)}{\|A\|_F^2}\right] \delta(x_{t-1}),$$

where the expectation is taken with respect to the i.i.d. uniform at random selection of the column indices j.

Notice that when the system $Ax = b$ is consistent, $b = Ax_*$ and $L(x_*) = 0$. The above inequality can be reinterpreted as

$$\mathbb{E}\left[\|Ax_t - Ax_*\|_2^2\right] \leq (1 - \gamma_1)^t \|Ax_0 - Ax_*\|_2^2, \tag{6}$$

where $\gamma_1 = \frac{\sigma_{\min}^2(A)}{\|A\|_F^2}$.

1.3.2. *The Randomized Kaczmarz Method (RK).* The RK method we discuss here was first analyzed in [SV09]. At each iteration, the RK method projects the current estimate orthogonally onto the solution hyperplane $\langle A_i, x\rangle = b_i$. The algorithm is described as

$$x_{t+1} = x_t + \frac{b_i - \langle A_i, x_t\rangle}{\|A_i\|_2^2} A_i, \tag{7}$$

where A_i denotes the ith row of matrix A and each row is selected with probability proportional to its Euclidean norm. When we assume a standardized matrix A, each row is selected uniformly at random. Vershynin and Strohmer [SV09] proved a linear rate of convergence that only depends on the scaled condition number of A but not on the number of equations in the system. Given any initial estimate x_0,

$$\mathbb{E}\|x_t - x_*\|_2^2 \leq \left[1 - \frac{1}{K(A)^2}\right]^t \|x_0 - x_*\|_2^2.$$

1.3.3. *The Block Randomized Kaczmarz with row-paving condition.* Elfving and Eggermont [Elf80] first proposed a block RK method. The method we consider here first partitions the rows $\{1, ..., m\}$ into N blocks, and the partition is denoted $\tau_{\{1,...,N\}}$. At each iteration, a block τ_i is uniformly selected at random, and the current estimate is projected orthogonally onto the solution space $A_{\tau_i} x = b_{\tau_i}$. The algorithm is described as

$$x_{t+1} = x_t + (A_{\tau_i})^\dagger (b_{\tau_i} - A_{\tau_i} x_t), \tag{8}$$

where A_{τ_i} and b_{τ_i} respectively denote the row submatrix and the subvector of b indexed by τ_i.

To prove convergence, Needell and Tropp [NT13] utilized a *row-paving*. A row-paving with parameters (s, α, β) is defined as a partition $P = \tau_{\{1,...,s\}}$ such that

$$\alpha \leq \lambda_{\min}(A_\tau A_\tau^*) \text{ and } \lambda_{\max}(A_\tau A_\tau^*) \leq \beta \text{ for each } \tau \in P.$$

In other words, s determines the size of the partition, and α and β restrict the lower and upper bound of the singular values of the partitioned submatrices. As a simple but illustrative example, consider the 40×10 matrix that consists of four copies of the identity matrix stacked on top of each other. Then a row-paving of this matrix with $s = 4$ might correspond to the partition $\tau_1 = \{1, \ldots 10\}$, $\tau_2 = \{11, \ldots 20\}$, $\tau_3 = \{21, \ldots 30\}$, and $\tau_4 = \{31, \ldots 40\}$. Then since each block in the partition consists of a single copy of the 10×10 identity matrix, the singular values of each block are all 1. Thus we have $\alpha = \beta = 1$ in this case.

Now consider the least-squares problem (3) when a row-paving $P = (s, \alpha, \beta)$ is applied to matrix A with full column rank. We have the expected error bound

$$\mathbb{E}\|x_t - x_*\|_2^2 \leq \left[1 - \frac{\sigma_{\min}^2(A)}{\beta m}\right]^t \|x_0 - x_*\|_2^2 + \frac{\beta}{\alpha} \frac{\|r_*\|_2^2}{\sigma_{\min}^2(A)},$$

where x_* and r_* are defined in Section 1.1.

[NT13] also cites [Ver06] and [Tro09] to demonstrate that every standardized matrix has a good row-paving.

1.4. **The Randomized Block Gauss-Seidel (RBGS).**

1.4.1. *Deriving the Algorithm.* Our goal here is to utilize a paving in order to analyze a block variant of RGS. However, contrary to RK, which projects the current state onto a row plane or space, RGS selects one coordinate (one column) at each iteration. As a result, we partition the columns, not the rows, for RGS. At each iteration, the objective $L(x) = \frac{1}{2}\|b - Ax\|_2^2$ will be minimized with respect to all the coordinates represented in the selected partition, thereby minimizing through multiple directions at the same time.

Given the system (2) and a block $\tau \in P = \tau_{\{1,...s\}}$, we want to minimize $L(x_{t+1}) = \frac{1}{2}\|b - Ax_{t+1}\|_2^2$ given x_t and the residual vector $r_t = b - Ax_t$. Inspired by the classical RGS method, we presuppose

$$x_{t+1} = x_t + \sum_{k=1}^{T} \alpha_k e_k, \tag{9}$$

where T is the number of coordinates included in the selected τ. We write the set $\tau = \{c_1, c_2, ..., c_T\}$, where c_i is a column index for the subset. The set $\{e_1, e_2, ..., e_T\}$ is one-to-one with the set τ, where $\forall i \in \{1, 2, ..., T\}$, e_i is the c_ith coordinate basis column vector ($e_i = e_{(c_i)}$). The set $\{\alpha_1, ..., \alpha_T\}$ is the set of constants which minimizes $L(x_{t+1})$ given x_t.

To find $\{\alpha_1, ..., \alpha_T\}$, we want to minimize $L(x_{t+1}) = \frac{1}{2}\|b - Ax_{t+1}\|_2^2$,

$$\text{and } \min_{\alpha_1,...,\alpha_T} L(x_{t+1}) = \min_{\alpha_1,...,\alpha_T} \frac{1}{2}\left(\sum_{i=1}^{m} \langle A_i, x_t + \sum_{k=1}^{T} \alpha_k e_k\rangle - b_i\right)^2. \tag{10}$$

Taking the derivative of $L(x_{t+1})$ with respect to α_u and setting it to zero, we find that

$$0 = -\|A^u\|_2^2 \alpha_u + \langle r_t, A^u\rangle - \sum_{k\neq u}^{T} \alpha_k \langle A^k, A^u\rangle.$$

To solve (10), we now have to solve the system of equations

$$\forall u \in \{1, ..., T\}, \quad \sum_{i=1}^{T} \alpha_i \langle A^u, A^i\rangle = \langle r_t, A^u\rangle. \tag{11}$$

With some close observation, the above system is actually equivalent to

$$A_\tau^* A_\tau \alpha^* = (r_t^* A_\tau)^*, \tag{12}$$

where α denotes the vector $(\alpha_1, ..., \alpha_T)$, and A_τ recall denotes the column submatrix of A indexed by τ. We can solve (12) and get

$$\alpha^* = (A_\tau^* A_\tau)^{-1}(r_t^* A_\tau)^* = (A_\tau^* A_\tau)^{-1} A_\tau^* r_t = A_\tau^\dagger r_t. \tag{13}$$

Next, to fully determine a block Gauss-Seidel algorithm, we must decide on what blocks of indices are acceptable. We propose a selection method by two design decisions. First, inspired by [NT13], we define a column-paving of A as row-paving of $A^\dagger$. We again define the row-paving of $A^\dagger$ with (s, α, β) as a partition $P = \tau_{\{1,...,s\}}$ on $A^\dagger$ such that

$$\alpha \leq \lambda_{\min}(A_\tau^\dagger A_\tau^{\dagger *}) = \sigma_{\min}^2(A_\tau^\dagger) \text{ and } \sigma_{\max}^2(A_\tau^\dagger) = \lambda_{\max}(A_\tau^\dagger A_\tau^{\dagger *}) \leq \beta \text{ for each } \tau \in P, \tag{14}$$

where s is called the size of the partition, and α and β determine the lower and upper bound. Notice that $\alpha = 0$ unless $A_\tau^\dagger$ is a "fat" submatrix which has more columns than rows. The row paving guarantees that each block is well-conditioned, so that when we project onto that block's solution space, we make substantial progress toward the solution of the overall system. Indeed,

in the proof of our main result we will see that these bounds on the singular values are crucial in order to bound the improvement in the solution error.

Secondly, at each iteration, independent of all previous choices, we select a block τ uniformly at random from the partition P. These two decisions lead to Algorithm 2 described below. Similar to the RGS method (Algorithm 1), the RBGS Method iteratively improves the approximation by adjusting the value of multiple coordinates (adding the vector α to the current estimate), which finally converges to the least-squares solution x_*.

Algorithm 2 The Randomized Block Gauss-Seidel (RBGS)

procedure (A, b, T,h) ▷ $m \times n$ matrix A, $b \in \mathbb{C}^m$, $T = \frac{n}{s}$ the number of coordinates in each block τ, maximum iterations h
  Initialize $x_0 = 0$, $r_0 = b - Ax_0$
  for $t = 1, 2, ..., h$ **do**
    Choose τ uniformly from partition $P = \tau_{\{1,...,s\}}$, assume $\tau = \{c_1, ..., c_T\}$.
    Create a block of A, A_τ containing columns of A indexed by τ.
    Generate E, an $n \times T$ matrix. $\forall i \in \{1, ..., T\}$, the ith column of E, E^i, has all zeros with a 1 in the c_ith position, where c_i is the ith entry in the selected τ as indicated above.
    Set $x_t = x_{t-1} + EA_\tau^\dagger r_{t-1}$
    Update $r_t = b - Ax_t$.
  end for
  Output x_t
end procedure

2. Analysis of the Randomized Block Gauss-Seidel Method

This section states our main result, which gives linear convergence for the RBGS Method described in Algorithm 1. The proof itself is inspired by [NT13] on the linear convergence of the block RK method and by [MNR15] on the linear convergence of the RGS method.

2.1. **Main result.**

Theorem 2. *Given a standardized real $m \times n$ matrix A and an $m \times 1$ vector b, let P be a column partition (s, α, β) as defined in* (14)*. Consider the least-squares problem*

$$\text{minimize } \|Ax - b\|_2^2.$$

Let x_* be the unique minimizer, and define the residual $r_* := Ax_* - b$. For any initial estimate x_0, the Randomized Block Gauss-Seidel Method (RBGS) described in Algorithm 1 produces a sequence $\{x_t : t \geq 0\}$ of iterates that satisfies:

$$\mathbb{E}\|x_t - x_*\|_2^2 \leq \kappa^2(A)\gamma^t \|x_0 - x_*\|_2^2, \tag{15}$$

where $\gamma = 1 - \frac{\alpha\sigma_{\min}^2(A)}{s}$ and $\kappa(A)$ is the condition number.

Before we prove the main result, we prove three lemmas.

Lemma 1. *$A(x_t - x_{t-1})$ and $A(x_t - x_*)$ are orthogonal.*

Proof:

According to the update rule in Algorithm 1,

$$x_t = x_{t-1} + EA_\tau^\dagger r_{t-1}$$

where E is a $n \times T$ matrix. In addition, $\forall i \in \{1,2,...,T\}$, the ith column of E has all zeros with a 1 in the c_ith position when $\tau = \{c_1,...c_T\}$ and $T = \frac{n}{s}$.

Multiplying both sides of the above equation with A, we get

$$A(x_t - x_{t-1}) = AEA_\tau^\dagger r_{t-1} = A_\tau A_\tau^\dagger r_{t-1}.$$

Now note that classical analyses (e.g. [MNR15]) observe that when only selecting one column j at a time, $A(x_t - x_{t-1})$ is parallel to A^j. Since $\forall j \in \tau$, $A(x_t - x_{t-1})$ is parallel to A^j, $A(x_t - x_{t-1})$ is "parallel" to the column space of A_τ.

By the way we derive our algorithm in (10), $\forall u \in \tau$, $\frac{\partial L(x_t)}{\partial A_u} = 0$. Since u is chosen arbitrarily, $\frac{\partial L(x_t)}{\partial A_\tau} = 0$, so $A(x_t - x_*)$ is orthogonal to the column space of A_τ. Then $A(x_t - x_{t-1})$ is orthogonal to $A(x_t - x_*)$.

■

Lemma 2. *For any vector u,* $\mathbb{E}\|A_\tau A_\tau^\dagger u\|_2^2 \geq \frac{\alpha\sigma_{\min}^2(A)}{s}\|u\|_2^2$.

Proof: First, we have

$$\mathbb{E}\|A_\tau v\|_2^2 = \frac{1}{s}\sum_{\tau\in P}\|A_\tau v\|_2^2 = \frac{1}{s}\|Av\|_2^2 \geq \frac{1}{s}\sigma_{\min}^2(A)\|v\|_2^2.$$

Now take $v = A_\tau^\dagger u$. Then

$$\begin{aligned}\mathbb{E}\|A_\tau A_\tau^\dagger u\|_2^2 &\geq \frac{1}{s}\sigma_{\min}^2(A)\mathbb{E}\|A_\tau^\dagger u\|_2^2 \\ &\geq \frac{1}{s}\sigma_{\min}^2(A)\mathbb{E}[\sigma_{\min}^2(A_\tau^\dagger)\|u\|_2^2] \geq \frac{\alpha}{s}\sigma_{\min}^2(A)\|u\|_2^2.\end{aligned}$$

■

Lemma 3. *We have* $A_\tau^\dagger r_* = 0$.

Proof: Recall that the residual error $r_* = Ax_* - b$ is orthogonal to every column of A. $\forall$ column A^j in A_τ, $\langle A^j, r_*\rangle = 0$, so $A_\tau^* r_* = 0$.

Then

$$A_\tau^\dagger r_* = (A_\tau^* A_\tau)^{-1} A_\tau^* r_* = 0.$$

■

The proof for the main result follows directly from the above three lemmas.

Proof (of Theorem 2):

According to Lemma 1,

$$\|Ax_t - Ax_*\|_2^2 = \|Ax_{t-1} - Ax_*\|_2^2 - \|Ax_t - Ax_{t-1}\|_2^2.$$

At each iteration, the expected value is taken conditional on the first $t-1$ iterations, so we have

$$\mathbb{E}\|Ax_t - Ax_*\|_2^2 = \|Ax_{t-1} - Ax_*\|_2^2 - \mathbb{E}\|Ax_t - Ax_{t-1}\|_2^2. \tag{16}$$

By the definition of the algorithm estimate,

$$x_t = x_{t-1} + EA_\tau^\dagger r_{t-1} = x_{t-1} + EA_\tau^\dagger(b - Ax_{t-1}) = x_{t-1} + EA_\tau^\dagger(Ax_* - r_* - Ax_{t-1}),$$

where the third equation is based upon Definition 1.

We then apply Lemma 3 to get that

$$\begin{aligned} A(x_t - x_{t-1}) &= A_\tau A_\tau^\dagger (Ax_* - Ax_{t-1} - r_*) \\ &= A_\tau A_\tau^\dagger (Ax_* - Ax_{t-1}) - A_\tau A_\tau^\dagger r_* \\ &= A_\tau A_\tau^\dagger (Ax_* - Ax_{t-1}), \end{aligned}$$

and

$$\mathbb{E}(\|A(x_t - x_{t-1})\|_2^2) = \mathbb{E}(\|A_\tau A_\tau^\dagger (Ax_* - Ax_{t-1})\|_2^2 \tag{17}$$

If we combine (16) and (17), we have

$$\mathbb{E}\|Ax_t - Ax_*\|_2^2 = \|Ax_{t-1} - Ax_*\|_2^2 - \mathbb{E}\left(\|A_\tau A_\tau^\dagger (Ax_* - Ax_{t-1})\|_2^2\right). \tag{18}$$

Now according to Lemma 2, plug in $u = Ax_* - Ax_{t-1}$ to the inequality, we have

$$\mathbb{E}\|A_\tau A_\tau^\dagger (Ax_* - Ax_{t-1})\|_2^2 \geq \frac{\alpha \sigma_{\min}^2(A)}{s} \|Ax_* - Ax_{t-1}\|_2^2. \tag{19}$$

Now plug in (19) to (18), we have

$$\begin{aligned} \mathbb{E}\|Ax_t - Ax_*\|_2^2 &\leq \|Ax_{t-1} - Ax_*\|_2^2 - \frac{\alpha \sigma_{\min}^2(A)}{s} \|Ax_* - Ax_{t-1}\|_2^2 \\ &= \left[1 - \frac{\alpha \sigma_{\min}^2(A)}{s}\right] \|Ax_* - Ax_{t-1}\|_2^2. \end{aligned}$$

Finally, notice that

$$\|x_t - x_*\|_2^2 = \|A^\dagger A(x_t - x_*)\|_2^2 \leq \|A^\dagger\|^2 \|A(x_t - x_*)\|_2^2.$$

Then,

$$\begin{aligned} \mathbb{E}\|x_t - x_*\|_2^2 &\leq \mathbb{E}\|A^\dagger\|^2 \|A(x_t - x_*)\|_2^2 \leq \|A^\dagger\|^2 \mathbb{E}\|Ax_t - Ax_*\|_2^2 \\ &\leq \sigma_{\max}^2(A^\dagger) \left[1 - \frac{\alpha \sigma_{\min}^2(A)}{s}\right] \|A\|^2 \|x_* - x_{t-1}\|_2^2 \\ &\leq \sigma_{\max}^2(A^\dagger) \sigma_{\max}^2(A) \left[1 - \frac{\alpha \sigma_{\min}^2(A)}{s}\right] \|x_* - x_{t-1}\|_2^2 \\ &= \frac{\sigma_{\max}^2(A)}{\sigma_{\min}^2(A)} \left[1 - \frac{\alpha \sigma_{\min}^2(A)}{s}\right] \|x_* - x_{t-1}\|_2^2 \\ &= \kappa^2(A) \left[1 - \frac{\alpha \sigma_{\min}^2(A)}{s}\right] \|x_* - x_{t-1}\|_2^2, \end{aligned}$$

where $\kappa(A)$ is the condition number of A.

By iterating the above result and taking $\gamma = 1 - \frac{\alpha \sigma_{\min}^2(A)}{s}$, we have

$$\mathbb{E}\|x_t - x_*\|_2^2 \leq \kappa^2(A) \gamma^t \|x_* - x_0\|_2^2,$$

and we have proved our main result.

■

2.2. **Interpreting the Result.** First, we compare RBGS Method with the block RK Method. When $\kappa(A)$ is a constant, the convergence rate of RBGS only depends on the size and lower bound of the partition. At the same time, even when the system is inconsistent, there is no "convergence horizon" in the result; RBGS will always converge to the solution to the least-squares problem of the system (3). On the other hand, the block RK Method will experience a convergence horizon. This is intuitive by the geometry of the RK approaches, which projects iterates onto hyperplanes, never allowing the method to converge to a least-squares solution [Nee10] (note however, that properly chosen step sizes or residual projections can alleviate this issue [CEG83, HN90, Tan71, WM67, Pop98, ZF13]). When restricting the partition, we want a small s and a large α for a fast convergence rate. In addition, regardless of the characteristics of the partition, RBGS performs especially well when the condition number of A is relatively small.

Second, we compare Theorem 2 with the results of classical RGS, Theorem 1. To better compare Theorem 1 with the main result from Theorem 2, we may obtain a corollary from Theorem 2 when $s = n$.

Corollary 1. *Given a standardized real $m \times n$ matrix A and an $m \times 1$ vector b. Let P be a column partition of size n, in other words every A_τ only contains a column of A. Consider the least-squares problem:*

$$\text{minimize } \|Ax - b\|_2^2.$$

Let x_* be the unique minimizer. Let $L(x)$ and $\delta(x)$ be as defined in Theorem 1. Let α be the lower bound of the partition P defined in (14)*. For any initial estimate x_0, the RBGS Method described in Algorithm 1 produces a sequence $\{x_t : t \geq 0\}$ of iterates that satisfies*

$$\mathbb{E}\|Ax_t - Ax_*\|_2^2 \leq (1 - \gamma_2)\|Ax_{t-1} - Ax_*\|_2^2 \tag{20}$$

where $\gamma_2 = \frac{\alpha\sigma_{\min}^2(A)}{n}$.

Let ρ_{simple} and ρ_{block} denote the convergence rate of the simple RGS and RBGS respectively. Then we have $\rho_{\text{simple}} \geq \gamma_1$ and $\rho_{\text{block}} \geq \gamma_2$. Notice that when A is standardized, $\|A\|_F^2 = n$ and $\alpha = \beta = 1$, so that in Corollary 1, $\gamma_2 = \frac{\sigma_{\min}^2(A)}{n} = \frac{\sigma_{\min}^2(A)}{\|A\|_F^2} = \gamma_1$. In other words, when every subset partitioned in RBGS only contains one column, its convergence rate is the same as the convergence rate of the simple RGS, as expected.

Beyond the basic case, RBGS may significantly improve the convergence rate. When we have partition $P = (s, \alpha, \beta)$, $\rho_{\text{block}} \geq \frac{\alpha\sigma_{\min}^2(A)}{s}$. To achieve the same reduction in error, the simple RGS method requires a factor $\frac{n\alpha}{s}$ more iterations than the RBGS method. Ideally, for better improvement, $\frac{n\alpha}{s}$ should be as large as possible. In other words, each block A_τ should contain as many columns as possible to make s small, while maintaining a great lower bound for the singular values in A_τ. However, the most arithmetically expensive step in Algorithm 2 is computing $A_\tau^\dagger$, and containing as much columns as possible in A_τ might significantly lower the computational speed. There should be a balance between fast convergence rate and fast implementation speed. See [NT13] for further discussion of this trade-off and for examples of matrices with fast computations.

As the block method might involves more arithmetic than the simple method, it is more fair for RGS to compare the convergence rate per epoch, which is the minimum number of iterations the algorithm takes to hit each column of A once. For RGS, an epoch consists of n iterations. For RBGS, an epoch consists of s iterations, where s denotes the size of the partition. Under this new setting, both RGS and RBGS require similar amount of arithmetic in one epoch,

so we now consider the per epoch convergence rate. The new convergence rates of RGS and RBGS become $n\rho_{simple}$ and $s\rho_{block}$, and we have

$$n\rho_{simple} \leq \sigma^2_{min}(A), s\rho_{block} \leq \alpha\sigma^2_{min}(A).$$

We see that, in theory, the per-epoch convergence of RBGS is worse, and the disadvantage decreases with the lower bound α on the paving.

3. Experiments

To test our algorithm, we used MATLAB random matrices as well as real data to test the convergence of the RBGS method applied to overdetermined systems of equations.

In the first experiment, we wanted to see how the size of the partition influenced the convergence rate for both consistent and inconsistent systems. To test the consistent system, we created a 120×1 vector x_* where each entry was selected independently from a standard normal distribution, and we set $b = Ax_*$. A was a random 300×120 standardized matrix A, with each entry of A selected independently from a normal distribution and then standardized. The number 120 is selected so that it could divide 1,2,3 and 4. Notice that since x_* is the solution to the system $Ax = b$, it is also the least-squares solution to the problem (3). To test the inconsistent system, we created a 300×1 vector b where each entry was selected independently from a standard normal distribution, and we set $x_* = A^\dagger b$, so that x_* is the least-squares solution to the problem (3). In the experiment, we took $s = 120, 60, 40, 30$. After we had A, x_*, b and T as inputs for our experiment, we randomly selected T columns from the matrix A to form a submatrix A_τ. We updated iterate $\{x_t : t \geq 0\}$ using Algorithm 2, and we stopped after 1000 iterations. For both consistent and inconsistent systems, at each iteration, we recorded the error $\|x_t - x_*\|_2$. We then repeated the procedure 50 times and took the average of all the errors.

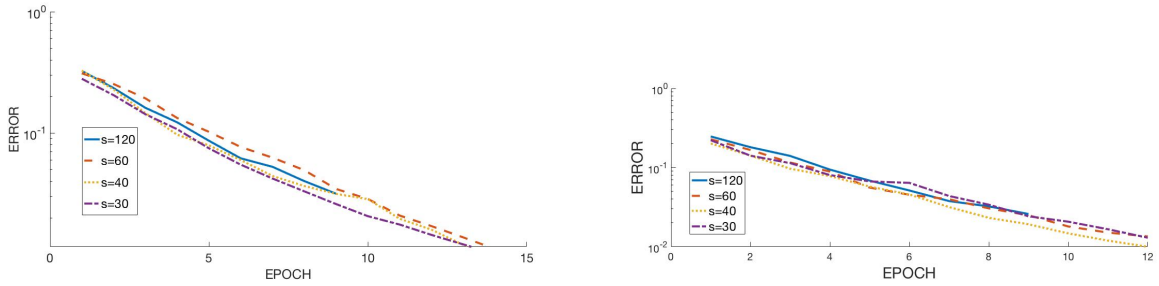

FIGURE 1. (RBGS method applied to consistent vs inconsistent matrix) The random matrix A is 300×120. The error $\|x_t - x_*\|_2$ is plotted against epochs for various values of partition size s. Left: consistent system $Ax = b$. Right: inconsistent system $Ax = b + r$.

The left panel in Figure 1 demonstrates convergence for the RBGS method when $s = 120$, 60, 40, 30 was applied to the consistent system, and the right panel demonstrates convergence when $s = 120, 60, 40, 30$ was applied to the inconsistent system. In both panels, the error was plotted against the number of epochs. The case $s = n = 120$ is used as RGS so that we can better observe RBGS's improvement on the convergence rate. In the figure, it is clear that as s decreases, the convergence rates didn't change significantly. In other words, RGS and RBGS show a similar rate of convergence, which is coherent with our discussion in section 2.2.

In the second experiment, we wanted to see how the size of the partition could change the operation time for both consistent and inconsistent systems. We created A, x_* and b the same way as in the first experiment but change the size of A to 300×100 and the size of x and b accordingly. In addition, for more noticeable differences in the implementation speed, we used a different variation of s, $s = 100, 20, 10, \frac{10}{3}$. We used the same update rule, and we stopped after 1000 iterations, recording the CPU time for each. We then repeated the procedure 50 times and took the average of both the error vectors and the time vectors to eliminate discrepancies.

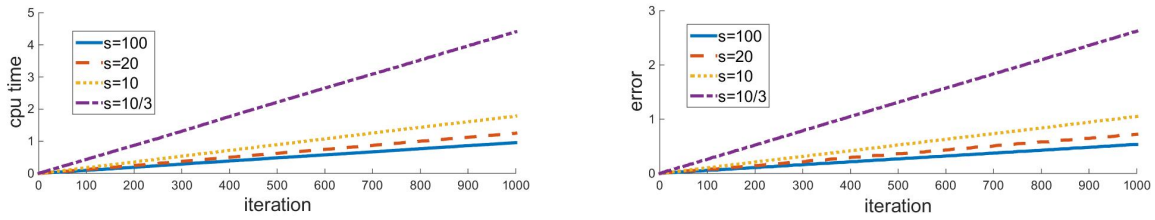

FIGURE 2. (RBGS method applied to consistent vs inconsistent matrix) The matrix A is 300×100 Gaussian. The error $\|x_t - x_*\|_2$ is plotted against CPU time in seconds. for various values of partition size s. Left: consistent system $Ax = b$. Right: inconsistent system $Ax = b + r$.

Figure 2 shows the CPU time per iteration of the RBGS method for various block sizes. These plots confirm that larger block sizes require more computation per iteration, as expected. One may then question if there is a gain in convergence speed using larger blocks. Fortunately, due to fast matrix multiplies, the increase in computation cost per iteration is overcome by the increase in convergence rate of the entire method, as we see in the next experiment.

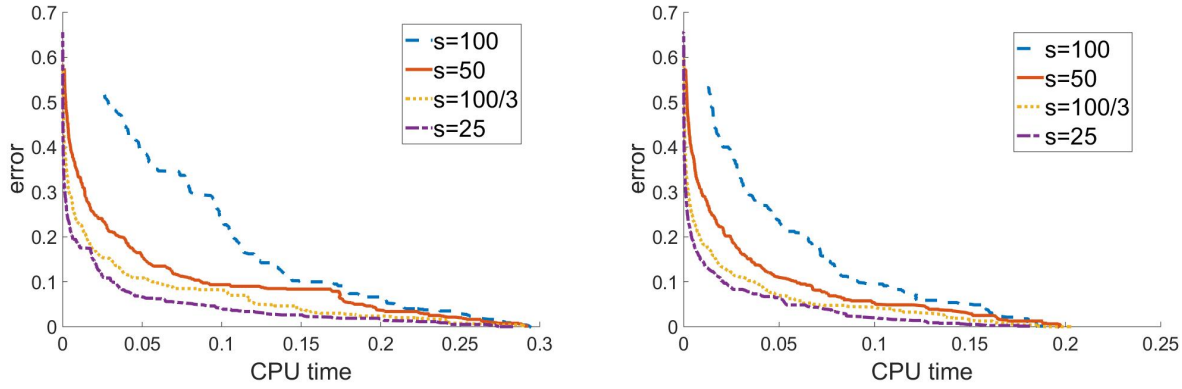

FIGURE 3. (RBGS method applied to consistent vs inconsistent matrix) The matrix A is 300×100 Gaussian. The error $\|x_t - x_*\|_2$ is plotted against CPU time in seconds for various values of partition size s. Left: consistent system $Ax = b$. Right: inconsistent system $Ax = b + r$.

Figure 3 demonstrates the error as a function of total CPU time when RBGS was applied to a consistent and inconsistent matrix. Since no claim is made about the linear rate of convergence against the operation time, the graphs are displayed in linear rather than logarithmic scale. We see that larger blocks do indeed correspond to lower overall runtime, motivating the use of the

block method. This observation, although not guaranteed mathematically, is consistent with empirical evidence for other types of block methods (discussed above).

Finally, we tested the usefulness of the RBGS algorithm with real data on wine quality and bike rental data. Both data sets are obtained from the UCI Machine Learning Repository [Lic13]. The wine data set is a sample of $m = 1599$ red wines with $n = 11$ physio-chemical properties of each wine, which gives us an $m \times n$ matrix A. The entries correspond to the amount of each physio-chemical present in the Portuguese "Vinho Verde" wine. The bike data contains hourly counts of rental bikes in a bike share system. There are $m = 17379$ samples and $n = 9$ attributes, including weather and seasonal data.

In each data source, the matrix A and the target vector b are given, and we want to find the solutions to the system (3). As the size of A varies in each data set, we used partitions whose blocks A_τ contain a fixed T columns. Due to the capacity of the computers used in the experiments, T was set for 1,2,4 and 10, where $T = 1$ is the same as simple RGS. We updated iterate $\{x_t : t \geq 0\}$ using Algorithm 2, and we stopped after 1000 iterations. Without ground truth, we recorded the errors $\|Ax_t - b\|_2$. We then repeated the procedure for 50 times and took the average of all the errors.

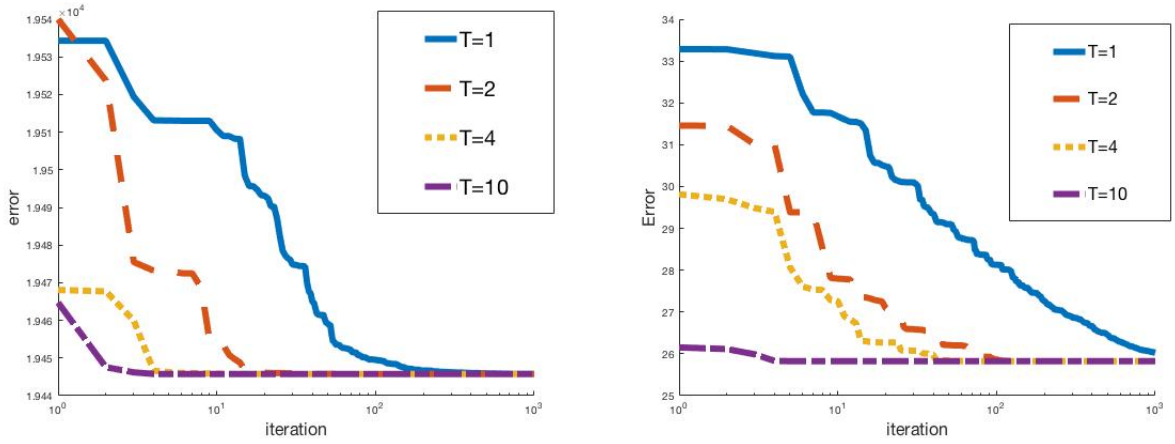

FIGURE 4. (RBGS method applied to real data) The approximation error $\|Ax_t - b\|_2^2$ is plotted against the number of iterations. Convergence for block sizes $T = 1,2,4,10$ are displayed. Left: approximation error as a function of the number of iterations for the bike data. Right: approximation error as a function of the number of iterations for the wine data.

The left panel in Figure 4 demonstrates convergence for the RBGS method when $T = 1,2,4,10$ was applied to the bike problem, and the right panel demonstrates convergence when $T = 1,2,4,10$ was applied to the wine problem. In both panels, the error was plotted against the number of iterations. The case $T = 1$ is used as a basic case so that we can better observe RBGS's improvement on the convergence rate. In Figure 4, it is clear that as T increases (s decreases), the convergence rates become significantly faster for both real problems, although the linear convergence is less consistent compared to the computer generated systems.

4. FUTURE DIRECTIONS

There are still many interesting open questions associated with Algorithm 2 that were not fully addressed in this paper. Firstly, in Algorithm 2, when the columns are already paved, each

block A_τ is selected uniformly for each iteration because we standardized the matrix A. For non-standardized matrices, there exist many other ways to select the blocks that may improve the rate of convergence and even reduce the dependence on characteristics of A (e.g.$\sigma^2_{\min}(A)$). In addition, when choosing A_τ, it is highly possible that selecting *with* versus *without* replacement might affect the convergence rate. In the experiments and convergence analysis, the blocks are all chosen with replacement until every block has been used at least once. However, it has been well observed that selecting without replacement often yields improved performance [RR12].

Secondly, in the experiments described in Section 3, the columns are paved using a simple random partition described and discussed by Needell and Tropp [NT13]:

Definition 2. *(Random Partition) Suppose that π is a permutation on $\{1,2,...,n\}$, chosen uniformly at random. In each iteration, define the set*

$$\tau_i = \{\pi(k) : k = \lfloor (i-1)n/m \rfloor + 1, \lfloor (i-1)n/m \rfloor + 2, ..., \lfloor in/m \rfloor\}.$$

It is clear that $T = \tau_1, ..., \tau_m$ is a partition of $\{1,...,n\}$ into m blocks of approximately equal sizes. In the experiments, we used the identical permutation $\pi(i) = i$ for all the iterations after the blocks are exhausted for the first round. It is possible that if we chose a different permutation for each round while retaining the same partition characteristics (s, α, β), the convergence rate could be improved.

Thirdly, recall that in our main result, to get greater convergence rate, the size of the partition should be as small as possible. In fact, in the optimally ideal case, we want to take the pseudo-inverse of A to get the least-squares solution in one single step. However, for a large system, it will be computationally impossible to invert the entire matrix and we have to take one or several columns at a time, which inspires the idea of RGS and RBGS. Similarly, when s is too small, it will be computationally expensive to take the pseudo-inverse of A_τ. Although we want to achieve the desired error bound in the least iterations and the least operation time, the two things cannot be achieved at the same time. Depending on the operational performance of the devices used to implement the algorithm, the size s should be adjusted accordingly to achieve a balance in both factors.

ACKNOWLEDGEMENTS

We would like to thank Anna Ma for fruitful discussions that helped improve this manuscript, and to Winston Ou as a second reader of the thesis that sparked this paper. We would also like to thank the reviewers and editor who handled this paper for their helpful suggestions.

REFERENCES

[CEG83] Y. Censor, P. P. Eggermont, and D. Gordon. Strong underrelaxation in Kaczmarz's method for inconsistent systems. *Numer. Math.*, 41(1):83–92, 1983.

[CZ97] Y. Censor and S. A. Zenios. *Parallel optimization: Theory, algorithms, and applications*. Oxford University Press on Demand, 1997.

[Elf80] T. Elfving. Block-iterative methods for consistent and inconsistent linear equations. *Numer. Math.*, 35(1):1–12, 1980. NUMMA7; 65F10; 583651 (83e:65059).

[GBH70] 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.

[HN90] M. Hanke and W. Niethammer. On the acceleration of kaczmarz's method for inconsistent linear systems. *Linear Alg. Appl.*, 130:83–98, 1990.

[HNR17] A. Hefny, D. Needell, and A. Ramdas. Rows vs. columns: Randomized kaczmarz or gauss-seidel for ridge regression. *SIAM J. Sci. Comput.*, 2017. To appear.

[Kac37] S. Kaczmarz. Angenäherte auflösung von systemen linearer gleichungen. *Bulletin International del Academie Polonaise des Sciences et des Lettres*, 35:355–357, 1937.

[Lic13] M. Lichman. UCI machine learning repository, 2013.

[LL10] D. Leventhal and A. S. Lewis. Randomized methods for linear constraints: convergence rates and conditioning. *Math. Oper. Res.*, 35(3):641–654, 2010. 65F10 (15A39 65K05 90C25); 2724068 (2012a:65083); Raimundo J. B. de Sampaio.

[MNR15] A. Ma, D. Needell, and A. Ramdas. Convergence properties of the randomized extended Gauss-Seidel and Kaczmarz methods. *SIAM J. Matrix Anal. A.*, 36(4):1590–1604, 2015.

[Nat01] F. Natterer. *The mathematics of computerized tomography*, volume 32. Society for Industrial and Applied Mathematics, Philadelphia, PA; SIAM, 2001. 00A69 (44A12 65R10 68U99 92C55); 1847845 (2002e:00008); Fritz Keinert; Reprint of the 1986 original.

[Nee10] D. Needell. Randomized Kaczmarz solver for noisy linear systems. *BIT*, 50(2):395–403, 2010.

[NT13] D. Needell and J. A. Tropp. Paved with good intentions: Analysis of a randomized block kaczmarz method. *Linear Alg. Appl.*, 2013.

[Pop98] C. Popa. Extensions of block-projections methods with relaxation parameters to inconsistent and rank-deficient least-squares problems. *BIT*, 38(1):151–176, 1998. 65F20; 1621092 (99d:65121); W. C. Rheinboldt.

[RR12] B. Recht and C. Ré. Beneath the valley of the noncommutative arithmetic–geometric mean inequality: Conjectures, case studies, and consequences. In *Proc. 25th Ann. Conf. Learning Theory*, Edinburgh, June 2012.

[SV09] T. Strohmer and R. Vershynin. A randomized kaczmarz algorithm with exponential convergence. *J. Fourier Anal. Appl.*, 15:262–278, 2009.

[Tan71] K. Tanabe. Projection method for solving a singular system of linear equations and its applications. *Numer. Math.*, 17(3):203–214, 1971.

[Tro09] J. A. Tropp. Column subset selection, matrix factorization, and eigenvalue optimization. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 978–986, Philadelphia, PA, 2009. SIAM. 65F30 (15A18); 2807539 (2012i:65079).

[Ver06] R. Vershynin. *Random sets of isomorphism of linear operators on Hilbert space*, volume 51 of *High dimensional probability*, pages 148–154. Inst. Math. Statist, Beachwood, OH, 2006. 46B09 (46B07 46B20); 2387766 (2009h:46023); Dirk Werner.

[WM67] T. M. Whitney and R. K. Meany. Two algorithms related to the method of steepest descent. *SIAM J. Numer. Anal.*, 4(1):109–118, 1967.

[Xu18] Y. Xu. Hybrid jacobian and gauss–seidel proximal block coordinate update methods for linearly constrained convex programming. *SIAM J. Optimization*, 28(1):646–670, 2018.

[ZF13] A. Zouzias and N. M. Freris. Randomized extended kaczmarz for solving least squares. *SIAM J. Matrix Anal. A.*, 34(2):773–793, 2013.