2017

Paving the Randomized Gauss-Seidel

Wei Wu
Scripps College

Paving the Randomized Gauss-Seidel Method

Wei (Maggie) Wu

Professor Deanna Needell
Professor Winston Ou

Submitted to Scripps College in partial fulfillment of
the degree of Bachelor of Arts

March 10, 2017

Department of Mathematics

Abstract

The Randomized Gauss-Seidel Method (RGS) is an iterative algorithm that solves overdetermined systems of linear equations $Ax = b$. This paper studies an update on 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 a Randomized Block Gauss-Seidel Method (RBGS) which uses a randomized control method to choose a subset of columns of A at each step. This algorithm is the first block RGS method with an expected linear convergence rate which can be described by the properties of the matrix A and its column submatrices. The analysis demonstrates that RBGS improves RGS more when given 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 simple RGS method.

Contents

1 Introduction

The Randomized Gauss-Seidel Method (RGS)[LL10] is an iterative algorithm for solving linear systems of equations, and is most effective when the system is over-constrained. 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). In RK, each iteration projects the estimate from one constraint to the other, converging to the least-squares solution also at a linear rate. The RK algorithm[SV09] has been updated to the block Kaczmarz algorithm[NT13]. While the simple Kaczmarz algorithm enforces one single constraint at each iteration, the block update enforces multiple constraints simultaneously at each iteration. Inspired by the block Kaczmarz algorithm, this paper attempts to demonstrate that the block method can also be applied to the Randomized Gauss-Seidel Method to increase the convergence rate.

1.1 Model and Notation

Consider a system of linear equations

$$Ax = b \tag{1}$$

where A is a real $m \times n$ matrix. The ℓ_p vector norm for $p \in [1, \infty]$ is denoted $\|\cdot\|_p$. $\|\cdot\|$ denotes the spectral norm and $\|\cdot\|_F$ denotes the Frobenius norm. The set of columns is denoted $\{A^1, A^2, ..., A^n\}$, and the set of rows is denoted $\{A_1, A_2, ..., A_m\}$. At the same time, A_τ denotes the submatrix of A indexed by a set τ. In the block RK, A_τ denotes the row submatrix; in RBGS, A_τ denotes the column submatrix.

For a Hermitian matrix, λ_{min} and λ_{max} are the algebraic minimum and maximum eigenvalues. For an $m \times n$ matrix A, the singular values are arranged such that

$$\sigma_{max}(A) := \sigma_1(A) \geq \sigma_2(A) \geq ... \geq \sigma_{min\{m,n\}}(A) =: \sigma_{min}(A). \tag{2}$$

We also define the condition number $\kappa(A) := \sigma_{max}(A)/\sigma_{min}(A)$. The transpose of a matrix A is denoted A^T, and the adjoint of a matrix A is denoted A^*. The Moore-Penrose pseudoinverse of matrix A is denoted $A^\dagger$. When matrix A has linearly independent columns, the pseudoinverse $A^\dagger := (A^*A)^{-1}A^*$.

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

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

To solve the linear system (1), we now consider solving the overdetermined least-squares problem over x:

$$\text{minimize } \|Ax - b\|_2^2, \tag{3}$$

which is the same as minimizing $L(x) = \frac{1}{2}\|b - Ax\|_2^2$ and as finding the least-squares solution to a linear system.

To standardize the following discussion, we define the following two notations.

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

Definition 2. *To solve the system* (1)*,* x_0 *is defined as the starting point in an algorithm .*

1.2 Related Works

1.2.1 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 $L(x) = \frac{1}{2}\|b - Ax\|_2^2$ with respect to the selected coordinate to get

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

where $e_{(j)}$ refers to the jth coordinate basis column vector, which has all 1 in the jth position and 0 in the rest of positions.

Leventhal and Lewis [LL10] shows that this algorithm has an expected linear convergence rate, which I will further elaborate on in Chapter 2.

1.2.2 The Randomized Kaczmarz Method (RK)

The RK Method is first proposed in [SV09]. At each iteration, the RK method projects the current state orthogonally onto the solution hyperplane $\langle A_i, x\rangle = b_i$. The algorithm is described as

$$x_{t+1} = x_t + \frac{b_i - \langle A_i, x_t\rangle}{\|A_i\|_2^2} A_i, \tag{6}$$

where A_i denotes the ith row of matrix A and is selected with probability proportional to their norms. When we assume standardized matrix A, each row is selected uniformly at random. Vershynin and Strohmer [SV09] prove 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,

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

1.2.3 The Block Randomized Kaczmarz with road paving condition

Elfving and Eggermont [Elf80] update the RK method to the block RK method. The method 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 state is projected orthogonally onto the solution space $A_{\tau_i}x = b_{\tau_i}$. The algorithm is described as

$$x_{t+1} = x_t + (A_{\tau_i})^\dagger(b_{\tau_i} - A_{\tau_i}x_t) \tag{7}$$

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

To prove convergence, the block RK method is further restricted by Needell and Tropp [NT13] using the idea of row paving. A row paving (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 restricts the lower and upper bound of the eigenvalues of the partitioned subsets. 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_* defined in Section 1.1.

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

1.3 Contribution: The Randomized Block Gauss-Seidel (RBGS)

1.3.1 Deriving the Algorithm

This paper applies the row paving idea to RGS. However, on the 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 greedily minimizing through multiple directions at the same time.

Given system (1) and a partition $\tau \in P = \tau_{\{1,...s\}}$, 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 simple RGS method, we presuppose

$$x_{t+1} = x_t + \sum_{k=1}^{T} \alpha_k e_k, \tag{8}$$

where T is the number of coordinates included in the selected τ. The set $\tau = \{c_1, c_2, ..., c_T\}$, where $\forall c_i$ is a column index for the partitioned subset. The set $\{e_1, e_2, ..., e_T\}$ is one-to-one

with the set τ, and $\forall i \in \{1, 2, ..., T\}$, e_i is the c_ith coordinate basis column vector. 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 $L(x_{t+1}) = \frac{1}{2}\|b - Ax_{t+1}\|_2^2$, and

$$\min_{\alpha_1,...,\alpha_T} L(x_{t+1}) = \min_{\alpha_1,...,\alpha_T} \frac{1}{2}\left(\sum_{i=1}^{m}\langle A_i, x_t + \sum_{k=1}^{T}\alpha_k e_k\rangle - b_i\right)^2. \tag{9}$$

Take the derivative of $L(x_{t+1})$ with respect to $\forall\alpha_u$ and set it to zero, we find that $\|A^u\|_2^2\alpha_u = \langle r_t, A^u\rangle - \sum_{k\neq u}^{T}\alpha_k\langle A^k, A^u\rangle$. To solve (9), we now have to solve the system of equations

$$\forall u \in \{1, ..., T\}, \sum_{i=1}^{T}\alpha_i\langle A^u, A^i\rangle = \langle r_t, A^u\rangle. \tag{10}$$

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

$$A_\tau^T A_\tau \alpha^T = (r_t^T A_\tau)^T, \text{where } \alpha \text{ denotes the vector } (\alpha_1, ..., \alpha_T), \tag{11}$$

and A_τ denotes the column submatrix of A indexed by τ. Since A has only real entries, $A^T = A^*$ and $(r_t^T A_\tau)^T = (r_t^* A_\tau)^*$; then we can solve (11) and get

$$\alpha^T = (A_\tau^* A_\tau)^{-1}(r_t^* A_\tau)^* = (A_\tau^* A_\tau)^{-1}A_\tau^* r_t = A_\tau^\dagger r_t. \tag{12}$$

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

$$\alpha \leq \lambda_{min}(A_\tau^\dagger A_\tau^{\dagger *}) = \sigma_{min}^2(A_\tau^\dagger) \text{ and } \sigma_{max}^2(A_\tau^\dagger) = \lambda_{max}(A_\tau^\dagger A_\tau^{\dagger *}) \leq \beta \text{ for each } \tau \in P \tag{13}$$

where s is called the size of the partition, and α and β determines the lower and upper bound. Notice that $\alpha = 0$ unless $A_\tau^\dagger$ is a "fat" submatrix which has more columns than rows.

Secondly, at each iteration, independent of all previous choice, we select a block τ uniformly at random from the partition P. These two decisions lead to Algorithm 1 described in the following subsection.

1.3.2 The Algorithm

Consider the system (1) with $m > n$. Let x_* be the unique minimizer for (3), which is also the least squares solution to the overdetermined system. The algorithm for the RBGS method is described by the following code. Similar to the RGS method, the RBGS Method iteratively improves the approximation by adjusting the value of multiple coordinates (adding the vector α to the current state), which finally converges to the least square solution x_*.

Algorithm 1 The Randomized Block Gauss-Seidel (RBGS)

procedure (A, b, T,$h$maxIter) ▷ $m \times n$ matrix A, $b \in \mathbb{C}^m$, $T = \frac{n}{s}$ the number of coordinates in each block τ, maximum iterations h
    Initialize $x_0 = \mathbf{0}$, $r_0 = b - Ax_0$
    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 a $n \times T$ 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

1.4 Organization

Chapter 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. Chapter 3 analyzes the results of the experiments comparing RBGS with different sizes of partitions applied to both consistent and inconsistent systems. Chapter 4 concludes the paper and indicates further questions and conditions not fully addressed in the paper.

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 1. *Given a standardized real $m \times n$ matrix A and a $m \times 1$ vector b. Let P be a column partition (s, α, β) as defined in* (13)*. 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:

$$\mathbb{E}\|x_t - x_*\|_2^2 \leq \kappa^2(A)\gamma^t \|x_0 - x_*\|_2^2, \tag{14}$$

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 noticing that page 6 in [MNR15] has demonstrated that when we are 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 (9), $\forall u \in \tau$, $\frac{\partial L(x_t)}{\partial A_u} = 0$. 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:

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

Take $v = A_\tau^\dagger u$.

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

This proof is inspired by [NT13].

■

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

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 $A_\tau^T r_* = 0$

Since A only has real entries, $A_\tau^* = A_\tau^T$. 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:

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

$$\mathbb{E}\|Ax_t - Ax_*\|_2^2 = \|Ax_{t-1} - Ax_*\|_2^2 - \mathbb{E}\|Ax_t - Ax_{t-1}\|_2^2 \tag{15}$$

Using the algorithm defined in Theorem 1,

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

where the third equation is based upon Definition 1.

We then apply Lemma 3

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

and

$$\mathbb{E}(\|A(x_t - x_{t-1})\|_2^2) = \mathbb{E}(\|A_\tau A_\tau^\dagger (Ax_* - Ax_{t-1})\|_2^2 \tag{16}$$

If we combine (15) and (16), we have

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

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

$$\mathbb{E}\|A_\tau A_\tau^\dagger (Ax_* - Ax_{t-1})\|_2^2 \geq \frac{\alpha \sigma_{min}^2(A)}{s} \|Ax_* - Ax_{t-1}\|_2^2. \tag{18}$$

Now plug in (18) to (17), 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, take $\gamma = 1 - \frac{\alpha \sigma_{min}^2(A)}{s}$, 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, compare RBGS Method with the block RK Method. When the matrix A is given, in other words 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). At the same time, for inconsistent systems, the block RK Method will ave a convergence horizon. When restricting the partition, we want a small s and a large α for 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, because this constant will also affect the convergence rate.

Second, we compare Theorem 1 with the results of simple RGS method. [LL10] only discusses the case when the system $Ax = b$ is consistent. Their algorithm update (5) converges linearly in expectation to a least-squares solution, and their result is as follows.

Theorem 2. *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*

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

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

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

Notice that when the system $Ax = b$ has solutions, $b = Ax_*$ and $f(x_*) = 0$. In addition, $A^T A = A^* A$ when A has only real entries. The above inequality can be reinterpreted as

$$\mathbb{E}\left[\|Ax_t - Ax_*\|_2^2\right] \leq (1 - \gamma_1)\|Ax_{t-1} - Ax_*\|_2^2 \tag{19}$$

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

To better compare Theorem 2 with the main result from Theorem 1, we have to first extend a corollary from Theorem 1 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:*

$$minimize \ \|Ax - b\|_2^2.$$

Let x_* be the unique minimizer. Let $f(x)$ and $\delta(x)$ be as defined in Theorem 2. Let α be the lower bound of the partition P defined in (13)*. 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} denotes the convergence rate of the simple RGS and RBGS respectively, then we have $\rho_{simple} \geq \gamma_1$ and $\rho_{block} \geq \gamma_2$. 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 the every subset partitioned in RBGS only contain one column, its convergence rate the same as the convergence rate of the simple RGS.

However, beyond the basic case, RBGS significantly improves the convergence rate. When we have partition $P = (s, \alpha, \beta)$, $\rho_{block} \geq \frac{\alpha\sigma^2_{min}(A)}{s}$. 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 much 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 1 is computing $A^\dagger_\tau$, and containing as much columns as possible in A_τ might significantly lower the computational speed of Algorithm 1. There should be a balance between fast convergence rate and fast implementation speed.

3 Experiments

To test our algorithm, we used MATLAB to run experiments, using random matrices to test the convergence of the RBGS method applied to overdetermined systems of equations. In each experiment, we created a random 300×100 standardized matrix A, with each entry of A selected independently from a unit normal distribution.

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 100×1 vector x_* where each entry was selected independently from a unit normal distribution, and we set $b = Ax_*$. 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 unit normal distribution, and we set $x_* = A^{\dagger}b$, and x_* is also the least-squares solution to the problem (3). In the experiment, we took $s = 100, 50, \frac{100}{3}, 25$, so that each block would contain T=1, 2, 3 or 4 columns. After we had A, x_*, b and T as inputs for our experiment, we randomly selected T columns from matrix A and form a submatrix A_τ. We updated iterate $\{x_t : t \geq 0\}$ using Algorithm 1, and we stopped after 1000 iterations. For both consistent and inconsistent systems, at each iteration, we recorded the error vector $err = \|x_t - x_*\|_2$. We then repeated the procedure for 50 times and took the average of all the error vectors to eliminate discrepancies. See the appendix for the MATLAB code for this experiment.

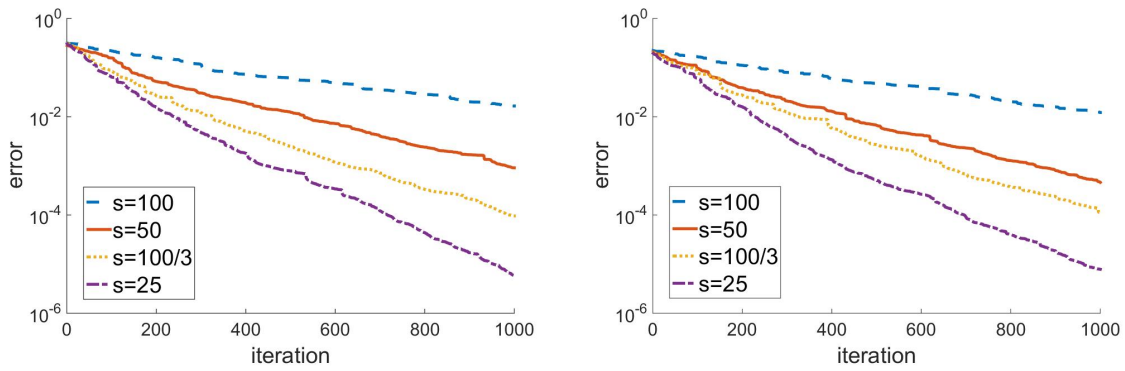

Figure 1: (RBGS method applied to consistent vs inconsistent matrix) The matrix A is 300×100.The error $\|x_t - x_*\|_2$ is plotted against the number of iterations. Convergence in $s = 100, 50, \frac{100}{3}, 25$ are displayed respectively in blue, orange, yellow and purple in both graphs. The lines represent the mean of 50 trials.[Left] Approximation error as a function of the number of iterations for consistent system $Ax = b$. [Right] Error as a function of the number of iterations for inconsistent system $Ax = b$.

The left panel in Figure 1 demonstrates convergence for the RBGS method when $s = 100, 50, \frac{100}{3}, 25$ was applied to the consistent system, and the right panel demonstrates convergence when $s = 100, 50, \frac{100}{3}, 25$ was applied to the inconsistent system. In both panels, the error was plotted against the number of iterations. The case $s = n = 100$ is used as a basic case 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 become significantly faster for both consistent and inconsistent systems. The improvement is consistent throughout

the 1000 iterations. In addition, Algorithm 1 does not behave differently for consistent or inconsistent systems, and there is no indication of the existence of a convergence horizon, which is coherent with our main results.

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. We also used the same variation of s and the same update rule, and we stopped after 1000 iterations. Before we started the algorithm, we recorded the CPU time in MATLAB. After each iteration, we subtracted the initial time from the current time to get the cumulative operation time the algorithm used, and the time vector was stored. We then repeated the procedure 50 times and took the average of both the error vectors and the time vectors to eliminate discrepancies. See the appendix for the MATLAB code for this experiment.

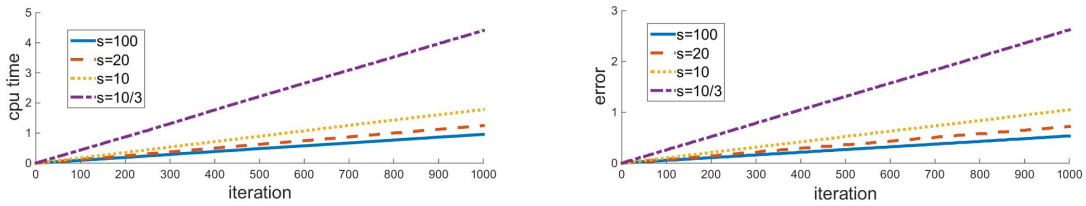

Figure 2: (RBGS method applied to consistent vs inconsistent matrix) The matrix A is 300×100.The error $\|x_t - x_*\|_2$ is plotted against CPU time in seconds. Convergence in $s = 100, 20, 10, \frac{10}{3}$ are displayed respectively in blue, orange, yellow and purple in both graphs. The lines represent the mean of 50 trials.[Left] Approximation error as a function of CPU time for consistent system $Ax = b$. [Right] Approximation error as a function of CPU time for inconsistent system $Ax = b$.

The left panel of Figure 2 demonstrates convergence for the RBGS method when $s = 100, 20, 10, \frac{10}{3}$ was applied to the consistent system, and the right panel demonstrates convergence when $s = 100, 20, 10, \frac{10}{3}$ was applied to the inconsistent system. The CPU time was plotted against iteration. Since no claim is made about the linear rate of convergence against the operation time, the graphs are not converted to demonstrate convergence rate linearly. It is clear that as s becomes smaller, the slope of the corresponding line becomes significantly larger, which means that the average time taken in each iteration becomes larger. This result is consistent with the assertion in section 2.2 that the smaller the partition is, the longer the implementing time will be. However, the algorithm is only tested on one computer. The operation time can vary greatly depending on the ability of the software and hardware of the computer or other mathematical instruments, which is beyond the scope of discussion of this paper.

Finally, we tested the usefulness of the RBGS algorithm with data from real life on wine quality and bike rental data. Both data sets are obtained from the UCI Machine Learning Repository. The wine data set is a sample of $m = 1599$ red wines with $n = 11$ physiochemical properties of each wine, which gives us an $m * n$ matrix A. 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, per sample, which gives us an $m * n$ matrix A.

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 different data set, we applied partition whose each blocks of A_τ contains 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 1, and we stopped after 1000 iterations. As x_* is hard to calculate, at each iteration, we recorded the error vector $err = \|Ax_t - b\|_2$. We then repeated the procedure for 50 times and took the average of all the error vectors to eliminate discrepancies. See the appendix for the MATLAB code for this experiment.

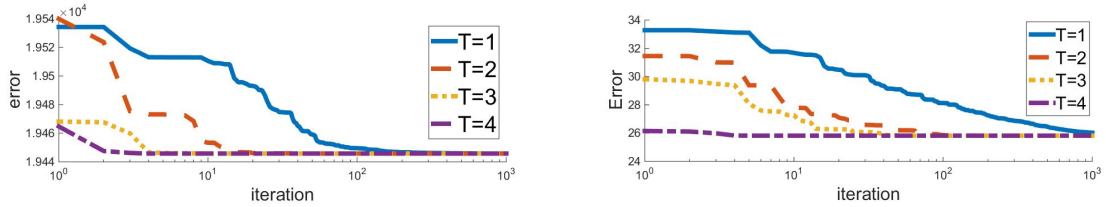

Figure 3: (RBGS method applied to real data) The matrix A is 300×100.The approximation error $\|Ax_t - b\|_2^2$ is plotted against the number of iterations. Convergence in $T = 1, 2, 4, 10$ are displayed respectively in blue, orange, yellow and purple in both graphs. The lines represent the mean of 50 trials. [Left] Approximation error as a function of the number of iterations for the bike problem $Ax = b$. [Right] Approximation error as a function of the number of iterations for the wine problem $Ax = b$.

The left panel in Figure 3 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 3, 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 system.

4 Future Directions

There are still many interesting open questions associated with Algorithm 1 that were not fully addressed in this paper. Firstly, in Algorithm 1, 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 $A(\sigma^2_{min}(A))$. In addition, when choosing A_τ, it is highly possible that choosing with or without replacement might affect the convergence rate. In the experiments, the blocks are all chosen with replacement until every block has been used at least once. The main result did not separate the two selecting methods either.

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

Definition 3. *(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 when $s = 100, 50, 25$. However, when $s = \frac{100}{3}$, since 3 doesn't divides 100, every round after the columns are exhausted, another permutation was implemented. It is not clear on the graphs that $s = \frac{100}{3}$ improves the convergence rate more than other values, because we didn't control the variables. It is possible that if we choose 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 arithmetically 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.

5 Appendix

1. MATLAB code: the Randomized Gauss-Seidel Method

```
1.  function[err1]=rgs(A,b,x);
2.  [m,n]=size(A);
3.  xest=randn(n,1);
4.  for t=1:10000
5.      j=datasample(1:n,1);
6.      aj=A(:,j);
7.      e=zeros(n,1);
8.      e(j)=1;
9.      xest=xest+aj'*(b-A*xest)*e/norm(aj)^2;
10.     err1(t)=norm(x-xest);
11. end
12. plot(err1,'b')
```

2. MATLAB code: the Randomized Block Gauss-Seidel Method: RBGS

```
1.  function[err2]=rrp1205(A,b,x,T)
2.  [m,n]=size(A);
3.  xest=randn(n,1);
4.  ER=zeros(100,30);
5.  for t=1:10000
6.      E=zeros(n,T);
7.      r=b-A*xest;
8.      j=datasample([1:n],T);
9.      Aj=A(:,j);
10.      for i=1:T
11.          E(j(i),i)=1;
12.      end
13.     xest=xest+E*pinv(Aj)*r;
14.     err2(t)=norm(x-xest);
15. end
16. plot(err2)
```

3. MATLAB code: Experiment 1: taking the average of fifty trials of the Randomized Block Gauss-Seidel Method plotted against iterations

```
1.  function[err2]=rrp0228(A,b,x,T)
2.  [m,n]=size(A);
3.  xest=randn(n,1);
4.  ER=zeros(50,1000);
5.  for trial=1:50
6.   for t=1:1000
7.      E=zeros(n,T);
8.      r=b-A*xest;
9.      j=datasample([1:n],T);
```

```
10.       Aj=A(:,j);
11.       for i=1:T
12.          E(j(i),i)=1;
13.       end
14.      xest=xest+E*pinv(Aj)*r;
15.      err2(t)=norm(x-xest);
16.    end
17.  ER(trial,:)=err2;
18.  end
19.  e=mean(ER);
20.  plot(e);
```

4. MATLAB code: Experiment 2: taking the average of fifty trials of the Randomized Block Gauss-Seidel Method plotted against CPU time

```
1.  function[err2]=cpu(A,b,x,T)
2.  [m,n]=size(A);
3.  xest=randn(n,1);
4.  ER=zeros(50,1000);
5.  CPU=zeros(50,1000);
6.  for trial=1:50
7.       t0=cputime;
8.       for t=1:1000
9.         E=zeros(n,T);
10.        r=b-A*xest;
11.        j=datasample([1:n],T);
12.        Aj=A(:,j);
13.        for i=1:T
14.          E(j(i),i)=1;
15.        end
16.        xest=xest+E*pinv(Aj)*r;
17.        err2(t)=norm(x-xest);
18.        duration(t)=cputime-t0;
19.      end
20.  ER(trial,:)=err2;
21.  CPU(trial,:)=duration;
22.  end
23.  e=mean(ER);
24.  plot(e,mean(CPU));
```

5. MATLAB code: Experiment 3: taking the average of fifty trials of the Randomized Block Gauss-Seidel Method with real data

```
1.  function[err2]=realrrp(A,b,T)
2.  [m,n]=size(A);
3.  xest=randn(n,1);
4.  ER=zeros(50,1000);
```

```
5. for trial=1:50
6. for t=1:1000
7.      E=zeros(n,T);
8.      r=b-A*xest;
9.      j=datasample([1:n],T);
10.     Aj=A(:,j);
11.     for i=1:T
12.         E(j(i),i)=1;
13.     end
14.         xest=xest+E*pinv(Aj)*r;
15. err2(t)=norm(b-A*xest);
16. end
17. ER(trial,:)=err2;
18. end
19. e=mean(ER);
20. plot(e);
```

References

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

[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] Anna Ma, Deanna Needell, and Aaditya Ramdas. Convergence properties of the randomized extended gauss-seidel and kaczmarz methods. *SIAM J. Matrix Anal. A.*, 36(4):1590–1604, 2015.

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

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

[Tro09] Joel 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.