A Doubly Stochastic Gauss-Seidel Algorithm for Solving Linear Equations and Certain Convex Minimization Problems

M. Razaviyayn,∗ M. Hong,∗ N. Reyhanian,
Z.-Q. Luo

Submitted in April 2018

Abstract Consider the classical problem of solving a general linear system of equations \( Ax = b \). It is well known that the (successively over relaxed) Gauss-Seidel scheme and many of its variants may not converge when \( A \) is neither diagonally dominant nor symmetric positive definite. Can we have a linearly convergent G-S type algorithm that works for any \( A \)? In this paper we answer this question affirmatively by proposing a doubly stochastic G-S algorithm that is provably linearly convergent (in the mean square error sense) for any feasible linear system of equations. The key in the algorithm design is to introduce a nonuniform double stochastic scheme for picking the equation and the variable in each update step as well as a stepsize rule. These techniques also generalize to certain iterative alternating projection algorithms for solving the linear feasibility problem \( Ax \leq b \) with an arbitrary \( A \), as well as certain high-dimensional convex minimization problems. Our results demonstrate that a carefully designed randomization scheme can make an otherwise divergent G-S algorithm converge.

∗ equal contributions. M. Razaviyayn is with the Department of Industrial and Systems Engineering, the University of Southern California. E-mail: razaviya@usc.edu; M. Hong and N. Reyhanian are with the Department of Electrical and Computer Engineering, University of Minnesota, USA. E-mail: {mhong, navid}@umn.edu; Z.-Q. Luo is with The Shenzhen Research Institute of Big Data, The Chinese University of Hong Kong, Shenzhen, China. E-mail: luozq@cuhk.edu.cn This research is supported by the NSFC grants 61731018 and 61571384, the Peacock project of SRIIBD.
1 Introduction: Solving Linear System of Equations

Consider the generic problem of solving a linear system of equations

$$ Ax = b, $$

where $A \in \mathbb{R}^{m \times n}$, $x, b \in \mathbb{R}^n$. We use $[n]$ and $[m]$ to denote the set $\{1, \ldots, n\}$ and $\{1, \ldots, m\}$, respectively. A classical approach to solve (1) is by the Gauss-Seidel (G-S) algorithm, whereby at each iteration only one variable is updated by using the information from only one equation. More precisely, let $A_j$, and $b_j$ denote the $j$th row of $A$ and $j$th element of $b$, respectively. We define the G-S type algorithm as follows.

**Definition 1** An iterative algorithm for solving (1) is of Gauss-Seidel type, if at each iteration the algorithm updates one variable $x_i$, $i \in [n]$, by utilizing only $(A_j, b_j)$ for some $j \in [m]$.

A natural application of the G-S type algorithms is in the setting where $n$ players play a game in which $x_i$ is the variable of player $i$. In this case we have $m = n$, and the objective of player $i$ is to satisfy the $i$-th equation

$$ \sum_{j=1}^{n} a_{ij} x_j = b_i. $$

The best response strategy for player $i$ is given by

$$ \hat{x}_i = b_i - \sum_{j \neq i} a_{ij} x_j. $$

Using a step-size $\alpha \geq 0$, this best response strategy leads to the following successive over-relaxation (SOR) update rule:

$$ x_i^{r+1} = (1 - \alpha)x_i^r + \alpha \frac{b_i - \sum_{j \neq i} a_{ij} x_j^r}{a_{ii}}, $$

where $a_{ij}$ is the $(i,j)$th element of $A$; $b_i$ is the $i$th element of $b$. The central question is how to choose the step-size $\alpha$ and determine the order in which the players update their variables so that the G-S type algorithm (3) will eventually lead to an equilibrium (or equivalently, a solution to (1)).

1.1 Background on the Convergence of G-S Type Algorithm

To better understand the convergence behavior of G-S algorithm and its variants, let us consider the following example.

**Example 1**: Consider the following $2 \times 2$ special case of (1):

$$ A = \begin{bmatrix} 1 & -\tau \\ -\tau & 1 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, $$

where $\tau > 1$ is some given constant. The best response strategy in (3) leads to the following update rule:

$$ x_1^{r+1} = (1 - \alpha)x_1^r + \alpha \tau x_2^r, \quad \text{when } x_1 \text{ is updated before } x_2, $$

$$ x_2^{r+1} = (1 - \alpha)x_2^r + \alpha \tau x_1^r, \quad \text{when } x_2 \text{ is updated before } x_1. $$

Assume \( x_1^0 = x_2^0 > 0 \).

Let us consider the following five different update rules which are all of the G-S type.

1. **Cyclic Successive Over Relaxation (SOR):** At iteration \( r + 1 \), we perform
\[
x_1^{r+1} = (1 - \alpha)x_1^r + \alpha \tau x_2^r, \quad x_2^{r+1} = (1 - \alpha)x_2^r + \alpha \tau x_1^{r+1}.
\] (6)

2. **Symmetric SOR:** At iteration \( r + 1 \), the variables are updated using a forward-sweep G-S step followed by a backward-sweep G-S step [10],
\[
x_1^{r+1/2} = (1 - \alpha)x_1^r + \alpha \tau x_2^r, \quad x_2^{r+1/2} = (1 - \alpha)x_2^r + \alpha \tau x_1^{r+1/2}.
\]
\[
x_2^{r+1/2} = (1 - \alpha)x_2^{r+1/2} + \alpha \tau x_1^{r+1/2}, \quad x_1^{r+1} = (1 - \alpha)x_1^{r+1/2} + \alpha \tau x_2^{r+1}.
\]

3. **Uniformly Randomized (UR) SOR:** At iteration \( r + 1 \), randomly pick one variable from \( \{x_1, x_2\} \) with equal probability. Update according to (4) or (5) based on which variables are selected, while fixing the other variable at its previous value.

4. **Non-Uniformly Randomized (NUR) SOR:** At iteration \( r + 1 \), let \( p_1^{r+1} > 0 \) and \( p_2^{r+1} > 0 \) satisfy \( p_1^{r+1} + p_2^{r+1} = 1 \); randomly pick \( x_i \) according to \( p_i^{r+1} \). Update according to (4) or (5) based on which variables are selected, while fixing the remaining variable at its previous value.

5. **Random Permutation (RP) SOR:** At iteration \( r + 1 \), randomly select a permutation \( \pi \) of the index set \( \{1, 2\} \); The variables are updated according to
\[
x_{\pi(1)}^{r+1} = (1 - \alpha)x_{\pi(1)}^r + \alpha \tau x_{\pi(2)}^r, \quad x_{\pi(2)}^{r+1} = (1 - \alpha)x_{\pi(2)}^r + \alpha \tau x_{\pi(1)}^{r+1}.
\] (7)

Note that this method is referred to as the **shuffled SOR** in [18].

It is easily seen that for any update order listed above, the resulting algorithm have the following property:
\[
\min \{x_1^r, x_2^r\} > \min \{x_1^0, x_2^0\} > 0, \quad \forall r, \forall \alpha > 0.
\]

On the other hand, the solution of the system of linear equation is \( x_1^* = x_2^* = 0 \); hence none of these algorithms will find the solution.

Next we give a brief literature review on the convergence analysis of the G-S type, as well as other related algorithms for solving a linear system of equations or inequalities.

**The convergence of G-S type algorithm.** It is well-known that when \( A \) is either diagonally dominant, or symmetric positive definite (PD), then the classical SOR method with cyclic update rule converges to the solution of [1] Proposition 6.7, 6.8, 6.10. More specifically, if \( A \) is symmetric and PD, the convergence of SOR [for any \( \alpha \in (0, 2) \)] can be established by showing that each iteration of the SOR algorithm is equivalent to a step of the coordinate descent (CD) algorithm for minimizing the strictly convex cost function \( \frac{1}{2} x^T Ax - b^T x \) [1] Section 2.6.3. Note that the convergence rate in this case is linear, although the rate is not easily expressible in terms of the condition number of matrix \( A \) if the classical cyclic rule is used [3]. Without the symmetry or the positive definiteness of \( A \), the convergence of the SOR algorithm is only known when \( A \) is diagonally dominant [1] Section 2.6.2. In particular, using the matrix splitting \( A = L + D + U \) where \( L, U, D \) are
the lower-triangular, upper-triangular and the diagonal part of $A$, respectively, we can write the SOR iteration as

$$x^{r+1} = (1 + \alpha D^{-1}L)^{-1} [(1 - \alpha)I - \alpha D^{-1}U] x^r + \alpha \left( I + \alpha D^{-1}L \right)^{-1} D^{-1}b.$$  

(8)

Hence, the convergence of the SOR algorithm is guaranteed if the spectral norm of the iteration matrix is strictly less than one. Recently, the work [18] shows that when $A$ is symmetric and positive semidefinite (PSD), then the G-S algorithm with RP rule can yield better convergence rate compared with the cyclic G-S (in the asymptotic region where $n$ is large). From the above discussion it is clear that the classical G-S type algorithm does not work for any matrix $A$. A natural question is: Can a G-S type algorithm converge for any matrix $A$?

The convergence of Kaczmarz type algorithm. Another popular method that bears similarity to the G-S type method for iteratively solving (1) is the Kaczmarz method [7], whose iteration is expressed as

$$x^{r+1} = x^r + \frac{b_i - \langle A_i, x^r \rangle}{\|A_i\|^2} A_i^T,$$  

(9)

where $A_i$ is the $i$th row of $A$. This method has been used in many applications, but its rate of convergence was only analyzed in 2008 by Strohmer and Vershynin [20] who proposed a randomized Kaczmarz (RK) method for over-determined linear systems. In the RK method, the $i$-th equation is selected for update randomly with probability proportional to $\|A_i\|^2$. This method can be seen as a particular case of stochastic gradient descent algorithm for minimizing the cost function

$$\frac{1}{2} \frac{(b_i - \langle A_i, x \rangle)^2}{\|A_i\|^2},$$

and the iterates converge linearly to a solution of (1). The RK method has a convergence rate dependent only on a certain scaled condition number of matrix $A$.

In a related work [9], Leventhal and Lewis studied randomized variants of two classical algorithms, one is the CD for solving systems of linear equations (as has been discussed above), the other is the iterated projection [2] for systems of linear inequalities (which contains the RK algorithm [20] as a special case). The authors show that for the first algorithm when $A$ is symmetric (of size $n \times n$) and PSD, and for the second algorithm when the system has nonempty solution set, the global linear convergence can be established. Further the authors show that the linear rate can be bounded in terms of natural linear-algebraic condition numbers of the problems. Note that for the iterated projection method and the RK algorithm, the global linear convergence does not require $A$ to have full column rank.

Other recent works along this line include [4,12,15–17,21]. In [4], a stochastic dual ascent (SDA) algorithm, which contains RK as a special case, was introduced for finding the projection of a given vector onto the solution space of a linear system. The method is dual in nature, with the dual being an unconstrained concave quadratic maximization problem. In each iteration of SDA, a dual variable is updated by choosing a point in a subspace spanned by the columns of a random matrix drawn independently from a fixed distribution. In [12], the authors combined the relaxation method of Motzkin [14] (also known as Kaczmarz method with the
“most violated constraint control”) and the randomized Kaczmarz method \cite{20} to obtain a family of algorithms called Sampling Kaczmarz-Motzkin (SKM) for solving the linear systems $Ax \leq b$. In SKM, at each time a subset of inequalities are picked, and the variables are updated based on the projection to the subspace corresponding to the most violated linear equality/inequality. The reference \cite{21} proposed a new algorithm in which each iteration consists of obtaining $\ell_\infty$ norm projection of current approximate solution (onto hyperplanes defined by individual equations), followed by proper combination of the projections to all equations to yield the next iterate. Different from the Kaczmarz method \cite{7}, this method requires information from all equations to update one variable. Needell \cite{15} extended the RK method to the case of inconsistent equations, and showed that global linear convergence can be achieved until some fixed convergence horizon is reached. Needell and Tropp \cite{16} analyzed a block version of the RK algorithm, in which at each iteration the iterate is projected onto the solution space of many equations simultaneously by selecting a block of rows rather than a single row. The convergence rate of the resulting algorithm is analyzed using the notion of row paving of a matrix. Recently Liu and Wright proposed schemes to accelerate the RK method. The resulting scheme converges faster than the RK algorithm on ill conditioned problems \cite{10}.

Recently, there are a few works analyzing the randomized CD method proposed in \cite{9} and the RK method \cite{20}, see \cite{5,13}. It is shown in \cite{13} that the randomized CD method can be extended to yield the minimum norm solution. In \cite{5}, variants of RK and randomized CD for solving Tikhonov regularized regression is proposed, and the corresponding convergence rates are derived. The rates derived indicate that RK based methods are preferable when $n > m$, while the randomized CD based methods are preferable when $m > n$.

It has been recognized that *randomization* can be effective in simplifying the analysis of RK method. In particular, Leventhal and Lewis \cite{9} have used randomization in the RK method and strengthened the convergence analysis of the resulting randomized algorithm. They concluded that “randomization here provides a framework for simplifying the analysis of algorithms, allowing easy bounds on the rates of linear convergence in terms of natural linear-algebraic condition measures...”. The present paper goes a step further in trying to understand the power of randomization.

\begin{table}[h]
\centering
\begin{tabular}{|l|}
\hline
(Q1) Can randomization make the (otherwise divergent) G-S algorithm convergent for a general linear system? \\
\hline
\end{tabular}
\end{table}

1.2 Contribution of This Work

In this paper, we answer the above question affirmatively. In particular, we propose a *doubly stochastic G-S algorithm* that is provably linearly convergent (in expectation) for any feasible linear system of equations. The key in the algorithm design is to introduce a *nonuniform double randomization* scheme for picking the equation and the variable in each update step of the G-S algorithm, along with an appropriate stepsize rule. Interestingly, these randomization techniques also generalize to certain iterative alternating projection algorithms for solving the linear feasibility problem $Ax \leq b$ with an arbitrary $A$, as well as to certain high-dimensional
convex minimization problems. Our results demonstrate that a carefully designed randomization scheme can make an otherwise divergent G-S algorithm converge linearly.

1.3 Notations

For any matrix $A$, let $\|A^{-1}\|_2$ denote the smallest constant $M$ such that $\|Ax\|_2 \geq \frac{1}{M}\|x\|_2$ for all $x$. Let us define the relative condition number of $A$ as $\kappa(A) := \|A\|_2\|A^{-1}\|_2$; the scaled condition number is defined as $\kappa_s(A) := \|A\|_F\|A^{-1}\|_2$. It is easy to verify that

$$1 \leq \frac{\kappa(A)}{\sqrt{n}} \leq k(A).$$

(10)

We use $A_{i,:}$ and $A_{j,:}$ to denote the $i$th column and $j$th row of $A$, respectively. For a symmetric matrix $B$, we use $\lambda_{\text{max}}(B)$, $\lambda_{\text{min}}(B)$ and $\lambda_{\text{min}}(B)$ to denote its maximum, the minimum and the minimum nonzero eigenvalues.

2 A Doubly Stochastic G-S Algorithm

2.1 Simultaneously selecting equations and variables?

Recall that in the G-S algorithm (3), we always use equation $i$ to update variable $x_i$. In other words, variable $x_i$ is locked to equation $i$. This locking is arbitrary since one can simply re-order the equations (or re-indexing the variables) without affecting the solution of the linear system, and yet different variable to equation coupling will give rise to a different G-S update scheme, some of which may be divergent while others may be convergent. Figure 1 gives an illustrative example in $\mathbb{R}^2$, whereby if we use equation 1 to update variable $x_1$, and equation 2 to update variable $x_2$, then the G-S algorithm diverges (left subfigure), but if we use equation 1 to update variable $x_2$, and equation 2 to update variable $x_1$, then the G-S algorithm converges linearly (right subfigure). This example suggests variable to equation association can greatly affect the convergence of the G-S algorithm.

Thus, a natural way to design a convergent G-S algorithm for a $n \times n$ general linear system (1) is to carefully select a fixed matching that determines which variable is to be updated by which equation. Clearly, the choice of a good matching (one that can lead to a convergent G-S algorithm) will be dependent on the coefficient matrix $A$. Unfortunately, this is a challenging task since the number of possible matchings is $n!$, which grows exponentially in $n$. Moreover, for a non-square linear system ($m \times n$), it is not clear how to define such a matching between variables and equations.

In this paper, we propose to unlock the fixed pairing of each variable to a unique equation in the G-S algorithm. Moreover, since determining which variable is to be updated by which equation is hard, we propose to simply do so randomly! More specifically, at each G-S iteration, we can randomly select a pair $(i, j)$ where $i$ is an index for an equation while $j$ is an index for a variable. Using this strategy,
after picking the pair \((i, j)\), one can update variable \(x_j\) using equation \(i\) as follows (according to the G-S update rule)

\[
x_r^{j+1} = (1 - \alpha)x_r^j + \alpha \frac{b_i - \sum_{k \neq j} a_{ik} x_k^r}{a_{ij}}, \quad x_r^{\ell+1} = x_r^\ell, \quad \forall \ell \neq j.
\]  

(11)

To answer (Q1), it is then natural to ask the following question:

**Q2**

Can unlocking the fixed variable-equation pairing and randomization ensure the convergence of a G-S type algorithm for an arbitrary linear system? 

To understand the impact of unlocking and randomization, let us consider the following example.

**Example 2:** Consider the same \((A, b)\) as given in Example 1, and let us consider the unlocked version of the G-S outlined above.

Specifically, after selecting the pair \((i, j)\), we can use one of the following four update rules to update the variables:

1. **Case 1.** \(i = 1, j = 1 \rightarrow x_1^{j+1} = (1 - \alpha)x_1^j + \alpha \tau x_2^r;
2. **Case 2.** \(i = 1, j = 2 \rightarrow x_2^{j+1} = (1 - \alpha)x_2^j + \frac{\alpha}{2} x_1^j;
3. **Case 3.** \(i = 2, j = 1 \rightarrow x_1^{j+1} = (1 - \alpha)x_1^j + \frac{\alpha}{2} x_2^j;
4. **Case 4.** \(i = 1, j = 2 \rightarrow x_2^{j+1} = (1 - \alpha)x_2^j + \alpha \tau x_1^j.

Consider a uniform randomized update rule where at each iteration, one of above update rules is selected (each with probability 1/4) and used to update the variable \(x\). Consider an initialization that \(x_1, x_2 > 0\). Define the random process \(z^r \triangleq \min\{x_1^r, x_2^r\}\). Using the uniform randomized update rule, one can show that

\[
z^{r+1} = \begin{cases} 
(1 - \alpha)z^r + \alpha \tau z^r & \text{with probability } 1/4 \text{ scenario 1} \\
(1 - \alpha)z^r + \frac{\alpha}{2} z^r & \text{with probability } 1/4 \text{ scenario 2} \\
z^r & \text{with probability } 1/2 \text{ scenario 3}.
\end{cases}
\]

(12)
In order to show the divergence of the algorithm, it suffices to show that $z^r \to \infty$ with probability one. To show this, we define the random process $\{w^r\}_{r=0}^{\infty}$ with $w^0 = z^0$ and

$$
w^{r+1} = \begin{cases} 
(1 - \alpha + \alpha \tau)w^r & \text{if scenario 1 happens in process } z^r \\
(1 - \alpha + \frac{\alpha}{\tau})w^r & \text{if scenario 2 happens in process } z^r \\
w^r & \text{if scenario 3 happens in process } z^r
\end{cases} \quad (13)
$$

Clearly, $z^r \geq w^r$, $\forall r$. Hence we only need to show $w^r \to \infty$ with probability one. Notice that $\log(w^r) = \sum_{i=1}^{r} \beta^i + \log(w^0)$ where $\beta^i$ is an i.i.d process with

$$
\beta^i \geq \begin{cases} 
\log(1 - \alpha + \alpha \tau) & \text{with probability } 1/4 \\
\log(1 - \alpha + \frac{\alpha}{\tau}) & \text{with probability } 1/4 \\
0 & \text{with probability } 1/2
\end{cases} \quad (14)
$$

It is not hard to see that $\mathbb{E}[\beta^i] > 0$, $\forall i$. Therefore, $\lim_{r \to \infty} \sum_{i=1}^{r} \beta^i = \infty$ due to the law of large numbers. Consequently, $\lim_{r \to \infty} \log(w^r) = \infty$ which implies $\lim_{r \to \infty} \|x^r\| = \infty$, regardless of stepsize $0 < \alpha < 1$.

Furthermore, if one uses the cyclic update rule, then at each iteration of the G-S algorithm, each one of the above cases will be selected once according to some deterministic rules. Therefore, in order to study the convergence of the resulting algorithm, one need to look at the spectral radius of the resulting mapping. For example, if we select the update rules in the order of 1), 2), 3), and 4), one needs to study the spectral radius $\rho(B_4B_3B_2B_1)$ where

$$
B_1 = \begin{bmatrix} 1 - \alpha \tau & \alpha \tau \\
0 & 1 \end{bmatrix}, \quad B_2 = \begin{bmatrix} 1 & 0 \\
\alpha / \tau & 1 - \alpha \end{bmatrix}, \quad B_3 = \begin{bmatrix} 1 - \alpha & \alpha / \tau \\
0 & 1 \end{bmatrix}, \quad B_4 = \begin{bmatrix} 1 & 0 \\
\alpha \tau & 1 - \alpha \end{bmatrix}.
$$

One can check that any permutation of the above matrices will result in a product matrix whose spectral radius is larger than 1. Consequently, the resulting algorithm will diverge for almost all initialization, and for any $0 < \alpha < 1$. Furthermore, one can numerically check that the randomly permuted rule will also diverge for almost all initialization and for any $0 < \alpha < 1$.

The above example suggests that by simply unlocking the variable-equation association, the G-S type methods still may not converge. Moreover, Figure 2 shows an example in $\mathbb{R}^2$ whereby the G-S type algorithm will not converge if $\alpha = 1$ regardless of variable-equation association, but will converge if $0 < \alpha < 1$ for any variable-equation association. Thus, stepsize control is also necessary (in addition to randomization) for the convergence of the G-S type algorithm.

Surprisingly, we will show in the subsequent sections that, by properly selecting a data-dependent updating probability as well as the stepsize $\alpha$, the randomized unlocked G-S algorithm (Algorithm 1 below) is globally linearly convergence in the mean squared error sense for any feasible linear system [1].

2.2 A Doubly Stochastic G-S Type Algorithm

We propose a doubly stochastic G-S type algorithm below. This randomized algorithm combines the unlocking idea with certain non-uniform random selection...
Fig. 2: Convergence behavior of the G-S algorithm under different stepsizes of both equations and variables. The key that ensures the convergence of the resulting algorithm, compared with the divergent cases in Example 2, is a judicious selection of the probability $p_{ij}$ that governs how the equations and the variables are picked, as well as a stepsize rule. Note that if a particular entry of the matrix $a_{ij} = 0$, then its corresponding $p_{ij} = 0$, so the $(i,j)$-th index will never be picked. Therefore the update (16) is well defined.

Algorithm 1. A doubly stochastic G-S (DSGS) algorithm.

Let $\alpha > 0$. At iteration 0, randomly generate $x^0$.

At iteration $r + 1$, randomly pick the index pair $(i, j)$ with probability

$$p_{ij} = \frac{a_{ij}^2}{\sum_{i,j} a_{ij}^2}.$$ (15)

Update $x_j$ by the following:

$$x_{j}^{r+1} = (1 - \alpha)x_j + \alpha \left( \frac{b_i - \sum_{k \neq j} a_{ik}x_k}{a_{ij}} \right) = x_j + \alpha \left( \frac{b_i - \sum_{k=1}^n a_{ik}x_k}{a_{ij}} \right).$$ (16)

2.3 Case 1: $A$ has full column rank

We first consider the case where $A$ has full column rank. Let $x^*$ be the feasible solution for $Ax = b$. For simplicity of notation we use the superscript “$+$” to denote the new iteration. Let us define

$$\Delta_j^+ = x_j^+ - x_j^*, \quad \Delta_j = x_j - x_j^*.$$ (17)

We have the following result.

**Theorem 1** Consider a consistent system $Ax = b$ with $A$ being full column rank. Then there holds

$$\mathbb{E} \left[ \| \Delta^+ \|^2 \mid x \right] \leq \Delta^T \left( I + \frac{n\alpha^2 - 2\alpha}{\sum_{i,j} a_{ij}^2 A^T A} \right) \Delta.$$
Thus the DSGS algorithm converges globally linearly in the mean squared error sense for \(0 < \alpha < 2/n\). Moreover, if we choose \(\alpha = 1/n\), then the DSGS algorithm achieves the following convergence rate:

\[
\mathbb{E} \left[ \| \Delta^+ \|^2 \mid x \right] \leq \left(1 - \frac{1}{nk^2(A)}\right) \| \Delta \|^2 \leq \left(1 - \frac{1}{nk^2(A)}\right) \| \Delta \|^2.
\]  

(18)

Proof. Clearly we have the following relation

\[
a_{ij}x_j^* = b_i - \sum_{k \neq j} a_{ik}x_k^*, \quad \forall i.
\]

(19)

Also let us choose \(p_{ij} = \frac{a_{ij}^2}{\sum_i a_{ij}^2}, \quad \forall i, j\). We have the following series of equalities

\[
\mathbb{E}[(\Delta^+)^2 \mid x] = \left(1 - \sum_i p_{ij}\right)(x_j - x_j^*)^2 + \sum_i p_{ij} \left(1 - \alpha\right)x_j + \alpha(b_i - \sum_{k \neq j} a_{ik}x_k - x_j^*)^2
\]

\[
= \left(1 - \frac{1}{\sum_{i,j} a_{ij}^2} \right) \Delta_j^2 + \frac{1}{\sum_{i,j} a_{ij}^2} \sum_i \left(1 - \alpha\right)x_j + \alpha(b_i - \sum_{k \neq j} a_{ik}x_k - \alpha x_j^*)^2
\]

\[
= \left(1 - \frac{1}{\sum_{i,j} a_{ij}^2} \right) \Delta_j^2 + \frac{1}{\sum_{i,j} a_{ij}^2} \sum_i \left(1 - \alpha\right)\Delta_j + \alpha(b_i - \sum_{k \neq j} a_{ik}x_k - \alpha x_j^*)^2
\]

\[
= \left(1 - \frac{1}{\sum_{i,j} a_{ij}^2} \right) \Delta_j^2 + \frac{1}{\sum_{i,j} a_{ij}^2} \sum_i \left(1 - \alpha\right)\sum_k a_{ik} \Delta_k
\]

\[
= \Delta_j^2 + \frac{1}{\sum_{i,j} a_{ij}^2} \left(\sum_i \alpha^2 \sum_k a_{ik} \Delta_k^2 - 2\alpha \Delta_j \sum_i \sum_k a_{ik} \Delta_k \Delta_{ij}\right)
\]

\[
= \Delta_j^2 + \frac{1}{\sum_{i,j} a_{ij}^2} \left(\sum_i \alpha^2 (\sum_k a_{ik} \Delta_k)^2 - 2\alpha \Delta_j \sum_i \sum_k a_{ik} \Delta_k \Delta_{ij}\right)
\]

In the last equation, the notation \(A_i\) denotes the ith row of the matrix \(A\).

Summing the above equation over \(j\), we have

\[
\mathbb{E} \left[ \| \Delta^+ \|^2 \mid x \right] = \| \Delta \|^2 + \alpha^2 \frac{n}{\sum_{i,j} a_{ij}^2} \Delta^T A^T \Delta - \frac{2\alpha}{\sum_{i,j} a_{ij}^2} \sum_j \Delta_j A_{ij}^T A \Delta
\]

\[
= \| \Delta \|^2 + \alpha^2 \frac{n}{\sum_{i,j} a_{ij}^2} \| \Delta \|^2 A^T A - \frac{2\alpha}{\sum_{i,j} a_{ij}^2} \| \Delta \|^2 A^T A
\]

\[
= \Delta^T \left( I + \alpha^2 \frac{n}{\sum_{i,j} a_{ij}^2} A^T A - \frac{2\alpha}{\sum_{i,j} a_{ij}^2} A^T A \right) \Delta
\]

\[
= \Delta^T \left( I + \alpha^2 \frac{n}{\sum_{i,j} a_{ij}^2} - \frac{2\alpha}{\sum_{i,j} a_{ij}^2} A^T A \right) \Delta.
\]
Clearly, to make the error converge geometrically, it suffices to have
\[ \alpha^2 n - 2\alpha < 0, \quad \text{or equivalently} \quad \alpha < \frac{2}{n}. \]

Let us pick \( \alpha = 1/n \), then we have
\[
E[\|\Delta^+\|^2 | x] \leq \|\Delta\|^2 \left( I - \frac{1}{n} \sum_{i,j} a_{ij} \lambda_{\min}(AT A) \right)
\]
\[
= \|\Delta\|^2 \left( I - \frac{1}{n\|A\|_F^2} \|A^{-1}\|^2 \right) = \|\Delta\|^2 \left( I - \frac{1}{nk^2(A)} \right).
\]

This shows that the expected value of the optimality gap shrinks globally geometrically.

Q.E.D.

Remark 1 Let us compare the rate obtained above with existing results in the literature. First, the rate of the randomized CD method obtained in [9, Theorem 3.4] (for solving (1) with \( A \) being symmetric and PD) is given by
\[
E[\|\Delta^+\|^2] \leq \left( 1 - \frac{1}{\|A^{-1}\|_2^2 \text{Tr}[A]} \right) \|\Delta\|^2
\]
\[
\leq \left( 1 - \frac{1}{\sqrt{n\kappa(A)}} \right) \|\Delta\|^2 \leq \left( 1 - \frac{1}{nk^2(A)} \right) \|\Delta\|^2. \quad (20)
\]

We can see that our rate obtained in (18) takes a similar form, except that our rate is proportional to \( 1 - \left( 1 - \frac{1}{\sqrt{n\kappa(A)}} \right)^2 \). This is reasonable due to the lack of symmetry and positive definiteness of \( A \).

Alternatively, the rate obtained in (20) when using RK method for solving (1) with \( A \) being full column rank is given by (see, e.g., [9, Theorem 4.2], [12, Proposition 2])
\[
E[\|\Delta^+\|^2 | x] \leq \left( 1 - \frac{2\alpha - \alpha^2}{\kappa^2(A)} \right) \|\Delta\|^2 \overset{\alpha = 1}{= 1} \left( 1 - \frac{1}{\kappa^2(A)} \right) \|\Delta\|^2. \quad (21)
\]

Note that at each iteration of RK, \( n \) variables are updated. In contrast, our rate in (18) is proportional to \( 1 - \frac{1}{nk^2(A)} \), but at each iteration only one variable is updated. When \( n \) is large and \( \kappa(A) \) is large, we have
\[
\left( 1 - \frac{1}{nk^2(A)} \right)^n \approx \exp(-1/\kappa^2(A)) \approx 1 - \frac{1}{\kappa^2(A)}
\]
which indicates that the two rates are asymptotically comparable. ■
2.4 Case II: \( A \) has no full column rank

In this subsection, we assume that \( A \) has no full column rank, and system (1) has a least one solution \( x^* \). In this case, the previous analysis does not work because (18) does not imply linear convergence. In this section, we use a different analysis technique.

Let us define\[ \beta := Ax - b \in \mathbb{R}^m, \quad \beta_k = A_k x - b_k. \] (22)

Below we will show that the quantity \( \| A(x - x^*) \|_2^2 = \| \beta \|_2^2 \) converges linearly to zero in expectation.

**Theorem 2** Consider a consistent system \( Ax = b \) with arbitrary \( A \). Let us pick\[ \alpha = \frac{1}{\| A \|_F^2} \lambda_{\min}(AA^T). \]

Then the double stochastic G-S algorithm achieves the following convergence rate\[ E[\| \beta^+ \|_2 \mid x] \leq \| \beta \|_2^2 \left( 1 - \left( \frac{1}{\| A \|_F^2} \lambda_{\min}(AA^T) \right)^2 \right)^2. \] (23)

**Proof.** First from the update rule (16), it is clear that when the tuple \((i,j)\) is picked, we have\[ x^+ = x - \alpha \left( \frac{A_i x - b_i}{a_{ij}} \right) e_j \] (24)

where \( e_j \) is the \( j \)th elementary vector. Left multiplying both sides by \( A_k \), we have\[ A_k x^+ = A_k x - \alpha \left( \frac{A_i x - b_i}{a_{ij}} \right) a_{kj}. \] (25)

According to the definition (22), we further have\[ \beta_k^+ = A_k(x^+ - x^*) = (A_k x^+ - b_k) = \beta_k - \alpha \frac{A_i x - b_i}{a_{ij}} a_{kj} = \beta_k - \alpha \frac{\beta_i}{a_{ij}} a_{kj}. \] (26)

Since each tuple \((i,j)\) is picked using probability \( p_{ij} \) in (15), we have the following estimate\[ E[\| \beta_k^+ \|_2^2] = \sum_{i,j} p_{ij} \left( \beta_k - \alpha \frac{\beta_i}{a_{ij}} a_{kj} \right)^2 = \frac{1}{\sum_{i,j} a_{ij}^2} \sum_{i,j} (\beta_k a_{ij} - \alpha \beta_i a_{kj})^2. \] (27)
Summing over $k$, we have
\[
\mathbb{E}[\|\beta^+\|^2] = \frac{1}{\sum_{i,j} a^2_{ij}} \sum_{k,i,j} (\beta_k a_{ij} - \alpha \beta_i a_{kj})^2
\]
\[
= \frac{1}{\sum_{i,j} a^2_{ij}} \sum_{k,i,j} \left( \beta_k^2 a_{ij}^2 - 2 \beta_k a_{ij} \alpha \beta_i a_{kj} + \alpha^2 a_{kij}^2 \beta_i^2 \right)
\]
\[
= \frac{1}{\sum_{i,j} a^2_{ij}} \left( (1 + \alpha^2) \|A\|_F^2 \|\beta\|^2 - 2 \alpha \sum_{k,i,j} (\beta_i a_{ij})(\beta_k a_{kij}) \right)
\]
\[
= \frac{1}{\sum_{i,j} a^2_{ij}} \left( (1 + \alpha^2) \|A\|_F^2 \|\beta\|^2 - 2 \alpha \beta^T A A^T \beta \right)
\]
\[
= \beta^T \left( (1 + \alpha^2) I - \frac{2 \alpha}{\sum_{i,j} a^2_{ij}} A A^T \right) \beta.
\]
Note that by definition $\beta := Ax - b$, and the system is consistent, so $\beta \in \text{col}(AA^T)$. It follows that we have
\[
\|A^T \beta\|^2 \geq \|\beta\|^2 \lambda_{\min}(AA^T).
\]
Therefore, the following sufficient conditions are needed
\[
(1 + \alpha^2) - 2 \alpha \sum_{i,j} a_{ij}^2 \lambda_{\max}(AA^T) > 0
\]
\[
(1 + \alpha^2) - 2 \alpha \sum_{i,j} a_{ij}^2 \lambda_{\min}(AA^T) < 1.
\]
These implies that
\[
\alpha < \frac{2}{\sum_{i,j} a_{ij}^2} \lambda_{\min}(AA^T).
\]
Let us pick
\[
\alpha = \frac{1}{\sum_{i,j} a_{ij}^2} \lambda_{\min}(AA^T)
\]
then we have
\[
\mathbb{E}[\|\beta^+\|^2] \leq \|\beta\|^2 \left( 1 - \left( \frac{1}{\sum_{i,j} a_{ij}^2} \lambda_{\min}(AA^T) \right)^2 \right).
\]
The claim is proved. Q.E.D.

Remark 2 Let us compare the rate obtained above with the rate of a randomized CD method (Algorithm 3.5 in [9]), a method which updates only one variable at each iteration, while utilizing one column of matrix $A$. It is shown that for a consistent linear systems of equations (1) with arbitrary non-zero matrix $A$, the randomized CD method achieves the following rate
\[
\mathbb{E}[\|\beta^+\|^2 | x] \leq \left( 1 - \frac{\lambda_{\min}(A^T A)}{\|A\|_F^2} \right) \|\beta\|^2.
\]
Clearly the above rate is closely related to the one given in [23].
Remark 3 Although we are mainly interested in addressing Question Q1, namely how to design a G-S scheme that converges for any matrix A, the proposed double stochastic algorithm does have important practical value. Consider the distributed computation setting where there is a central controller node 0 connected to a number of distributed computing nodes, each having a subset of rows of data matrix $A$ and $b$, respectively. The central controller stores the variable $x$, the error term $Ax - b$ and is capable of broadcasting to every distributed nodes. At each iteration, the central node randomly pick a pair $(i,j)$, and send the distributed node that has $A_i$ the scalar $A_i x - b_i$; the corresponding node will update $x_j$ according to (16) using its local information. Then the new $x_i$ will be transmitted back to node 0, and node 0 will recompute $Ax - b$ and continue the previous process. Therefore after $n$ iterations of the algorithm, the total number of messages transmitted between the local nodes to node 0 is $2 \times n$. In comparison, if one implements the RK method in the same distributed network, then each iteration $n$ messages have to be communicated from the local nodes to node 0.

3 A Doubly Stochastic Alternating Projection Algorithm

In this section, we extend our previous analysis to the problem of finding a point in the intersection of multiple polyhedral sets. The algorithm to be developed has the flavor of the classical alternating projection algorithm, except that we perform the alternating projection coordinate-wise. Specifically, we consider the following problem:

$$\text{Find } x \quad \text{s.t. } A_i x \leq b_i, \quad i = 1, \cdots, m. \quad (32)$$

We will assume in the rest of this section that the system $Ax \leq b$ is feasible.

The proposed algorithm is closely related to Algorithm 1, except that we only update those inequalities that are violated.

| Algorithm 2. The double stochastic alternating projection algorithm. |
|-----------------------------|
| At iteration 0, randomly generate $x^0$. |
| At iteration $r + 1$, randomly pick the index pair $(i,j)$ with probability $p_{ij} = \frac{a_{ij}^2}{\sum_{i,j} a_{ij}^2}$. |
| Update $x_j$ by the following |
| $x_j^{r+1} = x_j$, if $A_i x \leq b_i$ |
| $x_j^{r+1} = (1 - \alpha)x_j + \alpha \left( \frac{b_i - \sum_{k \neq j} a_{ik} x_k}{a_{ij}} \right)$, otherwise. |

(33)
To facilitate our analysis, let us define the following function

\[ f(x) := \sum_{i=1}^{m} f_i(x) = \frac{1}{2} \sum_{i=1}^{m} (a_i^T x - b_i)^2. \]  

We note that any feasible solution of (32) will imply \( f(x) = 0 \). Further, each function \( f_i \) is differentiable, and its gradient is \( \nabla f_i(x) = A_i^T (A_i x - b_i) \) if \( A_i x - b_i \geq 0 \), and it is 0 otherwise. For a given iteration \( r \), define the index set

\[ I_r := \{ i \mid a_i^T x^r - b_i > 0 \}, \]  

and define \( \Omega_r^I \in \mathbb{R}^{m \times m} \) as the diagonal matrix with \( \Omega^I_{r,j} = 1 \) if \( i \in I_r \) and \( \Omega^I_{r,i} = 0 \) otherwise. Then we have

\[ \sum_{i=1}^{m} f_i(x^r) = \sum_{i \in I_r} f_i(x^r) = \frac{1}{2} \| A^r x^r - b^r \|^2 \]  

where \( A^r = \Omega_r^I A \in \mathbb{R}^{m \times n} \), and \( b^r \) is defined similarly.

Note that by using the above definition, we have

\[ \sum_{i=1}^{m} f_i(x^r) = \frac{1}{2} \| A^r x^r - b^r \|^2 \leq \frac{1}{2} \| A x^r - b \|^2. \]  

3.1 Case 1: \( A \) has full row rank

In this subsection we make the following assumption

\[ \lambda_{\min}(AA^T) > 0. \]  

Similarly as before, we will use \( x^+ \) (resp. \( x \)) to denote the new (resp. previous) iteration; we will use \( A_I, b_I \) and \( \Omega_I \) to denote \( A^r, b^r, \Omega_r^I \) at iteration \( r \), respectively.

**Theorem 3** Suppose \( A \) has full row rank, and \( \alpha \) is chosen as

\[ \alpha < \frac{\lambda_{\min}(AA^T)}{\| A \|_F^2}. \]  

Then we have

\[ \mathbb{E}[f(x^+) \mid x] \leq \left( 1 - \frac{\lambda_{\min}(A^T A)}{2\| A \|_F^2} \right)^2 f(x) \]  

**Proof.** Suppose that the \((i,j)\)th pair gets selected, and that \( j \)th coordinate gets updated (this means that \( i \in I \)), then we can rewrite \( x^+ \) as following

\[ x^+ = x + \alpha \left( \frac{b_i - \sum_{k=1}^{n} a_{ik} x_k}{a_{ij}} \right) e_j. \]
In this case, we can estimate the component function $f_\ell(x^+)$ based on whether the $\ell$th inequality is satisfied for $x$. Suppose that $\ell \in I$ (i.e. the $\ell$th inequality is not satisfied), then we have

$$f_\ell(x^+) = f_\ell\left(x + \alpha \left( \frac{b_i - \sum_{k=1}^n a_{ik} x_k}{a_{ij}} \right) e_j \right)$$

$$\leq f_\ell(x) + \left( \nabla f_\ell(x), \alpha \left( \frac{b_i - \sum_{k=1}^n a_{ik} x_k}{a_{ij}} \right) e_j \right) + \frac{1}{2} \left\| \alpha a_{ij} \frac{b_i - \sum_{k=1}^n a_{ik} x_k}{a_{ij}} \right\|^2$$

$$= f_\ell(x) + \alpha \left( A_\ell : x - b_\ell, \left( \frac{b_i - \sum_{k=1}^n a_{ik} x_k}{a_{ij}} \right) e_j \right) + \frac{1}{2} \left\| \alpha a_{ij} \frac{b_i - \sum_{k=1}^n a_{ik} x_k}{a_{ij}} \right\|^2$$

$$= f_\ell(x) + \alpha \left( a_{ij} \sum_{k=1}^n a_{ik} x_k - b_\ell \right) e_j, \left( \frac{b_i - \sum_{k=1}^n a_{ik} x_k}{a_{ij}} \right) e_j$$

$$+ \frac{\alpha^2}{2} \left\| \frac{b_i - \sum_{k=1}^n a_{ik} x_k}{a_{ij}} \right\|^2. \quad (43)$$

Otherwise, if $\ell \notin I$ (i.e. the $\ell$th inequality is satisfied), we have

$$f_\ell(x^+) = f_\ell\left(x + \alpha \left( \frac{b_i - \sum_{k=1}^n a_{ik} x_k}{a_{ij}} \right) e_j \right)$$

$$= \frac{1}{2} \left( A_\ell : x - b_\ell + \alpha A_\ell \left( \frac{b_i - \sum_{k=1}^n a_{ik} x_k}{a_{ij}} \right) e_j \right)^2$$

$$\leq f_\ell(x) + \frac{1}{2} \left\| \alpha A_\ell \frac{b_i - \sum_{k=1}^n a_{ik} x_k}{a_{ij}} \right\|^2. \quad (44)$$

where the last inequality is due to the fact that for all $\ell \notin I$, $A_\ell : x - b_\ell \leq 0$. Further, if $(i, j)$th pair gets selected, but that $j$th coordinate are not updated, then we have $x^+ = x$. 

Using the above inequalities, we have that the following

\[
\mathbb{E}\left[ \sum_{\ell} f_{\ell}(x^+) \mid x \right] = \mathbb{E}\left[ \sum_{\ell \in \mathcal{I}} f_{\ell}(x^+) \mid x \right] = \sum_{\ell \in \mathcal{I}} \mathbb{E}\left[ f_{\ell}(x^+) \mid x \right]
\]

\[
(\text{i}) \leq \sum_{\ell \in \mathcal{I}} \sum_{i,j \in \mathcal{I}} p_{ij} \left( \alpha \left( a_{\ell j} \left( \sum_{k=1}^{t} a_{\ell k} x_k - b_{\ell} \right) e_j, b_i - \sum_{k=1}^{t} a_{ik} x_k \right) e_j \right) + \sum_{\ell \in \mathcal{I}} \sum_{i,j} p_{ij} f_{\ell}(x)
\]

\[
= \frac{\alpha}{\sum_{i,j} a_{ij}^2} \sum_{\ell \in \mathcal{I}} \sum_{i,j \in \mathcal{I}} \left( a_{\ell j} \left( \sum_{k=1}^{t} a_{\ell k} x_k - b_{\ell} \right) e_j, a_{ij} \left( b_i - \sum_{k=1}^{t} a_{ik} x_k \right) e_j \right)
\]

\[
+ \sum_{\ell \in \mathcal{I}} \sum_{i,j \in \mathcal{I}} \frac{\alpha^2}{2} \left( a_{\ell j} \left( b_i - \sum_{k=1}^{t} a_{ik} x_k \right) e_j \right) + \sum_{\ell \in \mathcal{I}} f_{\ell}(x)
\]

\[
(\text{ii}) \leq \sum_{\ell \in \mathcal{I}} f_{\ell}(x) - \frac{\alpha}{\sum_{i,j} a_{ij}^2} \| A_{\mathcal{I}}^T (A_{\mathcal{I}} x - b_{\mathcal{I}}) \|^2 + \frac{\alpha^2}{2} \| A_{\mathcal{I}} x - b_{\mathcal{I}} \|^2
\]

\[
\leq \left( 1 - \frac{\alpha \lambda_{\min} (A_{\mathcal{I}} A_{\mathcal{I}}^T)}{\| A \|^2_F} + \frac{\alpha^2}{2} \right) \sum_{\ell} f_{\ell}(x) \leq \left( 1 - \frac{\alpha \lambda_{\min} (A A^T)}{\| A \|^2_F} + \frac{\alpha^2}{2} \right) \sum_{\ell} f_{\ell}(x)
\]

where in (i) we have used the two cases \([43]\) and \([14]\); in (ii) we have used the fact that \(f_{\ell}(x) = 0\) for \(\ell \not\in \mathcal{I}\); in the last inequality we have used the the fact that \(A_{\mathcal{I}} A_{\mathcal{I}}^T\) is a principal submatrix of \(A A^T\), the fact that \(f_{\ell}(x) \geq 0\), and the fact that \(\alpha\) is chosen small enough such that

\[
- \frac{\alpha \lambda_{\min} (A A^T)}{\| A \|^2_F} + \frac{\alpha^2}{2} < 0.
\]

This concludes the proof. \(\text{Q.E.D.}\)

3.2 Case 2: \(A\) has no full row rank

In this subsection, we present an analysis of Algorithm 2 without the full row rankness assumption. To this end, we need to use the well-known Hoffman’s error bound.

**Lemma 1** Let \(S\) denote the solution set for the linear system in the constraint (32). Then there exists a constant \(\tau > 0\) independent of \(b\), with the following property

\[
x \in \mathbb{R}^n, S \neq \emptyset \rightarrow \text{dist}(x, S) \leq \tau \| (Ax - b)^+ \|.
\]

(45)

where we have defined

\[
(Ax - b)^+ = \max \{ 0, A_i x - b \}, \quad \text{dist}(x, S) := \inf_{y \in S} \| x - y \|.
\]

(46)
Assume that the system (32) is feasible, and let $S$ denote its solution set and let $x^* \in S$. Clearly we have $f(x^*) := \sum_{i=1}^m f_i(x^*) = 0$. We have the following claim.

**Theorem 4** Consider a feasible system $Ax \leq b$ with arbitrary $A$. Let us pick $\alpha = \frac{1}{n}$. Then Algorithm 2 achieves the following convergence rate

$$
\mathbb{E}[\text{dist}^2(x^+, S) \mid x] \leq \left( 1 - \frac{1}{n^2 \sum_{i,j} a_{ij}^2} \right) \text{dist}^2(x, S).
$$

(47)

**Proof.** Recall that from the update rule we have

$$
x^+ = x + \alpha \left( b_i - \sum_{k=1}^n a_{ik} x_k \right) e_j.
$$

(48)

Let us define the projection of $x$ to the feasible set as

$$
P(x) := \arg\min_{y \in S} \|x - y\|.
$$

(49)

We have the following relationship for $\mathbb{E} [\text{dist}(x^+, S)^2 \mid x]$

$$
\mathbb{E}[\text{dist}^2(x^+, S) \mid x] = \mathbb{E} \left[ \|x^+ - P(x^+)\|^2 \mid x \right] \leq \mathbb{E} \left[ \|x^+ - P(x)\|^2 \mid x \right]
$$

$$
= \mathbb{E} \left[ \|x^+ - x\|^2 \mid x \right] + 2\mathbb{E} \left[ \langle x^+ - x, x - P(x) \rangle \mid x \right] + \|x - P(x)\|^2
$$

where the inequality is due to the definition of the projection (49). Let us bound the above equality term by term. First we have

$$
\mathbb{E} \left[ \|x^+ - x\|^2 \mid x \right] = \sum_{(i,j) : i \in \mathcal{I}} p_{ij} \alpha^2 \left\| \left( b_i - \sum_{k=1}^n a_{ik} x_k \right) e_j \right\|^2
$$

$$
= \sum_{(i,j) : i \in \mathcal{I}} \frac{1}{\sum_{i,j} a_{ij}^2} \alpha^2 \left\| \left( b_i - \sum_{k=1}^n a_{ik} x_k \right) e_j \right\|^2
$$

$$
= \frac{1}{\sum_{i : i \in \mathcal{I}} \sum_{j} a_{ij}^2} \alpha^2 \left\| b_i - \sum_{k=1}^n a_{ik} x_k \right\|^2
$$

$$
= \frac{\alpha^2 n}{\sum_{i,j} a_{ij}^2} \sum_{(i,j) : i \in \mathcal{I}} \|Ax - b\|^2.
$$
The second term in (50) is given by

\[ E \left[ (x^+ - x, x - P(x)) \mid x \right] = 2\alpha \sum_{(i,j) \in I} p_{ij} \left\langle \frac{b_i - \sum_{k=1}^n a_{ik} x_k}{a_{ij}}, e_j, x - P(x) \right\rangle \]

\[ = 2\alpha \sum_{i,j} \frac{1}{a_{ij}} \sum_{(i,j) \in I} \left\langle \left( a_{ij} (b_i - \sum_{k=1}^n a_{ik} x_k) \right), e_j, x - P(x) \right\rangle \]

\[ = 2\alpha \sum_{i,j} \frac{1}{a_{ij}} \left\langle A^T_i (b - A x), x - P(x) \right\rangle \]

\[ = -2\alpha \sum_{i,j} \frac{1}{a_{ij}} \left\langle \nabla f(x), x - P(x) \right\rangle \]

\[ \leq -2\alpha \sum_{i,j} \frac{1}{a_{ij}} \| A_i x - b_i \|^2. \]

where the first equality is due to the fact that \( x - P(x) \) is constant when conditioned on \( x \); the first inequality is due to the convexity of \( f \); and the last equality is because the system is feasible so \( f(P(x)) = 0 \).

Therefore, overall we have

\[ E[\text{dist}^2(x^+, S) \mid x] \leq \frac{n\alpha^2}{\sum_{i,j} a_{ij}^2} \| A_i x - b_i \|^2 + \| x - P(x) \|^2. \]

Therefore if \( 0 < \alpha < 2/n \), we can apply the Hoffman condition (45)

\[ E[\text{dist}^2(x^+, S) \mid x] \leq \frac{n\alpha^2}{\tau^2 \sum_{i,j} a_{ij}^2} \text{dist}^2(x, S) + \| x - P(x) \|^2 \]

\[ = \left( 1 + \frac{n\alpha^2}{\tau^2 \sum_{i,j} a_{ij}^2} \right) \text{dist}^2(x, S) \]

\[ \leq \left( 1 - \frac{1}{n\tau^2 \sum_{i,j} a_{ij}^2} \right) \text{dist}^2(x, S). \] (50)

Therefore we conclude that the algorithm converges linearly in expectation. Q.E.D.

**Remark 4** Let us compare the above result with the rate for a randomized iterative projection method (Algorithm 4.6) developed in [9] (cf. [9, Theorem 4.7]), which is given below

\[ E[\text{dist}^2(x^+, S) \mid s] \leq \left( 1 - \frac{1}{\tau^2 \sum_{i,j} a_{ij}^2} \right) \text{dist}^2(x, S). \] (51)

Note that the randomized iterative projection method updates \( n \) variables at each iteration, therefore we can use an argument similar to Remark 2 to see the two methods have comparable rates.
4 Generalization to Double Stochastic Gradient Descent

In this section, we extend the previous analysis and algorithm design to solve problems beyond linear system of equalities and inequalities.

Consider the following problem

$$\min_x f(x) := \sum_{i=1}^{m} f_i(x_1, \ldots, x_n)$$

(52)

where each $f_i : \mathbb{R}^n \to \mathbb{R}$. We assume the following throughout this section.

**Assumption 1.**

1. Each function $f_i$ is a close and convex function, and that $f_i(x) \geq 0, \ \forall x \in \mathbb{R}^n$.
2. Assume that each $f_i$ has Lipschitz continuous gradient with constant $L_i$, i.e.,

$$\|\nabla f_i(x) - \nabla f_i(z)\| \leq L_i \|x - z\|, \ \forall x, z, \in \mathbb{R}^n, \ i = 1, \cdots, n.$$ 

(53)

Further assume that, for all $x \in \mathbb{R}^n$, $f$ satisfies

$$f(x) - f(P(x)) \leq \langle \nabla f(x), x - P(x) \rangle - \frac{\gamma}{2} \|P(x) - x\|^2,$$ 

(54)

where $P(x)$ is the projection of $x$ to the set of global minimizers for problem (52) similarly as defined in (49); $\gamma > 0$ is some constant.

3. Assume that the global optimal objective value of problem (52) is zero, i.e.,

$$\min_x f(x) = 0.$$

(55)

The assumption in (54) is slightly weaker than the usual strong convexity assumption. For example, the remark below shows that the function defined in (35) satisfies all the conditions in Assumption 1. Additionally, for any optimal solution $x^*$ for problem (52), Assumption 1 implies that

$$\nabla f_i(x^*) = 0, \ \forall i.$$ 

(55)

**Remark 5** The function defined in (35) satisfies Assumption 1. To see this, one can first easily verify that the function defined in (35) satisfies condition (53) with $L_i = \|a_i\|^2$. Moreover, for a given point $x$, define $I = \{i | a_i^T x - b_i \geq 0\}$. Noticing that $A_{Ix} - b_I \geq 0$ and $A_{Ix}P(x) - b_I \leq 0$, we obtain

$$\langle A_{Ix} - b_I, A_{Ix}P(x) - b_I \rangle \leq 0.$$

Adding the term $\|A_{Ix} - b_I\|^2$ to both sides and rearranging the terms, we obtain

$$\|A_{Ix} - b_I\|^2 \leq \langle A_{Ix}^T (A_{Ix} - b_I), x - P(x) \rangle.$$ 

(56)

On the other hand, Hoffman error bound implies that there exists a constant $\gamma$ (independent of $x$) such that

$$\frac{\gamma}{2} \|x - P(x)\|^2 \leq \|A_{Ix} - b_I\|^2.$$ 

(57)

Multiplying (56) by two and adding to (57), we get

$$\|A_{Ix} - b_I\|^2 + \frac{\gamma}{2} \|x - P(x)\|^2 \leq 2\langle A_{Ix}^T (A_{Ix} - b_I), x - P(x) \rangle,$$

i.e., (54) holds.
Remark 6 We also note that the condition \( f(x) - f(P(x)) \leq (\nabla f(x), x - y) \) is weaker than the traditional strong convexity, and it is also weaker than the essentially strong convexity condition defined in [11]. In particular, the essentially strong convexity requires that

\[
f(x) - f(P(x)) \leq \frac{\gamma}{2} ||x - y||^2, \forall x, y, \text{ s.t. } P(x) = P(y). \tag{58}
\]

The above condition clearly implies \( f(x) - f(P(x)) \leq \frac{\gamma}{2} ||x - y||^2, \forall x, y, \text{ s.t. } P(x) = P(y). \]

Under the conditions of Assumption 1, let us consider the following gradient based algorithm algorithm:

| Algorithm 3. A doubly stochastic gradient algorithm. |
|-----------------------------------------------------|
| At iteration 0, randomly generate \( x^0 \).       |
| At iteration \( r + 1 \), pick the index pair \( (i, j) \) with probability \( p_{ij} = p = \frac{1}{mn} \). |
| Update \( x_j \) by the following \( x_j^{r+1} = x_j - \alpha \nabla_j f_i(x) \). |

In contrast to the previous two algorithms, now a uniform sampling probability is used in Algorithm 3. It turns out that for stochastic gradient based algorithm such a choice is sufficient to guarantee convergence.

Theorem 5 Suppose Assumption A is satisfied. Then applying Algorithm 3 to problem \( (52) \) with a stepsize

\[
0 < \alpha \leq \frac{\gamma^2}{2 \sum_{\ell} L_\ell \sum_{i=1}^N L_i^2}
\]

achieves the following convergence rate

\[
\mathbb{E}[f(x^+) \mid x] \leq \left(1 - \frac{\alpha \gamma}{mn}\right) f(x). \tag{60}
\]

Proof. Suppose that \((i, j)\)th pair gets selected, then we have

\[
x^+ = x - \alpha \nabla f_i(x) e_j. \tag{61}
\]

For a fixed \( \ell \in [m] \), and by using \( (53) \) we have the following

\[
f_\ell(x^+) = f_\ell(x - \alpha \nabla f_i(x) e_j) \\
\leq f_\ell(x) - (\nabla f_i(x), \alpha \nabla f_i(x) e_j) + \frac{L_\ell \alpha^2}{2} ||\nabla f_i(x) e_j||^2.
\]
Therefore we obtain the following relations:

\[ E[f(x^+)|x] \leq \frac{1}{mn} \sum_{\ell \in [m]} \sum_{i,j} \left( -\alpha \langle \nabla f_\ell(x)e_j, \nabla f_\ell(x)e_j \rangle + \frac{L_\ell \alpha^2}{2} \| \nabla f_\ell(x) \|_2^2 \right) + f(x) \]

\[ = \frac{1}{mn} \sum_{\ell \in [m]} \sum_{i,j} \left( -\alpha \langle \nabla f_\ell(x)e_j, \nabla f_\ell(x)e_j \rangle + \sum_{i=1}^N \frac{L_i \alpha^2}{2mn} \| \nabla f_i(x) \|_2^2 \right) + f(x) \]

\[ = f(x) - \frac{\alpha}{mn} \| \sum_{i=1}^m \nabla f_i(x) \|^2 + \sum_{i=1}^N \frac{L_i \alpha^2}{2mn} \| \nabla f_i(x) \|^2 \]

\[ \overset{(55)}{\leq} f(x) - \frac{\alpha}{mn} \| \nabla f(x) \|^2 + \sum_{i=1}^N \frac{L_i \alpha^2}{2mn} \| \nabla f_i(x) - \nabla f_i(x^*) \|^2 \]

\[ \overset{(62)}{\leq} f(x) - \frac{\alpha}{mn} \| \nabla f(x) \|^2 + \sum_{i=1}^N \frac{L_i \alpha^2}{2mn} \sum_{i=1}^N L_i^2 \| x - x^* \|^2. \]

Using the property (54), we have

\[ f(x) - f(x^*) \leq \langle \nabla f(x), x - x^* \rangle - \frac{\gamma}{2} \| x - x^* \|^2 \]

\[ \leq \frac{1}{\gamma} \| \nabla f(x) \|^2 + \frac{\gamma}{4} \| x - x^* \|^2 - \frac{\gamma}{2} \| x - x^* \|^2 \]

\[ = \frac{1}{\gamma} \| \nabla f(x) \|^2 - \frac{\gamma}{4} \| x - x^* \|^2. \]  

(63)

Combine the above with the assumption that \( f(x^*) = 0 \), we obtain

\[ -\frac{\alpha}{mn} \| \nabla f(x) \|^2 \leq -\frac{\alpha \gamma}{mn} f(x) - \frac{\gamma^2}{4} \frac{\alpha}{mn} \| x - x^* \|^2. \]  

(64)

Plugging this inequality into (62), we obtain

\[ E[f(x^+)|x] \leq \left( 1 - \frac{\alpha \gamma}{mn} \right) f(x) - \left( \frac{\gamma^2}{4} \frac{\alpha}{mn} - \sum_{\ell} \frac{L_\ell \alpha^2}{2mn} \sum_{i=1}^N L_i^2 \right) \| x - x^* \|^2. \]  

(65)

Therefore by choosing the following constant

\[ \alpha = \frac{\gamma^2}{2 \sum_{\ell} L_\ell \sum_{i=1}^N L_i^2} \]  

(66)

we obtain the desired result.

Q.E.D.

It should be noted that the stochastic gradient descent (SGD) algorithm is known to converge sublinearly. For problems satisfying Assumption 1 (especially the requirement that \( f_i(x^*) = f(x^*) = 0 \) at each optimal solution \( x^* \)), our analysis shows that SGD can indeed converge linearly with a constant stepsize.
Table 1: The Average Number of Required Iterations to Reach An Error of $10^{-10}$ For An Overdetermined System.

| Size $(m \times n)$ | DSGS | RK    | RGS    | REK    | S2CD   |
|---------------------|------|-------|--------|--------|--------|
| 40 $\times$ 20     | 3,612 | 4,334 | 4,342  | 4,949  | 8,671  |
| 40 $\times$ 30     | 17,211 | 29,653 | 30,174 | 34,067 | 59,892 |
| 60 $\times$ 30     | 7,715  | 7,205  | 7,230  | 8,348  | 14,392 |
| 60 $\times$ 50     | 36,503 | 41,638 | 41,829 | 47,545 | 81,542 |
| 80 $\times$ 40     | 13,262 | 9,026  | 9,058  | 10,591 | 18,195 |
| 80 $\times$ 60     | 83,857 | 84,974 | 84,871 | 95,807 | 166,224 |

5 Numerical Results

In this section, we evaluate the performance of the doubly stochastic G-S algorithm proposed in this paper against some existing algorithms, for solving linear systems \(Ax = b\). In particular, the performance of DSGS is compared with the S2CD from [8], the RK from [20], the RGS from [13], and the REK from [22]. Note that except the S2CD algorithm, the rest of the algorithms are designed specifically for solving linear systems. The S2CD is an extension of the SVRG algorithm [6] to include coordinate descent update, and it solves a strongly convex finite sum problem using a doubly stochastic update. For fair comparison, we increase the iteration counters of these algorithms when \(n\) updates of the variable coordinates are completed. We divide our experiments into two broad categories: the matrix \(A^T A\) is PD and \(A^T A\) is PSD. The elements of the matrix \(A\) is chosen randomly from a standard Gaussian distribution.

In the first set of experiments, we consider over-determined systems with \(m \geq n\). To implement S2CD, we consider an objective function \(f(x) = \frac{1}{m} \sum_{i=1}^{m} \|A_i x - b_i\|^2\) with \(f_i(x) = \|A_i x - b_i\|^2\). We terminate an algorithm when it reaches an error \(\|x - x^*\| \leq 10^{-10}\). Each entry in the table is computed by averaging the results for running a given algorithm over twenty randomly generated problems. It is observed from the Table 1 that when the matrix \(A\) gets close to a square matrix, the number of required iterations increases. This is reasonable since the condition numbers are increased. Moreover, for these harder cases, 40 $\times$ 30, 60 $\times$ 50 and 80 $\times$ 60, the proposed approach in this paper slightly outperforms other algorithms.

In the second set of experiments, we consider under-determined systems with \(m \leq n\). Here we do not include S2CD because it only works for strongly convex problems. The termination rule in this case is chosen as \(\|Ax - b\| \leq 10^{-10}\). It is observed that for all the tested settings, the proposed DSGS algorithm outperforms other algorithms.

Further we plot the least squares error versus the number of iterations in Fig. 3, where convergence rates of algorithms S2CD, DSGS, RK, RGS and REK are depicted for a random Gaussian matrix of size 40 $\times$ 30. It is observed that for the considered linear system with random matrix, DSGS slightly outperforms other algorithms and has a higher convergence rate. Since DSGS chooses each row and coordinate to update randomly, in some iterations, the residual of least squares in DSGS algorithm does not decrease monotonically. Next, we consider several realizations for the matrix \(A\) and solve the linear system with different approaches. In Figs. 4 & 5, ten different realizations are considered for the matrix \(A\). The linear
Table 2: The Average Number of Required Iterations to Reach An Error of $10^{-10}$ For An Underdetermined System.

| Size $(m \times n)$ | DSGS  | RK    | RGS   | REK   |
|---------------------|-------|-------|-------|-------|
| 40 $\times$ 60     | 11,354| 23,257| 23,097| 26,506|
| 40 $\times$ 100     | 3,141 | 6,297 | 6,267 | 6,971 |
| 80 $\times$ 100     | 65,030| 124,724| 128,957| 144,894|
| 80 $\times$ 140     | 15,171| 30,664| 30,168| 34,608|
| 160 $\times$ 200    | 151,858| 274,537| 281,331| 328,680|
| 160 $\times$ 300    | 25,692| 49,791| 49,908| 56,438|

system in Fig. 4 is of size $40 \times 30$ and it is seen that DSGS and RK have close convergence rates and both higher than the rate of S2CD. We increase the size of the matrix $A$ to $60 \times 50$ and similarly consider ten realizations. It is seen that the number of required iterations is increased.

In addition, we graphically demonstrate the convergence behavior of different algorithms. In Fig. 3, we compare one iteration of different algorithms for an underdetermined system of size $40 \times 100$. Similar to the results given in Table 2, it is seen that DSGD requires smaller number of iterations to converge. We also plot a number of different cases with 10 realizations for each of the algorithms in Figs. 4 – 6. In all these cases we observe that the means of the iterates generated by the proposed DSGD decrease quickly, while each realization of the algorithm can be noisy, due to the randomness in choosing both the coordinates and the equations. Also we observe that the S2CD is only designed for strongly convex problems, therefore it does not perform well for the under-determined system whose the objective function is only convex but not strongly convex.

![Fig. 3: Convergence behavior of different algorithms for an underdetermined system of size $40 \times 100$.](image-url)
Fig. 4: Convergence behavior for DSGS, RK and RGS algorithms for a linear system of size $40 \times 30$. In the figure 10 realizations are shown for each of the algorithms.

Fig. 5: Convergence rate for DSGS, RK and RGS algorithms for a linear system of size $60 \times 50$. In the figure 10 realizations are shown for each of the algorithms.

References

1. Bertsekas, D.P., Tsitsiklis, J.N.: Parallel and Distributed Computation: Numerical Methods, 2nd ed. Athena Scientific, Belmont, MA (1997)
2. Deutsch, F.: Best Approximation in Inner Product Spaces. Springer-Verlag, New York (2001)
3. Golub, G., van Loan, C.: Matrix Computation. Johns Hopkins University Press (1996)
4. Gower, R.M., Richtarik, P.: Stochastic dual ascent for solving linear systems (2015). ArXiv Preprint: arXiv:1512.06890
5. Hefny, A., Needell, D., Ramdas, A.: Rows versus columns: Randomized Kaczmarz or Gauss–Seidel for ridge regression. SIAM Journal on Scientific Computing 39(5), S528–S542 (2017)
6. Johnson, R., Zhang, T.: Accelerating stochastic gradient descent using predictive variance reduction. In: the Proceedings of the Neural Information Processing (NIPS) (2013)
7. Kaczmarz, S.: Approximate solution of systems of linear equations. International Journal of Control 57(6), 1269–1271 (1993). Translated from German original of 1933
8. Konecny, J., Qu, Z., Richtarik, P.: Semi-stochastic coordinate descent (2014). Preprint, available at arXiv:1412.6293
9. Leventhal, D., Lewis, A.S.: Randomized methods for linear constraints: Convergence rates and conditioning. Mathematics of Operations Research 35(3), 641–654 (2010)
10. Liu, J., J. Wright, S.: An accelerated randomized kaczmarz algorithm. Mathematics of Computation 85, 153–178 (2016)
11. Liu, J., S. J. Wright, C.R., Bittorf, V.: An asynchronous parallel stochastic coordinate descent algorithm. In: the Proceedings of the International Conference on Machine Learning (ICML) (2014)
Fig. 6: Convergence rate for DSGS, RK and RGS algorithms for a linear system with a matrix of size $40 \times 100$ and $80 \times 140$.

12. Loera, J.A.D., Haddock, J., Needell, D.: A sampling Kaczmarz–Motzkin algorithm for linear feasibility. SIAM Journal on Scientific Computing 39(5), S66–S87 (2017)
13. Ma, A., d. Needell, Ramdas, A.: Convergence properties of the randomized extended Gauss–Seidel and Kaczmarz methods. SIAM Journal on Matrix Analysis and Applications 36(4), 1590–1604 (2015)
14. Motzkin, T.S., Schoenberg., I.J.: The relaxation method for linear inequalities. Canadian J. Math. 6, 393–404 (1954)
15. Needell, D.: Randomized Kaczmarz solver for noisy linear systems. BIT Numerical Mathematics 50(2), 395–403 (2010)
16. Needell, D., Tropp, J.A.: Paved with good intentions: Analysis of a randomized block Kaczmarz method. Linear Algebra Appl. p. 199221 (2013)
17. Oswald, P., Zhou, W.: Convergence analysis for Kaczmarz-type methods in a Hilbert space framework. Linear Algebra Appl. 478, 1311–161 (2015)
18. Oswald, P., Zhou, W.: Random reordering in SOR-type methods. Numerische Mathematik 135(4), 1207–1220 (2017)
19. Saad, Y.: Iterative Methods for Sparse Linear Systems. Society for Industrial and Applied Mathematics (2003)
20. Strohmer, T., Vershynin, R.: A randomized Kaczmarz algorithm with exponential convergence. Journal of Fourier Analysis and Applications 15(2), 262 (2008)
21. Wheaton, I., Awoniyi, S.: A new iterative method for solving non-square systems of linear equations. Journal of Computational and Applied Mathematics 322(Supplement C), 1 – 6 (2017)
22. Zouzias, A., Freris, N.M.: Randomized extended Kaczmarz for solving least squares. SIAM Journal on Matrix Analysis and Applications 34(2), 773–793 (2013)