Mathematical Programming (2019) 176:465–496
https://doi.org/10.1007/s10107-019-01404-0

FULL LENGTH PAPER

Series B

A linearly convergent doubly stochastic Gauss–Seidel algorithm for solving linear equations and a certain class of over-parameterized optimization problems

Meisam Razaviyayn[1] · Mingyi Hong[2] · Navid Reyhanian[2] · Zhi-Quan Luo[3]

Received: 1 May 2018 / Accepted: 15 May 2019 / Published online: 22 May 2019

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 high-dimensional minimization problems for training over-parameterized models in machine learning. Our results demonstrate that a carefully designed randomization scheme can make an otherwise divergent G–S algorithm converge.

Keywords Gauss–Seidel algorithm · Linear systems of equations · Nonuniform block coordinate descent algorithm · Over-parameterized optimization

Mathematics Subject Classification 49xx Calculus of variations and optimal control; optimization

Meisam Razaviyayn and Mingyi Hong have contributed equally to this work.

This research is supported by the NSFC Grants 61731018 and 61571384, the Peacock project of SRIBD.

Extended author information available on the last page of the article

1 Introduction: solving linear system of equations

Consider the generic problem of solving a linear system of equations

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

where $A \in \mathbb{R}^{m\times n}$, $x,\ b \in \mathbb{R}^n$. We assume this system of equations has at least one solution. 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 [19]. More precisely, let $A_{j:}$ and b_j denote the jth row of A and jth 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 ith equation $\sum_{j=1}^n a_{ij}x_j = b_i$. The best response strategy for player i is given by

$$\widehat{x_i} = \frac{b_i - \sum_{j\neq i} a_{ij}x_j}{a_{ii}}. \tag{2}$$

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

$$x_i^{r+1} = (1-\alpha)x_i^r + \alpha\frac{b_i - \sum_{j\neq i} a_{ij}x_j^r}{a_{ii}}, \tag{3}$$

where a_{ij} is the (i, j)th element of A; b_i is the ith element of b. The central question is how to choose the step-size α 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 2×2 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:

$$x_1^{r+1} = (1-\alpha)x_1^r + \alpha\tau x_2^r, \quad \text{when } x_1 \text{ is updated before } x^2 \tag{4}$$

$$x_2^{r+1} = (1-\alpha)x_2^r + \alpha\tau x_1^r, \quad \text{when } x_2 \text{ is updated before } x^1. \tag{5}$$

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

$$x_1^{r+1} = (1-\alpha)x_1^r + \alpha\tau x_2^r, \quad x_2^{r+1} = (1-\alpha)x_2^r + \alpha\tau x_1^{r+1}. \tag{6}$$

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 [22]:

$$x_1^{r+1/2} = (1-\alpha)x_1^r + \alpha\tau x_2^r, \quad x_2^{r+1/2} = (1-\alpha)x_2^r + \alpha\tau x_1^{r+1/2}.$$
$$x_2^{r+1} = (1-\alpha)x_2^{r+1/2} + \alpha\tau x_1^{r+1/2}, \quad x_1^{r+1} = (1-\alpha)x_1^{r+1/2} + \alpha\tau x_r^{r+1}.$$

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 π of the index set $\{1, 2\}$; The variables are updated according to

$$x_{\pi(1)}^{r+1} = (1-\alpha)x_{\pi(1)}^r + \alpha\tau x_{\pi(2)}^r, \quad x_{\pi(2)}^{r+1} = (1-\alpha)x_{\pi(2)}^r + \alpha\tau x_{\pi(1)}^{r+1}. \tag{7}$$

   Note that this method is referred to as the *shuffled* SOR in [21].

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); see [19, Chapter 7] and [1, Propositions 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

$$x^{r+1} = (1 + \alpha D^{-1}L)^{-1}\left[(1-\alpha)I - \alpha D^{-1}U\right]x^r + \alpha\left(I + \alpha D^{-1}L\right)^{-1} D^{-1}b. \tag{8}$$

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 [21] 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

$$x^{r+1} = x^r + \frac{b_i - \langle A_{i:}, x^r\rangle}{\|A_{i:}\|^2} A_{i:}^T \tag{9}$$

where $A_{i:}$ is the ith 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 [23] who proposed a randomized Kaczmarz (RK) method for over-determined linear systems. In the RK method, the ith 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 [10], 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 [23] 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 [4,13,17,18,20,24]. 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 a 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 [13], the authors combined the relaxation method of Motzkin [15] (also known as Kaczmarz method with the "most violated constraint control") and the randomized Kaczmarz method [23] 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 [24] proposed a new algorithm in which each iteration consists of obtaining ℓ_∞ 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 [7], this method requires information from all equations to update one variable. Needell [17] 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 [18] 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 [11].

Recently, there are a few works analyzing the randomized CD method proposed in [10] and the RK method [23], see [5,14]. It is shown in [14] that the randomized CD method can be extended to yield the minimum norm solution. In [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 [10] have used randomization in the RK method and strengthened the convergence analysis of the resulting random-

ized 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.

(Q1) Can randomization make the (otherwise divergent) G–S algorithm convergent for a general linear system (1)?

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 over-parameterized minimization problems. Our results demonstrate that a carefully designed randomization scheme can make an otherwise divergent G–S algorithm converge linearly.

1.3 Notation

For any matrix A, let $\|A^\dagger\|_2$ 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 $k(A) := \|A\|_2\|A^\dagger\|_2$; the *scaled condition number* is defined as $\kappa(A) := \|A\|_F\|A^\dagger\|_2$. It is easy to verify that

$$1 \leq \frac{\kappa(A)}{\sqrt{n}} \leq k(A). \tag{10}$$

We use $A_{:i}$ and $A_{j:}$ to denote the ith column and jth row of A, respectively. For a symmetric matrix B, we use $\lambda_{\max}(B)$, $\lambda_{\min}(B)$ and $\underline{\lambda}_{\min}(B)$ 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

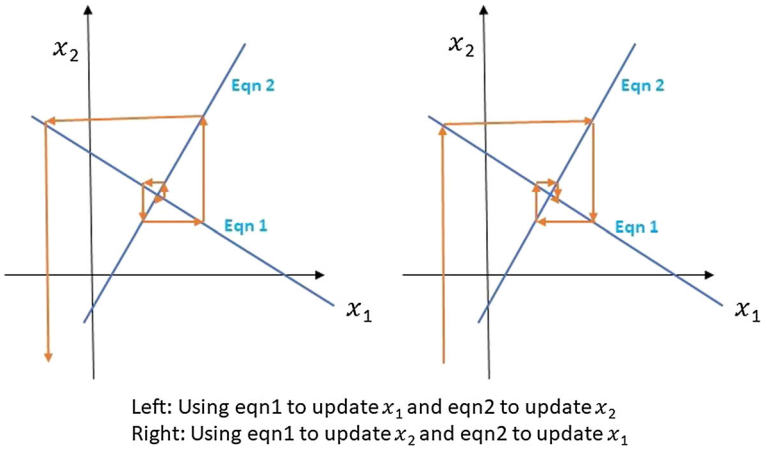

Fig. 1 Convergence behavior of the G–S algorithm under two different variable to equation associations

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)

$$x_j^{r+1} = (1-\alpha)x_j^r + \alpha \frac{b_i - \sum_{k \neq j} a_{ik} x_k^r}{a_{ij}}, \quad x_\ell^{r+1} = x_\ell^r, \ \forall\, \ell \neq j. \tag{11}$$

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 (1)?

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

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 and x_2 have the same sign. Without loss of generality let us assume $x_1, x_2 > 0$. Define the random process $z^r \triangleq \min\{x_1^r, x_2^r\}$. Using the uniform randomized update rule, one can show that

$$z^{r+1} \geq \begin{cases} (1-\alpha)z^r + \alpha\tau z^r & \text{with probability 1/4 scenario 1} \\ (1-\alpha)z^r + \frac{\alpha}{\tau}z^r & \text{with probability 1/4 scenario 2} \\ z^r & \text{with probability 1/2 scenario 3} \end{cases}. \tag{12}$$

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

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

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

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

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

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

$$B_4 = \begin{bmatrix} 1 & 0 \\ \alpha\tau & 1-\alpha \end{bmatrix}.$$

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, Fig. 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 α, the randomized unlocked G–S algorithm (Algorithm 1 below) is globally linearly convergent in the mean squared error sense for any feasible linear system (1). In this algorithm, at each iteration only one index pair (i, j) is selected. However, the selection is based on a non-uniform distribution defined over index pairs.

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 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

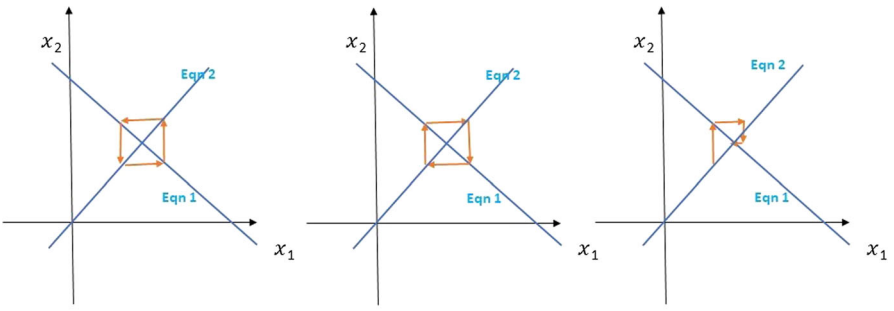

Fig. 2 Convergence behavior of the G–S algorithm under different stepsizes

corresponding $p_{ij} = 0$. Hence the corresponding (i, j)th index will never be picked with probability one. 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 select one index pair (i, j) with the probability

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

Update x_j by the following:

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

and for all $k, k \neq j$, set

$$x_k^{r+1} = x_k^r. \tag{17}$$

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 (1). For simplicity of notation we use the superscript "$+$" to denote the new iteration. Let us define

$$\Delta_j^+ = x_j^+ - x_j^*, \quad \Delta_j = x_j - x_j^*. \tag{18}$$

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:

$$\mathbb{E}\left[\|\Delta^+\|^2 \mid x\right] \leq \left(1 - \frac{1}{n\kappa^2(A)}\right)\|\Delta\|^2 \leq \left(1 - \frac{1}{n^2k^2(A)}\right)\|\Delta\|^2. \tag{19}$$

Proof Clearly we have the following relation

$$a_{ij}x_j^* = b_i - \sum_{k\neq j} a_{ik}x_k^*, \quad \forall\, i. \tag{20}$$

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

$$
\begin{aligned}
&\mathbb{E}[(\Delta_j^+)^2 \mid x] \\
&= \left(1 - \sum_i p_{ij}\right)(x_j - x_j^*)^2 + \sum_i p_{ij}\left((1-\alpha)x_j + \alpha\frac{b_i - \sum_{k\neq j} a_{ik}x_k}{a_{ij}} - x_j^*\right)^2 \\
&= \left(1 - \frac{\sum_i a_{ij}^2}{\sum_{i,j} a_{ij}^2}\right)\Delta_j^2 + \frac{1}{\sum_{i,j} a_{ij}^2}\sum_i \left(a_{ij}(1-\alpha)x_j + \alpha\left(b_i - \sum_{k\neq j} a_{ik}x_k\right)\right. \\
&\quad \left. -(1-\alpha)x_j^* a_{ij} - \alpha x_j^* a_{ij}\right)^2 \\
&\overset{(20)}{=} \left(1 - \frac{\sum_i a_{ij}^2}{\sum_{i,j} a_{ij}^2}\right)\Delta_j^2 + \frac{1}{\sum_{i,j} a_{ij}^2}\sum_i \left(a_{ij}(1-\alpha)\Delta_j + \alpha\left(b_i - \sum_{k\neq j} a_{ik}x_k\right)\right. \\
&\quad \left. -\alpha\left(b_i - \sum_{k\neq j} a_{ik}x_k^*\right)\right)^2 \\
&= \left(1 - \frac{\sum_i a_{ij}^2}{\sum_{i,j} a_{ij}^2}\right)\Delta_j^2 + \frac{1}{\sum_{i,j} a_{ij}^2}\sum_i \left(a_{ij}\Delta_j - \alpha\sum_k a_{ik}\Delta_k\right)^2 \\
&= \left(1 - \frac{\sum_i a_{ij}^2}{\sum_{i,j} a_{ij}^2}\right)\Delta_j^2 + \frac{1}{\sum_{i,j} a_{ij}^2}\sum_i \left(a_{ij}^2\Delta_j^2 + \alpha^2\left(\sum_k a_{ik}\Delta_k\right)^2\right. \\
&\quad \left. -2\alpha\sum_k a_{ik}\Delta_k a_{ij}\Delta_j\right) \\
&= \Delta_j^2 + \frac{1}{\sum_{i,j} a_{ij}^2}\left(\sum_i \alpha^2\left(\sum_k a_{ik}\Delta_k\right)^2 - 2\alpha\Delta_j\sum_i\sum_k a_{ik}\Delta_k a_{ij}\right) \\
&= \Delta_j^2 + \frac{1}{\sum_{i,j} a_{ij}^2}\left(\alpha^2\sum_i (A_{i:}\Delta)^2 - 2\alpha\Delta_j A_{:j}^T A\Delta\right).
\end{aligned}
$$

In the last equation, the notation $A_{i:}$ denotes the ith row of the matrix A.

Summing the above equation over j, we have

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

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

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

$$\mathbb{E}[\|\Delta^+\|^2 \mid x] \le \|\Delta\|^2 \left(1 - \frac{1}{n \sum_{i,j} a_{ij}^2} \lambda_{\min}(A^T A) \right)$$
$$= \|\Delta\|^2 \left(1 - \frac{1}{n \|A\|_F^2 \|A^\dagger\|_2^2} \right) = \|\Delta\|^2 \left(1 - \frac{1}{n\kappa^2(A)} \right).$$

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

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 [10, Theorem 3.4] (for solving (1) with A being symmetric and PD) is given by

$$\left(1 - \frac{1}{\|A^\dagger\|_2 \mathrm{Tr}[A]} \right) \le \left(1 - \frac{1}{\sqrt{n}\kappa(A)} \right) \le \left(1 - \frac{1}{nk(A)} \right), \tag{21}$$

where the last inequality is due to (10). We can see that our rate obtained in (19) takes a similar form, except that our rate is proportional to $\left(1 - \left(\frac{1}{\sqrt{n}\kappa(A)} \right)^2 \right)$. This is due to the lack of symmetry and positive definiteness of A.

Alternatively, the rate obtained in [23] when using RK method for solving (1) with A being full column rank is given by (see, e.g., [10, Theorem 4.2], [13, Proposition 2])

$$\mathbb{E}[\|\Delta^+\|^2 \mid x] \le \left(1 - \frac{2\alpha - \alpha^2}{\kappa^2(A)} \right) \|\Delta\|^2 \overset{\alpha=1}{=} \left(1 - \frac{1}{\kappa^2(A)} \right) \|\Delta\|^2. \tag{22}$$

Note that at each iteration of RK, n variables are updated. In contrast, our rate in (19) is proportional to $\left(1 - \frac{1}{n\kappa^2(A)} \right)$, but at each iteration only one variable is updated. When n is large and $\kappa(A)$ is large, we have

$$\left(1-\frac{1}{n\kappa^2(A)}\right)^n \approx \exp(-1/\kappa^2(A)) \approx 1-\frac{1}{\kappa^2(A)}$$

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 (19) does not imply linear convergence as $\kappa(A) = \infty$. In this section, we use a different analysis technique.

Let us define

$$\beta := Ax - b \in \mathbb{R}^m, \quad \text{with} \quad \beta_k = A_{k:}x - b_k. \tag{23}$$

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

Theorem 2 *Consider a consistent system $Ax = b$ with arbitrary A. Let us pick*

$$\alpha = \frac{1}{\|A\|_F^2}\underline{\lambda}_{\min}(AA^T),$$

and define $\beta^+ := Ax^+ - b$ to be the residual after one update of the algorithm. Then the double stochastic G–S algorithm achieves the following convergence rate

$$\mathbb{E}[\|\beta^+\|^2 \mid x] \le \|\beta\|^2\left(1-\left(\frac{1}{\|A\|_F^2}\underline{\lambda}_{\min}(AA^T)\right)^2\right). \tag{24}$$

Proof First from the update rule (16), it is clear that when the tuple (i, j) is picked, we have

$$x^+ = x - \alpha\left(\frac{A_{i:}x - b_i}{a_{ij}}\right)e_j \tag{25}$$

where e_j is the jth elementary vector. Left multiplying both sides by $A_{k:}$, we have

$$A_{k:}x^+ = A_{k:}x - \alpha\left(\frac{A_{i:}x - b_i}{a_{ij}}\right)a_{kj}. \tag{26}$$

According to the definition (23), we further have

$$\begin{aligned}\beta_k^+ &= A_{k:}(x^+ - x^*) = (A_{k:}x^+ - b_k)\\ &= \beta_k - \alpha\frac{A_{i:}x - b_i}{a_{ij}}a_{kj} = \beta_k - \alpha\frac{\beta_i}{a_{ij}}a_{kj}.\end{aligned} \tag{27}$$

Since each tuple (i, j) is picked using probability p_{ij} in (15), we have the following estimate

$$\mathbb{E}[\|\beta_k^+\|^2 \mid x] = \sum_{i,j} p_{ij} \left(\beta_k - \alpha \frac{\beta_i}{a_{ij}} a_{kj}\right)^2 = \frac{1}{\sum_{i,j} a_{ij}^2} \sum_{i,j} \left(\beta_k a_{ij} - \alpha \beta_i a_{kj}\right)^2. \quad (28)$$

Summing over k, we have

$$\begin{aligned}
\mathbb{E}[\|\beta^+\|^2 \mid x] &= \frac{1}{\sum_{i,j} a_{ij}^2} \sum_{k,i,j} \left(\beta_k a_{ij} - \alpha \beta_i a_{kj}\right)^2 \\
&= \frac{1}{\sum_{i,j} a_{ij}^2} \sum_{k,i,j} \left(\beta_k^2 a_{ij}^2 - 2\beta_k a_{ij} \alpha \beta_i a_{kj} + \alpha^2 a_{kj}^2 \beta_i^2\right) \\
&= \frac{1}{\sum_{i,j} a_{ij}^2} \left((1+\alpha^2)\|A\|_F^2 \|\beta\|^2 - 2\alpha \sum_{k,i,j} (\beta_i a_{ij})(\beta_k a_{kj})\right) \\
&= \frac{1}{\sum_{i,j} a_{ij}^2} \left((1+\alpha^2)\|A\|_F^2 \|\beta\|^2 - 2\alpha \beta^T A A^T \beta\right) \\
&= \beta^T \left((1+\alpha^2) I - \frac{2\alpha}{\sum_{i,j} a_{ij}^2} A A^T\right) \beta.
\end{aligned}$$

Note that by definition $\beta := Ax - b$, and the system is consistent. Therefore β belongs to the column space of AA^T. It follows that we have

$$\|A^T \beta\|^2 \geq \|\beta\|^2 \underline{\lambda}_{\min}(AA^T).$$

Therefore, the following sufficient conditions are needed

$$\begin{aligned}
(1+\alpha^2) - 2\frac{\alpha}{\sum_{i,j} a_{ij}^2} \lambda_{\max}(AA^T) > 0 \\
(1+\alpha^2) - 2\frac{\alpha}{\sum_{i,j} a_{ij}^2} \underline{\lambda}_{\min}(AA^T) < 1.
\end{aligned} \quad (29)$$

By picking the value

$$\alpha = \frac{1}{\sum_{i,j} a_{ij}^2} \underline{\lambda}_{\min}(AA^T), \quad (30)$$

we have

$$\mathbb{E}[\|\beta^+\|^2 \mid x] \leq \|\beta\|^2 \left(1 - \left(\frac{1}{\sum_{i,j} a_{ij}^2} \underline{\lambda}_{\min}(AA^T)\right)^2\right).$$

The claim is proved. □

Remark 2 Let us compare the rate obtained above with the rate of a randomized CD method (Algorithm 3.5 in [10]), 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 Eq. (1) with arbitrary non-zero matrix A, the randomized CD method achieves the rate

$$\mathbb{E}[\|\beta^+\|^2_{A^TA} \mid x] \leq \left(1 - \frac{\lambda_{\min}(A^TA)}{\|A\|_F^2}\right) \|\beta\|^2_{A^TA}. \tag{31}$$

Clearly the above rate is closely related to the one given in (24). □

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:

$$\text{Find } x \qquad \text{s.t.} \quad A_{i:}x \leq b_i, \quad i = 1, \ldots, m, \tag{32}$$

where we write the system of inequalities $Ax \leq b$ in the above form to emphasize the existence of m separate scalar inequalities. We also assume that this system of linear inequalities is feasible.

Our 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 an index pair (i, j) with probability

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

Update x_j by the following

$$x_j^{r+1} = x_j^r, \quad \text{if } A_{i:}x^r \leq b_i \tag{33}$$

$$\begin{aligned} x_j^{r+1} &= (1-\alpha)x_j^r + \alpha\left(\frac{b_i - \sum_{k\neq j} a_{ik}x_k^r}{a_{ij}}\right) \\ &= x_j^r + \alpha\left(\frac{b_i - \sum_{k=1} a_{ik}x_k^r}{a_{ij}}\right), \quad \text{otherwise.} \end{aligned} \tag{34}$$

For all $k, k \neq j$, set $x_k^{r+1} = x_k^r$.

To facilitate our analysis, let us define the following function

$$f(x) := \sum_{i=1}^{m} f_i(x), \tag{35}$$

where

$$f_i(x) := \frac{1}{2}(A_{i:}x - b_i)_+^2.$$

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

$$\mathcal{I}^r := \{i \mid A_{i:}x^r - b_i > 0\}, \tag{36}$$

and define $\Omega_{\mathcal{I}}^r \in \mathbb{R}^{m\times m}$ as the diagonal matrix with $\Omega_{\mathcal{I}}^r[i,i] = 1$ if $i \in \mathcal{I}^r$ and $\Omega_{\mathcal{I}}^r[i,i] = 0$ otherwise. Then we have

$$\sum_{i=1}^{m} f_i(x^r) = \sum_{i\in\mathcal{I}^r} f_i(x^r) = \frac{1}{2}\|A^r x^r - b^r\|^2 \tag{37}$$

where $A^r = \Omega_{\mathcal{I}}^r A \in \mathbb{R}^{m\times n}$, and b^r is defined similarly.

Note that by using the above definition, we have

$$\sum_{i=1}^{m} f_i(x^r) = \frac{1}{2}\|A^r x^r - b^r\|^2 \leq \frac{1}{2}\|Ax^r - b\|^2. \tag{38}$$

3.1 Case 1: *A* has full row rank

In this subsection we make the following assumption

$$\lambda_{\min}(AA^T) > 0. \tag{39}$$

Similarly as before, we will use x^+ (resp. x) to denote the new (resp. previous) iteration; we will use $A_{\mathcal{I}}, b_{\mathcal{I}}$ and $\Omega_{\mathcal{I}}$ to denote $A^r, b^r, \Omega_{\mathcal{I}}^r$ at iteration r, respectively.

Theorem 3 *Suppose A has full row rank, and* α *is chosen as*

$$\alpha < \frac{\lambda_{\min}(AA^T)}{\|A\|_F^2}. \tag{40}$$

Then we have

$$\mathbb{E}[f(x^+) \mid x] \leq \left(1 - \left(\frac{\lambda_{\min}(AA^T)}{2\|A\|_F^2}\right)^2\right) f(x) \tag{41}$$

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

$$x^+ = x + \alpha \left(\frac{b_i - \sum_k a_{ik}x_k}{a_{ij}}\right) e_j. \tag{42}$$

In this case, we can estimate the component function $f_\ell(x^+)$ based on whether the ℓth inequality is satisfied for x. Suppose that $\ell \in \mathcal{I}$ (i.e. the ℓth inequality is not satisfied), then we have

$$\begin{aligned} f_\ell(x^+) &= f_\ell\left(x + \alpha\left(\frac{b_i - \sum_k a_{ik}x_k}{a_{ij}}\right)e_j\right) \\ &\leq f_\ell(x) + \left\langle \nabla f_\ell(x), \alpha\left(\frac{b_i - \sum_k a_{ik}x_k}{a_{ij}}\right)e_j\right\rangle + \frac{1}{2}\left(\alpha a_{\ell j}\frac{b_i - \sum_k a_{ik}x_k}{a_{ij}}\right)^2 \\ &= f_\ell(x) + \alpha\left\langle A_{\ell:}^T(A_{\ell:}x - b_\ell), \left(\frac{b_i - \sum_k a_{ik}x_k}{a_{ij}}\right)e_j\right\rangle + \frac{1}{2}\left(\alpha a_{\ell j}\frac{b_i - \sum_k a_{ik}x_k}{a_{ij}}\right)^2 \end{aligned}$$

$$= f_\ell(x) + \alpha \left\langle a_{\ell j} \left(\sum_k a_{\ell k} x_k - b_\ell \right) e_j, \left(\frac{b_i - \sum_k a_{ik} x_k}{a_{ij}} \right) e_j \right\rangle + \frac{\alpha^2}{2} \left(a_{\ell j} \frac{b_i - \sum_k a_{ik} x_k}{a_{ij}} \right)^2. \tag{43}$$

Otherwise, if $\ell \notin \mathcal{I}$ (i.e. the ℓth inequality is satisfied), we have

$$\begin{aligned} f_\ell(x^+) &= f_\ell \left(x + \alpha \left(\frac{b_i - \sum_k a_{ik} x_k}{a_{ij}} \right) e_j \right) \\ &= \frac{1}{2} \left(A_{\ell:} x - b_\ell + \alpha A_{\ell:} \left(\frac{b_i - \sum_k a_{ik} x_k}{a_{ij}} \right) e_j \right)_+^2 \\ &\le f_\ell(x) + \frac{1}{2} \left(\alpha a_{\ell j} \frac{b_i - \sum_k a_{ik} x_k}{a_{ij}} \right)^2, \end{aligned} \tag{44}$$

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

Using the above inequalities, we have the following

$$\begin{aligned} \mathbb{E}\left[\sum_\ell f_\ell(x^+) \mid x \right] &= \sum_{\ell \in \mathcal{I}} \mathbb{E}\left[f_\ell(x^+) \mid x \right] + \sum_{\ell \notin \mathcal{I}} \mathbb{E}\left[f_\ell(x^+) \mid x \right] \\ &\overset{\text{(i)}}{\le} \sum_{\ell \in \mathcal{I}} \sum_{ij: i \in \mathcal{I}} p_{ij} \left(\alpha \left\langle a_{\ell j} \left(\sum_k a_{\ell k} x_k - b_\ell \right) e_j, \left(\frac{b_i - \sum_k a_{ik} x_k}{a_{ij}} \right) e_j \right\rangle \right) \\ &\quad + \sum_\ell \sum_{ij: i \in \mathcal{I}} p_{ij} \frac{\alpha^2}{2} \left(a_{\ell j} \frac{b_i - \sum_k a_{ik} x_k}{a_{ij}} \right)^2 + \sum_\ell \sum_{i,j} p_{ij} f_\ell(x) \\ &= \frac{\alpha}{\sum_{i,j} a_{ij}^2} \sum_{\ell \in \mathcal{I}} \sum_{ij: i \in \mathcal{I}} \left\langle a_{\ell j} \left(\sum_k a_{\ell k} x_k - b_\ell \right) e_j, a_{ij} \left(b_i - \sum_k a_{ik} x_k \right) e_j \right\rangle \\ &\quad + \sum_\ell \sum_{ij: i \in \mathcal{I}} \frac{\alpha^2}{2 \sum_{i,j} a_{ij}^2} \left(a_{\ell j} (b_i - \sum_k a_{ik} x_k) \right)^2 + \sum_\ell f_\ell(x) \\ &\overset{\text{(ii)}}{\le} \sum_{\ell \in \mathcal{I}} f_\ell(x) - \frac{\alpha}{\sum_{i,j} a_{ij}^2} \| A_{\mathcal{I}}^T (A_{\mathcal{I}} x - b_{\mathcal{I}}) \|^2 + \frac{\alpha^2}{2} \| A_{\mathcal{I}} x - b_{\mathcal{I}} \|^2 \\ &\le \left(1 - \frac{\alpha \lambda_{\min}(A_{\mathcal{I}} A_{\mathcal{I}}^T)}{\|A\|_F^2} + \frac{\alpha^2}{2} \right) \sum_\ell f_\ell(x) \le \left(1 - \frac{\alpha \lambda_{\min}(A A^T)}{\|A\|_F^2} + \frac{\alpha^2}{2} \right) \sum_\ell f_\ell(x) \end{aligned}$$

where in (i) we have used the two cases (43) and (44); in (ii) we have used the fact that $f_\ell(x) = 0$ for $\ell \notin \mathcal{I}$; in the last inequality we have used the fact that $A_{\mathcal{I}} A_{\mathcal{I}}^T$ is a principal submatrix of AA^T, the fact that $f_\ell(x) \geq 0$, and the fact that α is chosen small enough such that

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

This concludes the proof. □

3.2 Case 2: *A* does not have 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 $\mathcal{S}$ 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*

$$x \in \mathbb{R}^n, \ S \neq \emptyset \rightarrow \ dist(x, S) \leq \tau \|(Ax - b)^+\|, \tag{45}$$

where we have defined

$$(Ax - b)_i^+ = \max\{0, A_{i:}x - b\}, \quad dist(x, S) := \inf_{y \in S} \|x - y\|. \tag{46}$$

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*

$$\mathbb{E}[dist^2(x^+, S) \mid x] \leq \left(1 - \frac{1}{n\tau^2 \sum_{i,j} a_{ij}^2}\right) dist^2(x, S). \tag{47}$$

Proof Recall that from the update rule we have

$$x^+ = x + \alpha \left(\frac{b_i - \sum_k a_{ik} x_k}{a_{ij}}\right) e_j. \tag{48}$$

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

$$P(x) := \arg\min_{y \in S} \|x - y\|. \tag{49}$$

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

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

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

$$\begin{aligned}
\mathbb{E}\left[\|x^+ - x\|^2 \mid x\right] &= \sum_{(i,j):i\in\mathcal{I}} p_{ij}\alpha^2 \left\| \left(\frac{b_i - \sum_k a_{ik}x_k}{a_{ij}} \right) e_j \right\|^2 \\
&= \sum_{(i,j):i\in\mathcal{I}} \frac{1}{\sum_{i,j} a_{ij}^2}\alpha^2 \left\| \left(b_i - \sum_k a_{ik}x_k \right) e_j \right\|^2 \\
&= \sum_{i\in\mathcal{I}} \frac{1}{\sum_{i,j} a_{ij}^2}\alpha^2 n \left\| b_i - \sum_k a_{ik}x_k \right\|^2 \\
&= \frac{\alpha^2 n}{\sum_{i,j} a_{ij}^2} \sum_{i\in\mathcal{I}} \|A_{\mathcal{I}}x - b_{\mathcal{I}}\|^2 .
\end{aligned}$$

The second term in (50) is given by

$$\begin{aligned}
\mathbb{E}\left[\langle x^+ - x, x - P(x)\rangle \mid x\right] &= 2\alpha \sum_{(i,j):i\in\mathcal{I}} \left\langle p_{ij} \left(\frac{b_i - \sum_k a_{ik}x_k}{a_{ij}} \right) e_j, x - P(x) \right\rangle \\
&= 2\alpha \frac{1}{\sum_{i,j} a_{ij}^2} \sum_{(i,j):i\in\mathcal{I}} \left\langle \left(a_{ij}(b_i - \sum_k a_{ik}x_k) \right) e_j, x - P(x) \right\rangle \\
&= 2\alpha \frac{1}{\sum_{i,j} a_{ij}^2} \left\langle A_{\mathcal{I}}^T(b_{\mathcal{I}} - A_{\mathcal{I}}x), x - P(x) \right\rangle \\
&= -2\alpha \frac{1}{\sum_{i,j} a_{ij}^2} \langle \nabla f(x), x - P(x)\rangle \\
&\leq -2\alpha \frac{1}{\sum_{i,j} a_{ij}^2} (f(x) - f(P(x))) \\
&= -2\alpha \frac{1}{\sum_{i,j} a_{ij}^2} \|A_{\mathcal{I}}x - b_{\mathcal{I}}\|^2,
\end{aligned}$$

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

$$\mathbb{E}[\text{dist}^2(x^+, S) \mid x] \leq \frac{n\alpha^2 - 2\alpha}{\sum_{i,j} a_{ij}^2} \|A_{\mathcal{I}} x - b_{\mathcal{I}}\|^2 + \|x - P(x)\|^2.$$

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

$$\begin{aligned}\mathbb{E}[\text{dist}^2(x^+, S) \mid x] &\leq \frac{n\alpha^2 - 2\alpha}{\tau^2 \sum_{i,j} a_{ij}^2} \text{dist}^2(x, S) + \|x - P(x)\|^2 \\ &= \left(1 + \frac{n\alpha^2 - 2\alpha}{\tau^2 \sum_{i,j} a_{ij}^2}\right) \text{dist}^2(x, S) \\ &\overset{\alpha=\frac{1}{n}}{\leq} \left(1 - \frac{1}{n\tau^2 \sum_{i,j} a_{ij}^2}\right) \text{dist}^2(x, S). \end{aligned} \tag{50}$$

Therefore we conclude that the algorithm converges linearly in expectation. □

Remark 4 Let us compare the above result with the rate for a randomized iterative projection method (Algorithm 4.6) developed in [10] (cf. [10, Theorem 4.7]), which is given below

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

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 over-parameterized machine learning

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

$$\min_x \ f(x) := \sum_{i=1}^{m} f_i(x_1, \ldots, x_n) \tag{52}$$

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 non-negative, i.e.,

$$f_i(x) \geq 0, \quad \forall\, x \in \mathbb{R}^n.$$

(2) Assume that each f_i has Lipschitz continuous gradient with constant L_i, i.e.,

$$\|\nabla f_i(x) - \nabla f_i(z)\| \leq L_i \|x - z\|, \quad \forall\, x, z, \in \mathbb{R}^n, \quad i = 1, \cdots, n. \tag{53}$$

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

$$f(x) - f(P(x)) \leq \langle \nabla f(x), x - P(x) \rangle - \frac{\gamma}{2} \|P(x) - x\|^2, \tag{54}$$

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.,

$$\min_x f(x) = 0. \tag{55}$$

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

$$\nabla f_i(x^*) = 0, \quad \forall\, i. \tag{56}$$

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 $\mathcal{I} = \{i \mid a_i^T x - b_i \geq 0\}$. Noticing that $A_{\mathcal{I}} x - b_{\mathcal{I}} \geq 0$ and $A_{\mathcal{I}} P(x) - b_{\mathcal{I}} \leq 0$, we obtain

$$\langle A_{\mathcal{I}} x - b_{\mathcal{I}}, A_{\mathcal{I}} P(x) - b_{\mathcal{I}} \rangle \leq 0.$$

Adding the term $\|A_{\mathcal{I}} x - b_{\mathcal{I}}\|^2$ to both sides and rearranging the terms, we obtain

$$\|A_{\mathcal{I}} x - b_{\mathcal{I}}\|^2 \leq \langle A_{\mathcal{I}}^T \left(A_{\mathcal{I}} x - b_{\mathcal{I}}\right), x - P(x) \rangle. \tag{57}$$

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

$$\frac{\gamma}{2} \|x - P(x)\|^2 \leq \|A_{\mathcal{I}} x - b_{\mathcal{I}}\|^2. \tag{58}$$

Multiplying (57) by two and adding to (58), we get

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

i.e., (54) holds. □

Remark 6 We also note that the condition (54) is weaker than the traditional strong convexity, and it is also weaker than the essentially strong convexity condition defined in [8,12]. In particular, the essentially strong convexity requires that

$$f(x) - f(P(x)) \leq \langle \nabla f(x), x - y \rangle - \frac{\gamma}{2}\|x - y\|^2, \ \forall\, x, y, \ \text{s.t.} \quad P(x) = P(y). \tag{59}$$

The above condition clearly implies (54). Moreover, it is not hard to see that for any strongly convex function $h(\cdot)$, and any linear mapping A, the function $f(x) = h(Ax)$ satisfies (54). Notice that the condition (54) does not even imply the convexity of the function [16]. This condition is referred to as *quasi strong convexity* in [16] and as *weak strong convexity* in [8]. □

Remark 7 (*Connection to over-parameterized learning*) In training machine learning models via empirical risk minimization framework, the goal is to minimize the empirical risk loss

$$\min_x \ \sum_{i=1}^{m} \ell(a_i^T x, y_i), \tag{60}$$

where $\{(a_i, y_i)\}_{i=1}^m$ is the data and x is the parameter of the model. The loss function $\ell(\cdot, \cdot)$ is non-negative and $\ell(y', y) = 0$ if $y = y'$. In the over-parameterized regime where the number of parameters n is lager than the number of data points m, it is not hard to see that the minimum value of (60) is zero and hence (55) is satisfied. □

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

$$x_j^{r+1} = x_j^r - \alpha \nabla_j f_i(x^r), \tag{61}$$

and for all $k, k \neq j$, set

$$x_k^{r+1} = x_k^r \tag{62}$$

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 1 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

$$\mathbb{E}[f(x^+) \mid x] \leq \left(1 - \frac{\alpha\gamma}{mn}\right) f(x). \tag{63}$$

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

$$x^+ = x - \alpha \nabla f_i(x) e_j. \tag{64}$$

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

$$\begin{aligned} f_\ell(x^+) &= f_\ell(x - \alpha \nabla f_i(x) e_j) \\ &\leq f_\ell(x) - \langle \nabla f_\ell(x), \alpha \nabla f_i(x) e_j \rangle + \frac{L_\ell \alpha^2}{2} \|\nabla f_i(x) e_j\|^2. \end{aligned}$$

Therefore we obtain the following relations:

$$\begin{aligned} \mathbb{E}[f(x^+) \mid x] &\leq \frac{1}{mn} \sum_{\ell \in [m]} \sum_{i,j} \Big(- \alpha \langle \nabla f_\ell(x) e_j, \nabla f_i(x) e_j \rangle \\ &\quad + \frac{L_\ell \alpha^2}{2} \|\nabla f_i(x) e_j\|^2 \Big) + f(x) \\ &= \frac{1}{mn} \sum_{\ell \in [m]} \sum_{i,j} \left(-\alpha \langle \nabla f_\ell(x) e_j, \nabla f_i(x) e_j \rangle \right) \\ &\quad + \frac{\sum_\ell L_\ell \alpha^2}{2mn} \sum_{i=1}^n \|\nabla f_i(x)\|^2 + f(x) \\ &= f(x) - \frac{\alpha}{mn} \left\| \sum_{i=1}^m \nabla f_i(x) \right\|^2 + \frac{\sum_\ell L_\ell \alpha^2}{2mn} \sum_{i=1}^n \|\nabla f_i(x)\|^2 \\ &\overset{(56)}{=} f(x) - \frac{\alpha}{mn} \|\nabla f(x)\|^2 + \frac{\sum_\ell L_\ell \alpha^2}{2mn} \sum_{i=1}^n \|\nabla f_i(x) - \nabla f_i(x^*)\|^2 \\ &\overset{(53)}{\leq} f(x) - \frac{\alpha}{mn} \|\nabla f(x)\|^2 + \frac{\sum_\ell L_\ell \alpha^2}{2mn} \sum_{i=1}^n L_i^2 \|x - x^*\|^2. \end{aligned} \tag{65}$$

Using the property (54), we have

$$\begin{aligned} f(x) - f(x^*) &\leq \langle \nabla f(x), x - x^* \rangle - \frac{\gamma}{2} \|x - x^*\|^2 \\ &\leq \frac{1}{\gamma} \|\nabla f(x)\|^2 + \frac{\gamma}{4} \|x - x^*\|^2 - \frac{\gamma}{2} \|x - x^*\|^2 \\ &= \frac{1}{\gamma} \|\nabla f(x)\|^2 - \frac{\gamma}{4} \|x - x^*\|^2. \end{aligned} \tag{66}$$

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

$$-\frac{\alpha}{mn}\|\nabla f(x)\|^2 \leq -\frac{\alpha\gamma}{mn} f(x) - \frac{\gamma^2}{4}\frac{\alpha}{mn}\|x - x^*\|^2. \tag{67}$$

Plugging this inequality into (65), we obtain

$$\mathbb{E}[f(x^+) \mid x] \leq \left(1 - \frac{\alpha\gamma}{mn}\right) f(x) - \left(\frac{\gamma^2}{4}\frac{\alpha}{mn} - \frac{\sum_\ell L_\ell \alpha^2}{2mn}\sum_{i=1}^N L_i^2\right)\|x - x^*\|^2. \tag{68}$$

Therefore by choosing the following constant

$$\alpha = \frac{\gamma^2}{2\sum_\ell L_\ell \sum_{i=1}^N L_i^2} \tag{69}$$

we obtain the desired result. □

It should be noted that stochastic block coordinate descent methods or primal-dual methods have been proposed in the literature for optimization problems where only one function and one variable is sampled in each iteration. However, in order to obtain a linear rate of convergence, they either need to explicitly involve the dual variables [25] and/or apply variance reduction techniques by maintaining a running average of the gradients [26]. In contrast, Algorithm 3 works with only primal variables, does not maintain a running average of gradients, and does not require strong convexity assumption. In addition, in the case of system of linear equations, Algorithm 1 allows exact minimization per-iteration by non-uniform sampling of the equations and variables, while the works mentioned above does not necessarily converge with the exact optimization per iteration.

5 Numerical results

In this section, we evaluate the performance of the doubly stochastic G–S algorithm proposed in this paper against some of the existing algorithms, for solving linear systems of equations (1). In particular, the performance of DSGS is compared with S2CD [9], RK [23], RGS [14], and the REK method [27]. Note that except for the S2CD algorithm, the rest of the algorithms are designed specifically for solving linear system of equations. The S2CD algorithm 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, in all algorithms, we increase the iteration counters when n updates of the variable coordinates are completed. We divide our experiments into two broad categories: when $A^T A$ is PD and when $A^T A$ is PSD.

In the first set of experiments, we consider over-determined systems of equations 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 $f_i(x) = \|A_{i:}x - b_i\|^2$. Similar to [23,27], the elements of matrix A are independently drawn from standard Gaussian distribution $N(0, 1)$. Clearly, the condition number of such a matrix changes with the size. A randomly generated vector whose elements are drawn from standard Gaussian $N(0, 1)$ is considered as x^*. Accordingly, the vector b is found such that $b = Ax^*$. We terminate an algorithm when it reaches an error value of $\|x - x^*\| \leq 10^{-10}$. Each entry in the table is computed by averaging the results for running a given algorithm over ten randomly generated problems. The result of this experiment is summarized in Table 1. As can

Table 1 The average number of required iterations to reach the error level of 10^{-10} for an overdetermined system with a random coefficient matrix A

Size ($m \times n$)	DSGS	RK	RGS	REK	S2CD
40×20	**3612**	4334	4342	4949	8671
40×30	**17,211**	29,653	30,174	34,067	59,892
60×30	7715	**7205**	7230	8348	14,392
60×50	**36,503**	41,638	41,829	47,545	81,542
80×40	13,262	**9026**	9058	10,591	18,195
80×60	**83,857**	84,974	84,871	95,807	166,224

In each row, the bold number highlights the best obtained performance among all algorithms

Table 2 The average number of required iterations to reach the error level of 10^{-10} for an overdetermined system with a certain given condition number

Size ($m \times n$)	$k(A)$	DSGS	RK	RGS	REK
40×20	10^1	**9251**	16,731	16,324	19,813
40×20	10^2	**429,510**	1,521,400	1,530,549	1,779,000
40×20	10^3	**42,503,000**	147,368,000	152,590,000	175,640,000
40×30	10^1	**18,457**	24,742	24,668	29,461
40×30	10^1	**685,800**	2,216,400	2,217,500	2,615,400
40×30	10^3	**65,444,000**	220,290,000	217,826,000	263,440,000
60×30	10^1	**18,512**	24,338	24,569	28,206
60×30	10^2	**756,110**	2,206,800	2,262,800	2,708,700
60×30	10^3	**66,628,000**	212,990,000	220,510,000	256,190,000
60×50	10^1	**39,538**	40,330	40,930	47,523
60×50	10^2	**1,362,900**	3,644,200	3,797,800	4,407,700
60×50	10^3	**89,837,000**	363,740,000	366,690,000	417,850,000
80×40	10^1	**28,435**	31,764	32,177	38,858
80×40	10^2	**911,460**	2,989,500	2,927,400	3,486,000
80×40	10^3	**72,912,000**	296,892,000	298,320,000	359,853,000
80×60	10^1	54,069	49,828	**47,672**	55,977
80×60	10^2	**1,509,100**	4,284,300	4,469,300	5,270,600
80×60	10^3	**120,660,000**	437,880,000	452,480,000	530,308,000

In each row, the bold number highlights the best obtained performance among all algorithms

Table 3 The average required CPU time in seconds to reach the error level of 10^{-10} for an overdetermined system

Size ($m \times n$)	$k(A)$	DSGS	RK	RGS	REK
40×20	10^1	0.3750	0.9175	1.0575	**0.0875**
40×20	10^2	17.9780	**7.44**	13.83	7.9240
40×20	10^3	1517.7	**570.832**	1154.8	798.3517
40×30	10^1	1.5960	1.0190	1.2033	**0.1880**
40×30	10^2	59.6680	**12.9630**	26.8070	16.1760
40×30	10^3	5618	**1216.6**	2393.3	1648.9
60×30	10^1	2.3450	1.0780	1.2530	**0.2520**
60×30	10^2	93.8430	**19.0680**	35.4230	23.9490
60×30	10^3	8493.8	1986.6	3691.2	**1986.6**
60×50	10^1	12.6430	1.4200	2.0288	**0.6687**
60×50	10^2	463.2130	**49.4850**	107.6980	61.9989
60×50	10^3	26,367	**5026.4**	10,146	6,191
80×40	10^1	9.0680	1.3422	1.6833	**0.5980**
80×40	10^2	270.1660	**44.0870**	93.7690	53.4530
80×40	10^3	21,890	**4528.5**	9,194.9	5,559
80×60	10^1	3.4236	1.8625	2.9329	**1.2114**
80×60	10^2	966.9200	**88.9450**	196.6740	117.0433
80×60	10^3	77,498	**9090.2**	19,525	12,155

In each row, the bold number highlights the best obtained performance among all algorithms

Table 4 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
40×60	**11,354**	23,257	23,097	26,506
40×100	**3141**	6297	6267	6971
80×100	**65,030**	124,724	128,957	144,894
80×140	**15,171**	30,664	30,168	34,608
160×200	**151,858**	274,537	281,331	328,680
160×300	**25,692**	49,791	49,908	56,438

In each row, the bold number highlights the best obtained performance among all algorithms

be seen from this table, when matrix A gets close to a square matrix, the number of required iterations increases. This is reasonable since the condition numbers of our random matrix increases. Moreover, for these harder cases, 40×30, 60×50 and 80×60, the proposed approach in this paper slightly outperforms other algorithms.

We also test our algorithm on problems instances generated with different given condition numbers. To do so, we first generate a zero-mean, unit-variance random Gaussian matrix similar to the previous experiment. Then, we use SVD to modify the singular values to obtain a matrix with a desired condition number. This is done

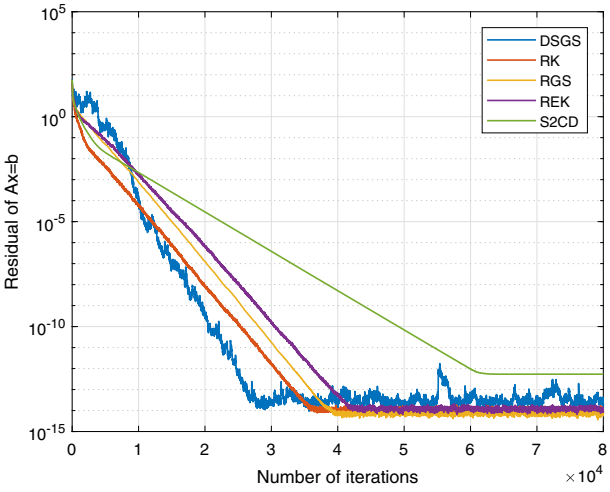

Fig. 3 Convergence behavior of different algorithms for an underdetermined system of size 40×100

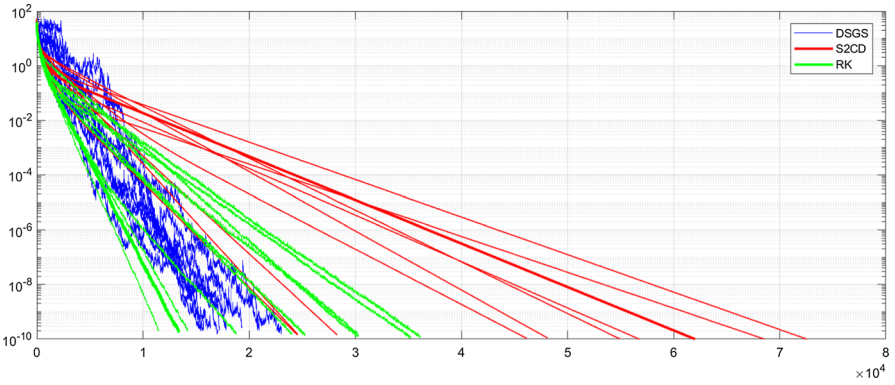

Fig. 4 Convergence behavior for DSGS, RK and RGS algorithms for a linear system of size 40×30. In this figure, 10 realizations are shown for each of the algorithms

by linearly scaling the differences of all singular values with the smallest non-zero singular value. Tables 2 and 3 summarize the result of our experiment. In particular, Table 2 shows the number of required iterations, while Table 3 presents the required time. We observe that as the condition number grows, the number of required iterations and CPU time increases. DSGs requires a fewer number of iterations compared to other algorithms to reach an error of 10^{-10} in almost all scenarios. In terms of CPU time, however, DSGS is not the fastest among algorithms considered.

In the second part of our experiments, we consider under-determined systems with $m \leq n$. Similar to the previous case, we first generate the entries of A independently according to a zero-mean, unit-variance random Gaussian distribution. Here, we do not include S2CD since it only works for strongly convex problems. The termination rule in this case is chosen as $\|Ax - b\| \leq 10^{-10}$. It is observed from Table 4 that for all

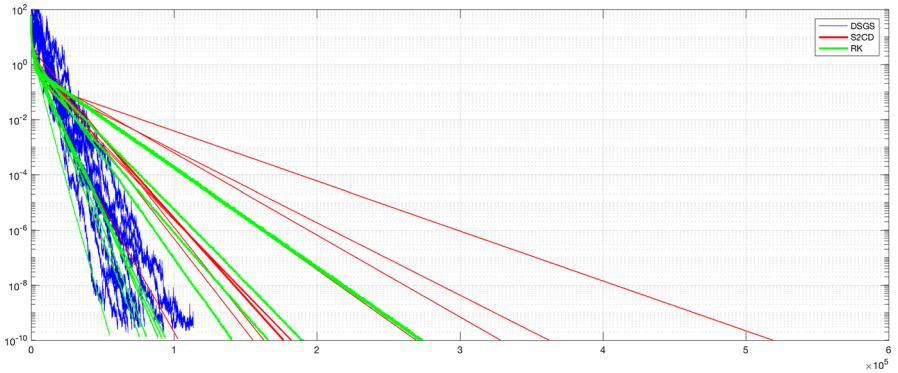

Fig. 5 Convergence rate for DSGS, RK and RGS algorithms for a linear system of size 60×50. In this figure, 10 realizations are shown for each of the algorithms

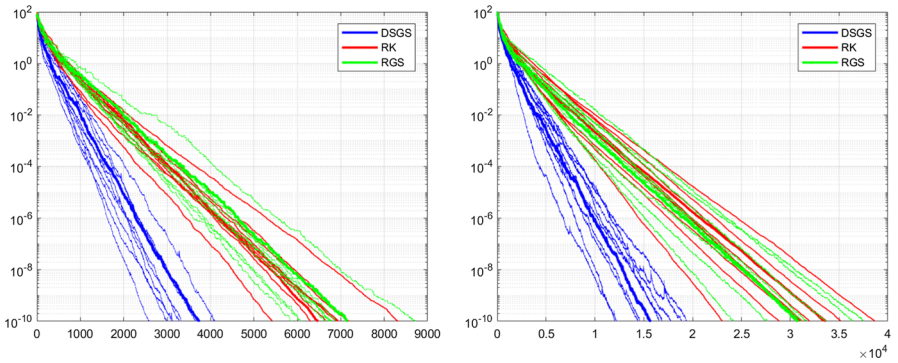

Fig. 6 Convergence rate for DSGS, RK and RGS algorithms for a linear system with a matrix of size 40×100 and 80×140

tested settings, the proposed DSGS algorithm outperforms other algorithms. Similar to the previous case, we compare the performance of different algorithms for different conditions numbers in Figs. 5 and 6.

We have also illustrated the convergence of various algorithms in terms of least squares error versus iteration number in Fig. 3. This plot is generated for a random Gaussian matrix of size 40×30. It is observed that for the considered linear system with random matrix, DSGS slightly outperforms other algorithms. Since DSGS chooses each row and coordinate to update randomly, the residual of least squares in DSGS algorithm does not decrease monotonically. We have repeated this experiment in Figs. 4 and 5 over ten different realization of matrix A. The linear system in Fig. 4 is of size 40×30 and it is seen that DSGS and RK have close convergence behavior and both faster than that of S2CD. We increase the size of matrix A to 60×50 and similar observation has been made in Fig. 5.

Finally, in Fig. 3 we compare the convergence behavior of different algorithms for an under-determined system of size 40×100. Similar to the results given in Table 4, it is seen that DSGD requires smaller number of iterations to converge. We also plot

Table 5 The average number of required iterations to reach an error of 10^{-10} for an underdetermined system with a certain given condition number

Size ($m \times n$)	$k(A)$	DSGS	RK	RGS	REK
40×60	10^1	**15,011**	32,261	32,901	37,957
40×60	10^2	**1,373,800**	3,304,000	3,468,600	3,999,500
40×60	10^3	**137,686,180**	305,210,000	306,252,290,	354,600,000
80×140	10^1	**29,600**	65,831	66,201	76,367
80×140	10^2	**2,829,840**	5,465,486	6,018,850	7,000,930
80×140	10^3	**249,641,600**	549,779,370	572,884,321	726,486,714

In each row, the bold number highlights the best obtained performance among all algorithms

Table 6 The average required CPU time in seconds to reach an error of 10^{-10} for an underdetermined system

Size ($m \times n$)	$k(A)$	DSGS	RK	RGS	REK
40×60	10^1	3.72	1.1360	1.6080	**0.4620**
40×60	10^2	**42.9820**	52.9140	120.4680	61.6675
40×60	10^3	41,824	**3195**	6589	3971
80×140	10^1	**3.4400**	4.1980	8.8	4.1400
80×140	10^2	332.2200	**158.7440**	368.0673	208.7914
80×140	10^3	27,558	**23,871**	47,411	31,097

In each row, the bold number highlights the best obtained performance among all algorithms

a number of different cases with 10 realizations for each of the algorithms in Figs. 4, 5 and 6. For under-determined systems with different condition numbers, Tables 5, 6 summarize the average required iterations and CPU times, respectively.

References

1. Bertsekas, D.P., Tsitsiklis, J.N.: Parallel and Distributed Computation: Numerical Methods, 2nd edn. Athena Scientific, Belmont (1997)
2. Deutsch, F.: Best Approximation in Inner Product Spaces. Springer, New York (2001)
3. Golub, G., van Loan, C.: Matrix Computation. Johns Hopkins University Press, Baltimore (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 J. Sci. Comput. **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. Int. J. Control **57**(6), 1269–1271 (1993). Translated from German original of 1933
8. Karimi, H., Nutini, J., Schmidt, M.: Linear convergence of gradient and proximal-gradient methods under the Polyak-Łojasiewicz condition. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 795–811. Springer (2016)
9. Konecny, J., Qu, Z., Richtarik, P.: Semi-stochastic coordinate descent (2014). Preprint arXiv:1412.6293

10. Leventhal, D., Lewis, A.S.: Randomized methods for linear constraints: convergence rates and conditioning. Math. Oper. Res. **35**(3), 641–654 (2010)
11. Liu, J., Wright, S.: An accelerated randomized Kaczmarz algorithm. Math. Comput. **85**, 153–178 (2016)
12. Liu, J., Wright, S.J., Ré, C., Bittorf, V.: An asynchronous parallel stochastic coordinate descent algorithm. In: The Proceedings of the International Conference on Machine Learning (ICML) (2014)
13. Loera, J.A.D., Haddock, J., Needell, D.: A sampling Kaczmarz–Motzkin algorithm for linear feasibility. SIAM J. Sci. Comput. **39**(5), S66–S87 (2017)
14. Ma, A., Needell, D., Ramdas, A.: Convergence properties of the randomized extended Gauss–Seidel and Kaczmarz methods. SIAM J. Matrix Anal. Appl. **36**(4), 1590–1604 (2015)
15. Motzkin, T.S., Schoenberg, I.J.: The relaxation method for linear inequalities. Can. J. Math. **6**, 393–404 (1954)
16. Necoara, I., Nesterov, Y., Glineur, F.: Linear convergence of first order methods for non-strongly convex optimization. In: Mathematical Programming, pp. 1–39 (2018)
17. Needell, D.: Randomized Kaczmarz solver for noisy linear systems. BIT Numer. Math. **50**(2), 395–403 (2010)
18. Needell, D., Tropp, J.A.: Paved with good intentions: analysis of a randomized block Kaczmarz method. Linear Algebra Appl. **441**, 199–221 (2013)
19. Ortega, J.M.: Numerical analysis: a second course. SIAM, Philadelphia (1990)
20. Oswald, P., Zhou, W.: Convergence analysis for Kaczmarz-type methods in a Hilbert space framework. Linear Algebra Appl. **478**, 131–161 (2015)
21. Oswald, P., Zhou, W.: Random reordering in SOR-type methods. Numer. Math. **135**(4), 1207–1220 (2017)
22. Saad, Y.: Iterative Methods for Sparse Linear Systems. Society for Industrial and Applied Mathematics, Philadelphia (2003)
23. Strohmer, T., Vershynin, R.: A randomized Kaczmarz algorithm with exponential convergence. J. Fourier Anal. Appl. **15**(2), 262 (2008)
24. Wheaton, I., Awoniyi, S.: A new iterative method for solving non-square systems of linear equations. J. Comput. Appl. Math. **322**(Supplement C), 1–6 (2017)
25. Yu, A.W., Lin, Q., Yang, T.: Doubly stochastic primal–dual coordinate method for bilinear saddle-point problem (2015). ArXiv preprint arXiv:1508.03390
26. Zhang, A., Gu, Q.: Accelerated stochastic block coordinate descent with optimal sampling. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 2035–2044. ACM (2016)
27. Zouzias, A., Freris, N.M.: Randomized extended Kaczmarz for solving least squares. SIAM J. Matrix Anal. Appl. **34**(2), 773–793 (2013)

Affiliations

Meisam Razaviyayn[1] · Mingyi Hong[2] · Navid Reyhanian[2] · Zhi-Quan Luo[3]

✉ Meisam Razaviyayn
razaviya@usc.edu

Mingyi Hong
mhong@umn.edu

Navid Reyhanian
navid@umn.edu

Zhi-Quan Luo
luozq@cuhk.edu.cn

1 Department of Industrial and Systems Engineering, University of Southern California, Los Angeles, CA 90089, USA

2 Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455, USA

3 Shenzhen Research Institute of Big Data, The Chinese University of Hong Kong, Shenzhen, China