SIAM J. MATRIX ANAL. APPL.
Vol. 39, No. 1, pp. 104–122

ITERATIVE METHODS FOR SOLVING FACTORIZED LINEAR SYSTEMS[*]

ANNA MA[†], DEANNA NEEDELL[‡], AND AADITYA RAMDAS[§]

Abstract. Stochastic iterative algorithms such as the Kaczmarz and Gauss–Seidel methods have gained recent attention because of their speed, simplicity, and the ability to approximately solve large-scale linear systems of equations without needing to access the entire matrix. In this work, we consider the setting where we wish to solve a linear system in a large matrix $\boldsymbol{X}$ that is stored in a factorized form, $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{V}$; this setting either arises naturally in many applications or may be imposed when working with large low-rank datasets for reasons of space required for storage. We propose a variant of the randomized Kaczmarz method for such systems that takes advantage of the factored form and avoids computing $\boldsymbol{X}$. We prove an exponential convergence rate and supplement our theoretical guarantees with experimental evidence demonstrating that the factored variant yields significant acceleration in convergence.

Key words. factorized linear systems, randomized iterative algorithms, Kaczmarz, Gauss–Seidel

AMS subject classifications. 68W20, 65F10

DOI. 10.1137/17M1115678

1. Introduction.

Recently, revived interest in stochastic iterative methods like the Kaczmarz [11, 25, 22, 23] and Gauss–Seidel [6, 19] methods has grown due to the need for large-scale approaches for solving linear systems of equations. Such methods utilize simple projections and require access to only a single row in a given iteration, hence having a low memory footprint. For this reason, they are very efficient and practical for solving extremely large, usually highly overdetermined, linear systems. In this work, we consider algorithms for solving linear systems when the matrix is available in a factorized form. As we discuss below, such a factorization may arise naturally in the application or may be constructed explicitly for efficient storage and computation. We seek a solution to the original system directly from its factorized form, without the need to perform matrix multiplication.

To that end, borrowing the notation of linear regression from statistics, suppose we want to solve the linear system $\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{y}$ with $\boldsymbol{X} \in \mathbb{C}^{m\times n}$. However, instead of the full system $\boldsymbol{X}$, we only have access to $\boldsymbol{U}, \boldsymbol{V}$ such that $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{V}$. In this case, we want to solve the linear system:

$$\boldsymbol{U}\boldsymbol{V}\boldsymbol{\beta} = \boldsymbol{y}, \tag{1}$$

where $\boldsymbol{U} \in \mathbb{C}^{m\times k}$ and $\boldsymbol{V} \in \mathbb{C}^{k\times n}$. Instead of taking the product of $\boldsymbol{U}$ and $\boldsymbol{V}$, to form $\boldsymbol{X}$, which may not be desirable, we approach this problem using stochastic iterative

[*]Received by the editors February 9, 2017; accepted for publication (in revised form) by P. Drineas September 28, 2017; published electronically January 18, 2018.
http://www.siam.org/journals/simax/39-1/M111567.html
Funding: The work of the second author was partially supported by NSF CAREER grant 1348721 and the Alfred P. Sloan Fellowship. The work of the first author was supported in part by NSF CAREER grant 1348721, the CSRC Intellisis Fellowship, and the Edison International Scholarship.

[†]Claremont Graduate University, Claremont, CA 91711 (anna.ma@cgu.edu).
[‡]University of California Los Angeles, Los Angeles, CA 90024 (deanna@math.ucla.edu).
[§]University of California Berkeley, Berkeley, CA 94720 (aramdas@berkeley.edu).

methods to solve the individual subsystems

$$\boldsymbol{U}\boldsymbol{x} = \boldsymbol{y}, \tag{2}$$

$$\boldsymbol{V}\boldsymbol{\beta} = \boldsymbol{x}, \tag{3}$$

in an alternating fashion. Note that $\boldsymbol{\beta}$ in (3) is the vector of unknowns that we want to solve for in (1) and $\boldsymbol{y}$ in (2) is the known right-hand side vector of (1). If we substitute (3) into (2), we acquire the full linear system (1). We will often refer to (1) as the "full system" and (2) and (3) as "subsystems" and say that a system is consistent if it has at least one solution (and inconsistent otherwise).

There are some situations when approximately knowing $\boldsymbol{x}$ would suffice. We assume that (for reasons of interpretability, or for downstream usage) the scientist is genuinely interested in solving the full system, i.e., she is interested in the vector $\boldsymbol{\beta}$, not in $\boldsymbol{x}$.

It is arguably of practical interest to give special importance to the case of $k < \min(m, n)$, which arises in modern data science as motivated by the following examples, but we discuss other settings later.

1.1. Motivation. If $\boldsymbol{X}$ is large and low-rank, one may have many reasons to work with a factorization of $\boldsymbol{X}$. We shall discuss three reasons below—algorithmic, infrastructural, and statistical.

Consider data matrices encountered in "recommender systems" in machine learning [2, 14, 20, 26, 27]. For concreteness, consider the Netflix (or Amazon, or Yelp) problem, where one has a users-by-movies matrix whose entries correspond to ratings given by users to movies. $\boldsymbol{X}$ is usually quite well approximated by low-rank matrices—intuitively, many rows and columns are redundant because every row is usually similar to many other rows (corresponding to users with similar tastes), and every column is usually similar to many other columns (corresponding to similar quality movies in the same genre). Usually we have observed only a few entries of $\boldsymbol{X}$ and wish to infer the unseen ratings in order to provide recommendations to different users based on their tastes. Algorithms for "low-rank matrix completion" have proved to be quite successful in the applied and theoretical machine learning community [5, 13, 28, 12]. One popular algorithm, alternating-minimization [10], chooses a (small) target rank k and tries to find $\boldsymbol{U}, \boldsymbol{V}$ such that $\boldsymbol{X}_{ij} \approx (\boldsymbol{U}\boldsymbol{V})_{ij}$ for all the observed entries (i, j) of $\boldsymbol{X}$. As its name suggests, the algorithm alternates between solving for $\boldsymbol{U}$ keeping $\boldsymbol{V}$ fixed and then solving for $\boldsymbol{V}$ keeping $\boldsymbol{U}$ fixed. In this case, at no point does the algorithm even form the entire completed (inferred) matrix $\boldsymbol{X}$, and the algorithm only has access to factors $\boldsymbol{U}, \boldsymbol{V}$ simply due to *algorithmic* choices.

There may be other instances where a data scientist may have access to the full matrix $\boldsymbol{X}$, but in order to reduce the memory storage footprint, or to communicate the data, may explicitly choose to decompose $\boldsymbol{X} \approx \boldsymbol{U}\boldsymbol{V}$ and discard $\boldsymbol{X}$ to work with the smaller matrices instead.

Consider an example motivated by "topic modeling" of text data. Suppose Google has scraped the internet for English documents (or maybe a subset of documents like news articles), to form a document-by-word matrix $\boldsymbol{X}$, where each entry of the matrix indicates the number of times that word occurred in that document. Since many documents could be quite similar in their content (like articles about the same incident covered by different newspapers), this matrix is easily seen to be low-rank. This is a classic setting for applying a machine learning technique called "nonnegative matrix factorization" [15, 29, 18], where one decomposes $\boldsymbol{X}$ as the product of two low-rank nonnegative matrices $\boldsymbol{U}, \boldsymbol{V}$; the nonnegativity is imposed for human interpretability, so that $\boldsymbol{U}$ can be interpreted as a documents-by-topics matrix, and $\boldsymbol{V}$ as a topics-by-words. In this case, we do not have access to $\boldsymbol{X}$ as a result of *systems infrastructure* constraints (memory/storage/communication).

Often, even for modestly sized data matrices, the relevant "signal" is contained in the leading singular vectors corresponding to large singular values, and the tail of small singular values is often deemed to be "noise." This is precisely the idea behind the classical topic of PCA, and the modern machine learning literature has proposed and analyzed a variety of algorithms to approximate the top k left and right singular vectors in a streaming/stochastic/online fashion [7]. Hence, the factorization may arise from a purely *statistical* motivation.

Given a vector $\boldsymbol{y}$ (representing age, or document popularity, for example), suppose the data scientist is interested in regressing $\boldsymbol{X}$ onto $\boldsymbol{y}$, for the purpose of scientific understanding or to take future actions. Can we utilize the available factorization efficiently, designing methods that work directly on the lower dimensional factors $\boldsymbol{U}$ and $\boldsymbol{V}$ rather than computing the full system $\boldsymbol{X}$?

Our goal will be to propose iterative methods that work directly on the factored system, eliminating the need for a full matrix product and potentially saving computations on the much larger full system.

1.2. Main contribution. We propose two stochastic iterative methods for solving system (1) without computing the product of $\boldsymbol{U}$ and $\boldsymbol{V}$. Both methods utilize iterates of well studied algorithms for solving linear systems. When the full system is consistent, the first method, called RK-RK, interlaces iterates of the randomized Kaczmarz (RK) algorithm to solve each subsystem and finds the optimal solution. When the full system is inconsistent, we introduce the REK-RK method, an interlacing of randomized extended Kaczmarz (REK) iterates to solve (2) and RK iterates to solve (3), that converges to the so-called ordinary least squares solution. We prove linear ("exponential") convergence to the solution in both cases.

1.3. Outline. In the next section, we provide background and discuss existing work on stochastic methods that solve linear systems. In particular, we describe the RK and REK algorithms as well as the randomized Gauss–Seidel (RGS) and randomized extended Gauss–Seidel (REGS) algorithms. In section 3 we investigate variations of settings for subsystems (2) and (3) that arise depending on the consistency and size of $\boldsymbol{X}$. Section 4 introduces our proposed methods, RK-RK and REK-RK. We provide theory that shows linear convergence in expectation to the optimal solution for both methods. Finally, we present experiments in section 5 and conclude with final remarks and future work in section 6.

1.4. Notation. Here and throughout the paper, matrices and vectors are denoted with boldface letters (uppercase for matrices and lowercase for vectors). We call $\boldsymbol{X}^i$ the i^{th} row of the matrix $\boldsymbol{X}$ and $\boldsymbol{X}_{(j)}$ the j^{th} column of $\boldsymbol{X}$. The Euclidean norm is denoted by $\|\cdot\|_2$ and the Frobenius norm by $\|\cdot\|_F$. Last, $\boldsymbol{X}^*$ denotes the adjoint (conjugate transpose) of the matrix $\boldsymbol{X}$. Motivated by applications, we allow $\boldsymbol{X}$ to be rank deficient and assume that $\boldsymbol{U}$ and $\boldsymbol{V}$ are full rank.

2. Background and existing work.

In this section we summarize existing work on stochastic iterative methods and different variations of linear systems.

2.1. Linear systems. Linear systems take on one of three settings determined by the size of the system, rank of the matrix $\boldsymbol{X}$, and the existence of a solution. First we discuss solutions to systems with full rank matrices $\boldsymbol{X}$ and then remark on how rank deficiency affects the desired solution.

In the full rank underdetermined case, $m < n$ and the system has infinitely many solutions; here, we often want to find the least Euclidean norm solution to (1):

$$\boldsymbol{\beta_{LN}} := \boldsymbol{X}^*(\boldsymbol{X}\boldsymbol{X}^*)^{-1}\boldsymbol{y}. \tag{4}$$

Clearly, $\boldsymbol{X}\boldsymbol{\beta_{LN}} = \boldsymbol{y}$, and all other solutions to an underdetermined system can be written as $\boldsymbol{b} = \boldsymbol{\beta_{LN}} + \boldsymbol{z}$, where $\boldsymbol{X}\boldsymbol{z} = \boldsymbol{0}$.

In the overdetermined setting, we have $m > n$ and the system can have a unique (exact) solution or no solution. If there is a unique solution, the linear system is called an overdetermined *consistent* system. When $\boldsymbol{X}$ is full rank, the optimal unique solution is $\boldsymbol{\beta}_{\text{uniq}}$ such that $\boldsymbol{X}\boldsymbol{\beta}_{\text{uniq}} = \boldsymbol{y}$:

$$\boldsymbol{\beta}_{\text{uniq}} := (\boldsymbol{X}^*\boldsymbol{X})^{-1}\boldsymbol{X}^*\boldsymbol{y}. \tag{5}$$

If there is no exact solution, the system is called an overdetermined *inconsistent* system. When a system is inconsistent and $\boldsymbol{X}$ is full rank, we often seek to minimize the sum of squared residuals, i.e., to find the ordinary least squares solution

$$\boldsymbol{\beta_{LS}} := (\boldsymbol{X}^*\boldsymbol{X})^{-1}\boldsymbol{X}^*\boldsymbol{y}. \tag{6}$$

The residual can be written as $\boldsymbol{r} = \boldsymbol{X}\boldsymbol{\beta_{LS}} - \boldsymbol{y}$. Note that $\boldsymbol{X}^*\boldsymbol{r} = \boldsymbol{0}$, which can be easily seen by substituting $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta_{LS}} + \boldsymbol{r}$ into (6). For simplicity, we will refer to the matrix $\boldsymbol{X}$ of a linear system as consistent or inconsistent when the system itself is consistent or inconsistent.

If the matrix $\boldsymbol{X}$ in the linear system $\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{y}$ is rank deficient, then there are infinitely many solutions to the system regardless of the size of m and n. In this case, we again want the least norm solution in the underdetermined case and the "least-norm least-squares" solution in the overdetermined case,

$$\boldsymbol{\beta_{LN}} := \begin{cases} \boldsymbol{X}^*(\boldsymbol{X}\boldsymbol{X}^*)^{\dagger}\boldsymbol{y} & \text{if } m < n, \\ (\boldsymbol{X}^*\boldsymbol{X})^{\dagger}\boldsymbol{X}^*\boldsymbol{y} & \text{if } m > n, \end{cases} \tag{7}$$

where $(\cdot)^{\dagger}$ is the pseudoinverse. General solutions to the linear system can be written as $\boldsymbol{b} = \boldsymbol{\beta_{LN}} + \boldsymbol{z}$, where $\boldsymbol{X}\boldsymbol{z} = \boldsymbol{0}$—note that $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{b} = \boldsymbol{X}\boldsymbol{\beta_{LN}} + \boldsymbol{X}\boldsymbol{z} = \boldsymbol{X}\boldsymbol{\beta_{LN}}$. Similar to the full rank case, when the low-rank system is inconsistent, we can write $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta_{LN}} + \boldsymbol{r}$, again where $\boldsymbol{X}^*\boldsymbol{r} = \boldsymbol{0}$.

2.2. Randomized Kaczmarz and its extension. The Kaczmarz algorithm [11] solves a linear system $\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{y}$ by cycling through rows of $\boldsymbol{X}$ and projects the estimate onto the solution space given by the chosen row. It was initially proposed by Kaczmarz [11] and has recently regained interest in the setting of computer tomography, where it is known as the algebraic reconstruction technique [8, 21, 4, 9]. The randomized variant of the Kaczmarz method introduced by Strohmer and Vershynin [25] was proved to converge linearly in expectation for consistent systems. Formally, given $\boldsymbol{X}$ and $\boldsymbol{y}$ of (1), RK chooses row $i \in \{1, 2, ...m\}$ of $\boldsymbol{X}$ with probability $\frac{\|\boldsymbol{X}^i\|_2^2}{\|X\|_F^2}$ and projects the previous estimate onto that row with the update

$$\boldsymbol{\beta_t} := \boldsymbol{\beta_{t-1}} + \frac{(\boldsymbol{y}_i - \boldsymbol{X}^i\boldsymbol{\beta_{t-1}})}{\left\|\boldsymbol{X}^i\right\|_2^2}(\boldsymbol{X}^i)^*.$$

Needell [22] later studied the inconsistent case and showed that RK does not converge to the least squares solution for inconsistent systems, but rather converges linearly to some convergence radius of the solution. To remedy this, Zouzias and Freris [30] proposed the REK algorithm to solve linear systems in all settings. For REK, row $i \in \{1, 2, ...m\}$ and column $j \in \{1, ...n\}$ of $\boldsymbol{X}$ are chosen at random with probability

$$\boldsymbol{P}(row = i) = \frac{\|\boldsymbol{X}^i\|_2^2}{\|\boldsymbol{X}\|_F^2}\ , \qquad \boldsymbol{P}(column = j) = \frac{\|\boldsymbol{X}_{(j)}\|_2^2}{\|\boldsymbol{X}\|_F^2},$$

and starting from $\boldsymbol{\beta}_0 = \boldsymbol{0}$ and $\boldsymbol{z}_0 = \boldsymbol{y}$, every iteration computes

$$\boldsymbol{\beta_t} := \boldsymbol{\beta_{t-1}} + \frac{(\boldsymbol{y}^i - \boldsymbol{z}_t^i - \boldsymbol{X}^i\boldsymbol{\beta_{t-1}})}{\|\boldsymbol{X}^i\|_2^2}(\boldsymbol{X}^i)^*, \quad \boldsymbol{z_t} := \boldsymbol{z}_{t-1} - \frac{\langle \boldsymbol{X}_{(j)}, \boldsymbol{z}_{t-1}\rangle}{\|\boldsymbol{X}_{(j)}\|_2^2}\boldsymbol{X}_{(j)}.$$

REK finds the optimal solution in all linear system settings. In the consistent setting, it behaves as RK. In the *overdetermined* inconsistent setting, $\boldsymbol{z}$ estimates the residual vector $\boldsymbol{r}$ and allows $\boldsymbol{\beta_t}$ to converge to the true least squares solution of the system. REK was shown to converge linearly in expectation to the least-squares solution by Zouzias and Freris [30].

2.3. Randomized Gauss–Seidel and its extension. The Gauss–Seidel method was originally published by Seidel but it was later discovered that Gauss had studied this method in a letter to his student [3]. Instead of relying on rows of a matrix, the Gauss–Seidel method relies on columns of $\boldsymbol{X}$. The randomized variant was studied by Leventhal and Lewis [16] shortly after RK was published. The randomized variant (RGS) requires a column j to be chosen randomly with probability $\frac{\|\boldsymbol{X}_{(j)}\|_2^2}{\|\boldsymbol{X}\|_F^2}$ and updates at every iteration

$$\boldsymbol{\beta_t} := \boldsymbol{\beta_{t-1}} + \frac{{\boldsymbol{X}_{(j)}}^*(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta_{t-1}})}{\|\boldsymbol{X}_{(j)}\|_2^2}\boldsymbol{e}_{(j)}, \tag{8}$$

where $\boldsymbol{e}_{(j)}$ is the j^{th} basis vector (a vector with 1 in the j^{th} position and 0 elsewhere). Leventhal and Lewis [16] showed that RGS converges linearly in expectation when $\boldsymbol{X}$ is overdetermined. However, it fails to find the least norm solution for an *underdetermined* linear system [19]. The REGS resolves this problem, much like REK did for RK in the case of *overdetermined* systems. The method chooses a random row and column of $\boldsymbol{X}$ exactly as in REK and then updates at every iteration

$$\begin{aligned}
\boldsymbol{\beta_t} &:= \boldsymbol{\beta_{t-1}} + \frac{{\boldsymbol{X}_{(j)}}^*(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta_{t-1}})}{\|\boldsymbol{X}_{(j)}\|_2^2}\boldsymbol{e}_{(j)}\ ,\\
\boldsymbol{P}_i &:= \boldsymbol{Id}_n - \frac{(\boldsymbol{X}^i)^*\boldsymbol{X}^i}{\|\boldsymbol{X}^i\|_2^2}\ ,\\
\boldsymbol{z}_t &:= \boldsymbol{P}_i(\boldsymbol{z}_{t-1} + \boldsymbol{\beta_t} - \boldsymbol{\beta_{t-1}}),
\end{aligned}$$

and at any fixed time t, outputs $\boldsymbol{\beta_t} - \boldsymbol{z}_t$ as the estimated solution to $\boldsymbol{X\beta} = \boldsymbol{y}$. Here, $\boldsymbol{Id}_n$ denotes the $n \times n$ identity matrix. This extension works for all variations of linear systems and was proved to converge linearly in expectation by Ma, Needell, and Ramdas [19].

The RK and RGS methods along with their extensions are extensively studied and compared in [19]. Table 1 summarizes the convergence properties of each of the randomized methods and their extensions.

In this paper, we focus on using combinations of RK and REK but also discuss RGS and REGS for comparison. We choose to focus on RK and REK because their

TABLE 1
Summary of convergence properties of randomized methods under all settings.

Method	Overdetermined, consistent: convergence to $\boldsymbol{\beta}_{\text{uniq}}$?	Overdetermined, inconsistent: convergence to $\boldsymbol{\beta}_{\boldsymbol{LS}}$?	Underdetermined: convergence to $\boldsymbol{\beta}_{\boldsymbol{LN}}$?
RK	Yes [25]	No [22]	Yes [19]
REK	Yes [30]	Yes [30]	Yes [19]
RGS	Yes [16]	Yes [16]	No [19]
REGS	Yes [19]	Yes [19]	Yes [19]

TABLE 2
Summary of notation for linear systems discussed and their solutions.

Linear system	Optimal solution
$\boldsymbol{X\beta} = \boldsymbol{y}$ (1)	$\boldsymbol{\beta}_\star$
$\boldsymbol{Ux} = \boldsymbol{y}$ (2)	$\boldsymbol{x}_\star$
$\boldsymbol{Vb} = \boldsymbol{x}$ (3)	$\boldsymbol{b}_\star$

TABLE 3
Summary of types of matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ for given m, n, and k relations. The first column indicates the setting where k is less than both m and n, the second column when k is between m and n, and the third when k is greater than both m and n. The cells in white indicate when our proposed methods will converge to the solution of the full system, and gray cells indicate when our methods will not. Recall that $\boldsymbol{X} \in \mathbb{C}^{m\times n}$, $\boldsymbol{U} \in \mathbb{C}^{m\times k}$ and $\boldsymbol{V} \in \mathbb{C}^{k\times n}$. We denote with "(In)con" systems that can be either consistent or inconsistent. Arguably, the $k < \min\{m, n\}$ setting is most practically relevant, and in this case our methods do indeed recover the optimal solution.

$\boldsymbol{X}$	$k < \min\{m, n\}$	$\min\{m, n\} < k < \max\{m, n\}$	$k > \max\{m, n\}$
Underdetermined	$\boldsymbol{U}$ = Over, Consis. $\boldsymbol{V}$ = Under (S1)	$\boldsymbol{U}$ = Under $\boldsymbol{V}$ = Under (S2)	$\boldsymbol{U}$ = Under $\boldsymbol{V}$ = Over, (In)con. (S2)
Overdetermined consis.	$\boldsymbol{U}$ = Over, Consis. $\boldsymbol{V}$ = Under (S1)	$\boldsymbol{U}$ = Over, Consis. $\boldsymbol{V}$ = Over, Consis. (S1)	$\boldsymbol{U}$ = Under $\boldsymbol{V}$ = Over, (In)con. (S2)
Overdetermined inonsis.	$\boldsymbol{U}$ = Over, Incon. $\boldsymbol{V}$ = Under (S3b)	$\boldsymbol{U}$ = Over, Incon. $\boldsymbol{V}$ = Over, (In)con. (S3a)	$\boldsymbol{U}$ = Under $\boldsymbol{V}$ = Over, (In)con. (S2)

updates consist only of scalar operations and inner products as opposed to REGS, which requires an outer product. The methods proposed are easily extendable to RGS and REGS.

3. Variations of factored linear systems.

Our proposed methods rely on interleaving solution estimates to subsystem (2) and subsystem (3). Because the convergence of RK, RGS, REK, and REGS are heavily dependent on the number of rows and columns in the linear system, it is important to discuss how the settings of (2) and (3) are determined by $\boldsymbol{X}$. In this section, we will discuss when we can expect our methods to solve the full system.

For simplicity in notation, we will denote $\boldsymbol{\beta}_\star$, $\boldsymbol{x}_\star$, and $\boldsymbol{b}_\star$ as the "optimal" solution of (1), (2), and (3), respectively, as summarized in Table 2. By "optimal" solution for (2) and (3), we mean the unique, least norm, or the least squares solution, depending on the type of system (overdetermined consistent, underdetermined, overdetermined inconsistent). Since we assume that $\boldsymbol{UV}$ may be low-rank, $\boldsymbol{\beta}_\star$ is going to be the least norm solution as described in (7). Table 3 presents such a summary depending on the size of k with respect to m and n.

We spend the rest of this section justifying the shading in Table 3. For this, we split Table 3 into three scenarios: (S1) $\boldsymbol{U}$ is overdetermined and consistent, (S2) $\boldsymbol{U}$ is underdetermined, and (S3a and S3b) $\boldsymbol{X}$ is overdetermined and inconsistent. It should be noted that in Scenarios S1 and S2, the overdeterminedness or underdeterminedness of each subsystem follows immediately from sizes of m, n, and k and the assumption that the subsystems are full rank. We use the over/underdeterminedness of each subsystem to show $\boldsymbol{b}_\star = \boldsymbol{\beta}_\star$ for Scenario S1 and $\boldsymbol{b}_\star \neq \boldsymbol{\beta}_\star$ for Scenario S2. In Scenario S3, a little more work needs to be done to conclude the consistency of each subsystem. For Scenario S3a and S3b, we first investigate how inconsistency in (1) affects the consistency of (2) and (3) and then show that for Scenario S3a, $\boldsymbol{b}_\star \neq \boldsymbol{\beta}_\star$, and for Scenario S3b, $\boldsymbol{b}_\star = \boldsymbol{\beta}_\star$. This section provides the intuition on when we should expect our methods (or similar ones based on interleaving solutions to the subsystems) to work. However, one may also skip ahead to the next section, where formally we present our algorithm and main results.

- **Scenario S1: U overdetermined, consistent.** When $\boldsymbol{U}$ is overdetermined and consistent, we find that solving (2) and (3) gives us the optimal solution of (1). Indeed, in the case where $\boldsymbol{V}$ is overdetermined and consistent, we have

$$\begin{aligned}\boldsymbol{b}_\star &= (\boldsymbol{V}^*\boldsymbol{V})^{-1}\boldsymbol{V}^*(\boldsymbol{U}^*\boldsymbol{U})^{-1}\boldsymbol{U}^*\boldsymbol{y}\\ &= (\boldsymbol{V}^*\boldsymbol{V})^{-1}\boldsymbol{V}^*(\boldsymbol{U}^*\boldsymbol{U})^{-1}\boldsymbol{U}^*\boldsymbol{U}\boldsymbol{V}\boldsymbol{\beta}_\star\\ &= (\boldsymbol{V}^*\boldsymbol{V})^{-1}\boldsymbol{V}^*\boldsymbol{V}\boldsymbol{\beta}_\star\\ &= \boldsymbol{\beta}_\star.\end{aligned}$$

  In the case where $\boldsymbol{V}$ is underdetermined, we have

$$\begin{aligned}\boldsymbol{V}\boldsymbol{b}_\star &= \boldsymbol{V}\boldsymbol{V}^*(\boldsymbol{V}\boldsymbol{V}^*)^{-1}(\boldsymbol{U}^*\boldsymbol{U})^{-1}\boldsymbol{U}^*\boldsymbol{y}\\ &= \boldsymbol{V}\boldsymbol{V}^*(\boldsymbol{V}\boldsymbol{V}^*)^{-1}(\boldsymbol{U}^*\boldsymbol{U})^{-1}\boldsymbol{U}^*\boldsymbol{U}\boldsymbol{V}\boldsymbol{\beta}_\star\\ &= \boldsymbol{V}\boldsymbol{\beta}_\star.\end{aligned}$$

  Since $\boldsymbol{X}$ is possibly low-rank, we still need to argue that this implies that $\boldsymbol{b}_\star = \boldsymbol{\beta}_\star$, i.e., $\boldsymbol{b}_\star$ is indeed the least norm solution to the full system. Suppose toward a contradiction that $\boldsymbol{\beta}_\star$ is the least norm solution of the full system but not subsystem (3); in other words assume that $\boldsymbol{\beta}_\star = \boldsymbol{b}_\star + \boldsymbol{b}$, where $\boldsymbol{V}\boldsymbol{b} = \boldsymbol{0}$ and $\boldsymbol{b}$ is nontrivial. Multiplying both sides by $\boldsymbol{X}$, we see that since $\boldsymbol{\beta}_\star$ has a nontrivial component $\boldsymbol{b}$ such that $\boldsymbol{X}\boldsymbol{b} = \boldsymbol{0}$, it cannot be the least norm solution to the full system as assumed, reaching a contradiction. Therefore, in the consistent case when $\boldsymbol{U}$ is overdetermined, we have that $\boldsymbol{b}_\star = \boldsymbol{\beta}_\star$ and may hope that our proposed methods will be able to solve the full system (1) utilizing the subsystems (2) and (3).
- **Scenario S2: U underdetermined.** When $\boldsymbol{U}$ is underdetermined, solving (2) and (3) for their optimal solutions does not guarantee the optimal solution of the full system. Intuitively, (2) has infinitely many solutions and $\boldsymbol{x}_\star = \boldsymbol{x}_{LN} \neq \boldsymbol{V}\boldsymbol{\beta}_\star$. Mathematically, investigating $\boldsymbol{b}_\star$, we find that

$$\begin{aligned}\boldsymbol{V}\boldsymbol{b}_\star &= \boldsymbol{V}\boldsymbol{V}^*(\boldsymbol{V}\boldsymbol{V}^*)^{-1}\boldsymbol{U}^*(\boldsymbol{U}\boldsymbol{U}^*)^{-1}\boldsymbol{y}\\ &= \boldsymbol{V}\boldsymbol{V}^*(\boldsymbol{V}\boldsymbol{V}^*)^{-1}\boldsymbol{U}^*(\boldsymbol{U}\boldsymbol{U}^*)^{-1}\boldsymbol{U}\boldsymbol{V}\boldsymbol{\beta}_\star\\ &= \boldsymbol{U}^*(\boldsymbol{U}\boldsymbol{U}^*)^{-1}\boldsymbol{U}\boldsymbol{V}\boldsymbol{\beta}_\star\\ &\neq \boldsymbol{V}\boldsymbol{\beta}_\star \text{ if } \boldsymbol{V} \text{ underdetermined}\\ \boldsymbol{b}_\star &= (\boldsymbol{V}^*\boldsymbol{V})^{-1}\boldsymbol{V}^*(\boldsymbol{U}^*(\boldsymbol{U}\boldsymbol{U}^*)^{-1}\boldsymbol{y} - \boldsymbol{r}_V)\\ &= (\boldsymbol{V}^*\boldsymbol{V})^{-1}\boldsymbol{V}^*\boldsymbol{U}^*(\boldsymbol{U}\boldsymbol{U}^*)^{-1}\boldsymbol{U}\boldsymbol{V}\boldsymbol{\beta}_\star\\ &\neq \boldsymbol{\beta}_\star \text{ if } \boldsymbol{V} \text{ overdetermined, inconsistent,}\end{aligned}$$

where we rewrite $\boldsymbol{x}_\star = \boldsymbol{V}\boldsymbol{b}_\star + \boldsymbol{r}_V$ for $\boldsymbol{r}_V \in null(\boldsymbol{V}^*)$ since subsystem (3) may be inconsistent. Note that if subsystem (3) is consistent, then we simply have $\boldsymbol{r}_V = 0$ and the above calculation still carries through. *Therefore, we do not expect our proposed methods to succeed when* $\boldsymbol{U}$ *is underdetermined.* Fortunately, this case seems to be of little practical interest, since factoring an underdetermined system does not typically save any computation.

- **Scenario S3: X inconsistent.** Before we discuss whether it's possible to recover the optimal solution to the full system, we must first discuss what $\boldsymbol{X}$ being inconsistent implies about the subsystems (2) and (3). In particular, one needs to determine whether inconsistency in the full system creates inconsistencies in the individual subsystems. If $\boldsymbol{X}$ is inconsistent, then we have $\boldsymbol{X}\boldsymbol{\beta}_\star + \boldsymbol{r} = \boldsymbol{y}$, where $\boldsymbol{\beta}_\star$ is the optimal solution of (1) and $\boldsymbol{X}^*\boldsymbol{r} = \boldsymbol{0}$. Now, consider decomposing $\boldsymbol{r} = \boldsymbol{r}_1 + \boldsymbol{r}_2$, where $\boldsymbol{U}^*\boldsymbol{r}_1 = \boldsymbol{0}$, $\boldsymbol{U}^*\boldsymbol{r}_2 \neq \boldsymbol{0}$, and $\boldsymbol{V}^*\boldsymbol{U}^*\boldsymbol{r}_2 = 0$. Notice that $\boldsymbol{X}^*\boldsymbol{r} = \boldsymbol{V}^*\boldsymbol{U}^*(\boldsymbol{r}_1+\boldsymbol{r}_2) = \boldsymbol{V}^*\boldsymbol{U}^*\boldsymbol{r}_1 + \boldsymbol{V}^*\boldsymbol{U}^*\boldsymbol{r}_2 = \boldsymbol{0}$, as desired. We want to decompose the full system $\boldsymbol{X}\boldsymbol{\beta}_\star + \boldsymbol{r} = \boldsymbol{y}$ into two subsystems. Following a similar thought process as before, we choose to decompose our full system into the following:

$$\boldsymbol{U}\boldsymbol{x} + \boldsymbol{r}_1 + \boldsymbol{r}_2 = \boldsymbol{y}, \tag{9}$$

$$\boldsymbol{V}\boldsymbol{b} = \boldsymbol{x}. \tag{10}$$

Clearly, (9) is inconsistent since $\boldsymbol{U}^*\boldsymbol{r}_1 = \boldsymbol{0}$ and $\boldsymbol{U}^*\boldsymbol{r}_2 \neq \boldsymbol{0}$. Because $\boldsymbol{U}$ must be overdetermined for $\boldsymbol{X}$ to be inconsistent, $\boldsymbol{x}_\star = (\boldsymbol{U}^*\boldsymbol{U})^{-1}\boldsymbol{U}^*(\boldsymbol{y}-\boldsymbol{r}_1-\boldsymbol{r}_2) = (\boldsymbol{U}^*\boldsymbol{U})^{-1}\boldsymbol{U}^*(\boldsymbol{y}-\boldsymbol{r}_2)$ is the least squares solution to (9).

  - **Case S3a: $\boldsymbol{V}$ overdetermined.** Note that the second subsystem (10) is possibly inconsistent (since there may be a component $\boldsymbol{x}_\star$ of in the null space of $\boldsymbol{V}^*$). Writing $\boldsymbol{x}_\star = \boldsymbol{V}\boldsymbol{b}_\star + \boldsymbol{r}_V$ such that $\boldsymbol{r}_V \in null(\boldsymbol{V}^*)$, we have

$$\begin{aligned}\boldsymbol{b}_\star &= (\boldsymbol{V}^*\boldsymbol{V})^{-1}\boldsymbol{V}^*\left((\boldsymbol{U}^*\boldsymbol{U})^{-1}\boldsymbol{U}^*(\boldsymbol{y}-\boldsymbol{r}_1-\boldsymbol{r}_2) - \boldsymbol{r}_V\right)\\ &= (\boldsymbol{V}^*\boldsymbol{V})^{-1}\boldsymbol{V}^*(\boldsymbol{U}^*\boldsymbol{U})^{-1}\boldsymbol{U}^*(\boldsymbol{y}-\boldsymbol{r}_2)\\ &= \boldsymbol{\beta}_\star - (\boldsymbol{V}^*\boldsymbol{V})^{-1}\boldsymbol{V}^*(\boldsymbol{U}^*\boldsymbol{U})^{-1}\boldsymbol{U}^*\boldsymbol{r}_2\\ &\neq \boldsymbol{\beta}_\star.\end{aligned}$$

    Similar to Scenario S2, if subsystem (10) is indeed consistent, then $\boldsymbol{r}_V = 0$ and the above calculation still carries through. *Therefore, in this case, we do not expect to find the optimal solution to* (1).

  - **Case S3b: $\boldsymbol{V}$ underdetermined.** In this case, $\boldsymbol{r}_2 = \boldsymbol{0}$ and solving (9) and (10) obtains the optimal solution to the full system since

$$\begin{aligned}\boldsymbol{V}\boldsymbol{b}_\star &= \boldsymbol{V}\boldsymbol{V}^*(\boldsymbol{V}\boldsymbol{V}^*)^{-1}(\boldsymbol{U}^*\boldsymbol{U})^{-1}\boldsymbol{U}^*(\boldsymbol{y}-\boldsymbol{r}_1)\\ &= \boldsymbol{V}\boldsymbol{V}^*(\boldsymbol{V}\boldsymbol{V}^*)^{-1}(\boldsymbol{U}^*\boldsymbol{U})^{-1}\boldsymbol{U}^*\boldsymbol{y}\\ &= \boldsymbol{V}\boldsymbol{V}^*(\boldsymbol{V}\boldsymbol{V}^*)^{-1}(\boldsymbol{U}^*\boldsymbol{U})^{-1}\boldsymbol{U}^*\boldsymbol{U}\boldsymbol{V}\boldsymbol{\beta}_\star\\ &= \boldsymbol{V}\boldsymbol{V}^*(\boldsymbol{V}\boldsymbol{V}^*)^{-1}\boldsymbol{V}\boldsymbol{\beta}_\star\\ &= \boldsymbol{V}\boldsymbol{\beta}_\star.\end{aligned}$$

    Following the same argument as in Scenario S1 when $\boldsymbol{V}$ is underdetermined, we reach the conclusion that $\boldsymbol{b}_\star = \boldsymbol{\beta}_\star$. *Thus, in this case our methods have the potential to solve the full system.*

These three scenarios fully explain the shading in Table 3. The focus of the remainder of this paper will be the case in which $k < m, n$ (i.e., left column of Table 3) since, as mentioned, it is practically the most relevant setting.

4. Methods and main results.

Our approach intertwines two iterative methods to solve subsystem (2) followed by subsystem (3). For the consistent setting, we propose Algorithm 1, which uses an iterate of RK on (2) intertwined with an iterate of RK to solve (3). For the inconsistent setting, we propose using REK to solve subsystem (9) followed by RK to solve subsystem (10) as shown in Algorithm 2. We view the latter method as a more practical approach and the former as interesting from a theoretical point of view. Recall that $\boldsymbol{x}_t^p$ is the pth element in the vector $\boldsymbol{x}_t$. Standard stopping criteria include terminating when the difference in the iterates is small or when the residual is less than a predetermined tolerance. To avoid adding complexity to the algorithm, the residual should be computed and checked approximately every m iterations. We propose an approach that interlaces solving subsystems (2) and (3); this has a couple of advantages over solving each subsystem separately. First, if we are given some tolerance ϵ that we allow on the full system, it is unclear when we should stop the iterates of the first subsystem to obtain such an error—if solving the first subsystem is terminated prematurely, the error may propagate through iterates when solving the second subsystem. Second, the interlacing allows for opportunities to implement these algorithms in parallel. We leave the specifics of such an implementation as future work as it is outside the scope of this paper.

Algorithm 1 RK-RK.

```
Input: $\boldsymbol{U}$, $\boldsymbol{V}$, $\boldsymbol{y}$
while stopping criteria not reached do
    Choose row $\boldsymbol{U}^i$ with probability $\frac{\|\boldsymbol{U}^i\|_2^2}{\|\boldsymbol{U}\|_F^2}$
    Update $\boldsymbol{x}_t := \boldsymbol{x}_{t-1} + \frac{(\boldsymbol{y}^i - \boldsymbol{U}^i \boldsymbol{x}_{t-1})}{\|\boldsymbol{U}^i\|_2^2}(\boldsymbol{U}^i)^*$
    Choose row $\boldsymbol{V}^p$ with probability $\frac{\|\boldsymbol{V}^p\|_2^2}{\|\boldsymbol{V}\|_F^2}$
    Update $\boldsymbol{b}_t := \boldsymbol{b}_{t-1} + \frac{(\boldsymbol{x}_t^p - \boldsymbol{V}^p \boldsymbol{b}_{t-1})}{\|\boldsymbol{V}^p\|_2^2}(\boldsymbol{V}^p)^*$
end while
```

Algorithm 2 REK-RK.

```
Input: $\boldsymbol{U}$, $\boldsymbol{V}$, $\boldsymbol{y}$
while stopping criteria not reached do
    Choose row $\boldsymbol{U}^i$ with probability $\frac{\|\boldsymbol{U}^i\|_2^2}{\|\boldsymbol{U}\|_F^2}$
    Choose column $\boldsymbol{U}_{(j)}$ with probability $\frac{\|\boldsymbol{U}_{(j)}\|_2^2}{\|\boldsymbol{U}\|_F^2}$
    Update $\boldsymbol{z}_t := \boldsymbol{z}_{t-1} - \frac{{\boldsymbol{U}_{(j)}}^* \boldsymbol{z}_{t-1}}{\|\boldsymbol{U}_{(j)}\|_2^2}\boldsymbol{U}_{(j)}$
    Update $\boldsymbol{x}_t := \boldsymbol{x}_{t-1} + \frac{(\boldsymbol{y}^i - \boldsymbol{z}_t^i + \boldsymbol{U}^i \boldsymbol{x}_{t-1})}{\|\boldsymbol{U}^i\|_2^2}(\boldsymbol{U}^i)^*$
    Choose row $\boldsymbol{V}^p$ with probability $\frac{\|\boldsymbol{V}^p\|_2^2}{\|\boldsymbol{V}\|_F^2}$
    Update $\boldsymbol{b}_t := \boldsymbol{b}_{t-1} + \frac{(\boldsymbol{x}_t^p - \boldsymbol{V}^p \boldsymbol{b}_{t-1})}{\|\boldsymbol{V}^p\|_2^2}(\boldsymbol{V}^p)^*$
end while
```

4.1. Main result.

Our main result shows that Algorithms 1 and 2 converge linearly to the desired solution. The convergence rate, as expected, is a function of the conditioning of the subsystems, and hence we introduce the following notation. Here and throughout, for any matrix $\boldsymbol{A}$ we write

$$\alpha_A := 1 - \frac{\sigma_{\min}^2(\boldsymbol{A})}{\|\boldsymbol{A}\|_F^2}\ , \tag{11}$$

$$\kappa_A^2 := \frac{\sigma_{\max}^2(\boldsymbol{A})}{\sigma_{\min}^2(\boldsymbol{A})}\ , \tag{12}$$

$$\theta_A := \frac{1}{\sigma_{\min}^2(\boldsymbol{A})}\ , \tag{13}$$

where $\sigma_{\min}^2(\boldsymbol{A})$ is the smallest nonzero singular value of $\boldsymbol{A}$, and κ_A^2 is the squared condition number of $\boldsymbol{A}$. Recall that the *optimal solution* to a system is either the least-norm, unique, or least-squares solution depending on whether the system is underdetermined, overdetermined consistent, or overdetermined inconsistent, respectively.

THEOREM 4.1. *Let $\boldsymbol{X}$ be low rank, $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{V}$ such that $\boldsymbol{U} \in \mathbb{C}^{m\times k}$ and $\boldsymbol{V} \in \mathbb{C}^{k\times n}$ are full rank, and the systems $\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{y}$, $\boldsymbol{U}\boldsymbol{x} = \boldsymbol{y}$, and $\boldsymbol{V}\boldsymbol{b} = \boldsymbol{x}$ have optimal solutions $\boldsymbol{\beta}_\star$, $\boldsymbol{x}_\star$ and $\boldsymbol{b}_\star$ respectively, and α_U, $\alpha_V, \kappa_U^2, \theta_V$ are as defined in* (11), (12), (13). *Setting $\boldsymbol{b}_0 = \boldsymbol{0}$ and assuming $k < m, n$, we have*

(a) *if $\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{y}$ is consistent, then $\boldsymbol{b}_\star = \boldsymbol{\beta}_\star$ and Algorithm* 1 *converges with expected error*

$$\mathbb{E}\|\boldsymbol{b}_t - \boldsymbol{\beta}_\star\|^2 \le \alpha_V^t\|\boldsymbol{b}_\star\|^2 + \theta_V\alpha_U^t\|\boldsymbol{x}_\star\|^2,$$

(b) *if $\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{y}$ is inconsistent, then $\boldsymbol{b}_\star = \boldsymbol{\beta}_\star$ and Algorithm* 2 *converges with expected error*

$$\mathbb{E}\|\boldsymbol{b}_t - \boldsymbol{\beta}_\star\|^2 \le \alpha_V^t\|\boldsymbol{b}_\star\|^2 + \theta_V\alpha_U^{\lfloor t/2\rfloor}(1 + 2\kappa_U^2)\|\boldsymbol{x}_\star\|^2.$$

Remarks.

1. Theorem 4.1(a) also applies to the setting in which $\boldsymbol{X}$ is overdetermined, consistent, and $n < k < m$. In the proof of Lemma 4.4, one must simply note that the bound (14) still holds for this setting and all other steps in the proof analogously follow.

2. The individual subsystems (2) and (3) are better conditioned than the full subsystem (1). It can easily be verified that $\alpha_V, \alpha_U \le \alpha_X$. Empirically, our experiments in the next section suggest that $\boldsymbol{U}$ and $\boldsymbol{V}$ can be substantially better conditioned than $\boldsymbol{X}$.

3. Algorithm 1 is interesting to discuss from a theoretical standpoint but in applications Algorithm 2 is more practical as linear systems are typically inconsistent. Algorithm 1 can be utilized in applications if error in the solution is tolerable. In particular, if $\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{y}$ is inconsistent, then Algorithm 1 will converge in expectation to some convergence horizon. This can be see by replacing the use of Proposition 4.2 in the bound (15) with the convergence bound of RK on inconsistent linear systems found in Theorem 2.1 of [22].

4. While not the main focus of this paper, we briefly note here that for matrices large enough that they cannot be stored entirely in memory, there is an additional cost that must be paid in terms of moving data between the disk and RAM. In a

truly large-scale implementation, the RK-RK algorithm might be more scalable than the REK-RK algorithm since RK only accesses random rows of both $\boldsymbol{U}$ and $\boldsymbol{V}$, which is efficient if both matrices are stored in row major form, but REK accesses both random rows and columns, and hence storing in either row major or column major format will be slow for one of the two operations.

4.2. Supporting results. To prepare for the proof of the above theorem (the central theoretical result of the paper), we state a few supporting results which will help simplify the presentation of the proof. We begin by stating known results on the convergence of RK and REK on linear systems. Let $\mathbb{E}_b$ denote the expected value taken over the choice of rows in $\boldsymbol{V}$ and $\mathbb{E}_x$ the expected value taken over the choice of rows in $\boldsymbol{U}$ and when necessary the choice of columns in $\boldsymbol{U}$. Also, let $\mathbb{E}$ denote the full expected value (over all random variables and iterations) and $\mathbb{E}^{t-1}$ be the expectation conditional on the first $t-1$ iterations.

Proposition 4.2 (see [24, Theorem 2]). *Given a consistent linear system $\boldsymbol{X\beta} = \boldsymbol{y}$, the RK algorithm, with initialization $\boldsymbol{\beta}_0 = \boldsymbol{0}$, as described in section* 2.2 *converges to the optimal solution $\boldsymbol{\beta}_\star$ with expected error*

$$\mathbb{E}\|\boldsymbol{\beta}_t - \boldsymbol{\beta}_\star\|^2 \leq \alpha_X^t \|\boldsymbol{\beta}_\star\|^2,$$

where α_X is as defined in (11).

Proposition 4.3 (see [30, Theorem 8]). *Given a linear system $\boldsymbol{X\beta} = \boldsymbol{y}$, the REK algorithm, with initialization $\boldsymbol{\beta}_0 = \boldsymbol{0}$, as described in section* 2.2 *converges to the optimal solution $\boldsymbol{\beta}_\star$ with expected error*

$$\mathbb{E}\|\boldsymbol{\beta}_t - \boldsymbol{\beta}_\star\|^2 \leq \alpha_X^{\lfloor t/2 \rfloor}(1 + 2\kappa_X^2)\|\boldsymbol{\beta}_\star\|^2,$$

where α_X is as defined in (11) *and κ_X^2 is as defined in* (12).

The proof of Theorem 4.1 builds directly on two useful lemmas. Lemma 4.4 addresses the impact of intertwining the algorithms. In particular, it shows useful relationships involving $\tilde{\boldsymbol{b}}_t$, the RK update solving the linear system $\boldsymbol{Vb} = \boldsymbol{x}_\star$ at the t^{th} iteration (with $\boldsymbol{b}_{t-1}$ as the previous estimate), and our update $\boldsymbol{b}_t$. Lemma 4.5 states that conditional on the first $t-1$ iterations, we can split the norm squared error $\|\boldsymbol{b}_t - \boldsymbol{b}_\star\|^2$ into two terms relating to the error from solving subsystem (2) and the error from solving subsystem (3). To complete the proof of Theorem 4.1, we bound the error from solving (2) depending on whether we use RK (as in Algorithm 1) or REK (as in Algorithm 2), and then apply the law of iterated expectations to bound the error from solving (3). We now state the aforementioned lemmas and then formally prove the theorem.

Lemma 4.4. *Let $\tilde{\boldsymbol{b}}_t = \boldsymbol{b}_{t-1} + \frac{\boldsymbol{x}_\star^p - \boldsymbol{V}^p \boldsymbol{b}_{t-1}}{\|\boldsymbol{V}^p\|^2}(\boldsymbol{V}^p)^*$. In Algorithms* 1 *and* 2 *we have that*

(a) $\mathbb{E}_b^{t-1}\left\langle \boldsymbol{b}_t - \tilde{\boldsymbol{b}}_t, \tilde{\boldsymbol{b}}_t - \boldsymbol{b}_\star \right\rangle = 0$,

(b) $\|\tilde{\boldsymbol{b}}_t - \boldsymbol{b}_\star\|^2 = \|\boldsymbol{b}_{t-1} - \boldsymbol{b}_\star\|^2 - \|\tilde{\boldsymbol{b}}_t - \boldsymbol{b}_{t-1}\|^2$.

In other words, part (a) states that the difference between an RK iterate solving the exact linear system $\boldsymbol{Vb} = \boldsymbol{x}_\star$ and our RK iterate (which solves the linear system resulting from intertwining $\boldsymbol{V}\boldsymbol{b} = \boldsymbol{x}_t$) is orthogonal to $\tilde{\boldsymbol{b}}_t - \boldsymbol{b}_\star$. This will come in handy in Lemma 4.5. Part (b) is a Pythagoras-style statement, which follows from well-known orthogonality properties of RK updates, included here for simplicity and completness.

Proof. To prove statement (b), we note that $(\tilde{\boldsymbol{b}}_t - \boldsymbol{b}_{t-1})$ is parallel to $\boldsymbol{V}^p$ and $(\tilde{\boldsymbol{b}}_t - \boldsymbol{b}_\star)$ is perpendicular to $\boldsymbol{V}^p$ since $\boldsymbol{V}^p(\tilde{\boldsymbol{b}}_t - \boldsymbol{b}_\star) = \boldsymbol{V}^p(\boldsymbol{b}_{t-1} + \frac{\boldsymbol{x}_\star^p - \boldsymbol{V}^p\boldsymbol{b}_{t-1}}{\|\boldsymbol{V}^p\|^2}(\boldsymbol{V}^p)^* - \boldsymbol{b}_\star) = \boldsymbol{V}^p\boldsymbol{b}_{t-1} + \boldsymbol{x}_\star^p - \boldsymbol{V}^p\boldsymbol{b}_{t-1} - \boldsymbol{x}_\star^p = \boldsymbol{0}$. We apply the Pythagorean theorem to obtain the desired result.

We prove statement (a) by direct substitution and expansion, as follows:

$$
\begin{aligned}
\mathbb{E}_b^{t-1}\left\langle \boldsymbol{b}_t - \tilde{\boldsymbol{b}}_t, \tilde{\boldsymbol{b}}_t - \boldsymbol{b}_\star \right\rangle &= \mathbb{E}_b^{t-1}\left\langle \frac{\boldsymbol{x}_t^p - \boldsymbol{x}_\star^p}{\|\boldsymbol{V}^p\|^2}(\boldsymbol{V}^p)^*, \boldsymbol{b}_{t-1} - \boldsymbol{b}_\star + \frac{\boldsymbol{x}_\star^p - \boldsymbol{V}^p\boldsymbol{b}_{t-1}}{\|\boldsymbol{V}^p\|^2}(\boldsymbol{V}^p)^* \right\rangle \\
&\overset{(i)}{=} \mathbb{E}_b^{t-1}\left\langle \frac{\boldsymbol{x}_t^p - \boldsymbol{x}_\star^p}{\|\boldsymbol{V}^p\|^2}(\boldsymbol{V}^p)^*, \frac{\boldsymbol{x}_\star^p - \boldsymbol{V}^p\boldsymbol{b}_{t-1}}{\|\boldsymbol{V}^p\|^2}(\boldsymbol{V}^p)^* \right\rangle \\
&\qquad + \mathbb{E}_b^{t-1}\left\langle \frac{\boldsymbol{x}_t^p - \boldsymbol{x}_\star^p}{\|\boldsymbol{V}^p\|^2}(\boldsymbol{V}^p)^*, \boldsymbol{b}_{t-1} - \boldsymbol{b}_\star \right\rangle \\
&\overset{(ii)}{=} \mathbb{E}_b^{t-1}\frac{(\boldsymbol{x}_t^p - \boldsymbol{x}_\star^p)(\boldsymbol{x}_\star^p - \boldsymbol{V}^p\boldsymbol{b}_{t-1})}{\|\boldsymbol{V}^p\|^2} + \left\langle \mathbb{E}_b^{t-1}\frac{\boldsymbol{x}_t^p - \boldsymbol{x}_\star^p}{\|\boldsymbol{V}^p\|^2}(\boldsymbol{V}^p)^*, \boldsymbol{b}_{t-1} - \boldsymbol{b}_\star \right\rangle \\
&\overset{(iii)}{=} \sum_p \frac{(\boldsymbol{x}_t^p - \boldsymbol{x}_\star^p)(\boldsymbol{x}_\star^p - \boldsymbol{V}^p\boldsymbol{b}_{t-1})}{\|\boldsymbol{V}^p\|^2}\frac{\|\boldsymbol{V}^p\|^2}{\|\boldsymbol{V}\|_F^2} \\
&\qquad + \left\langle \sum_p \frac{(\boldsymbol{x}_t^p - \boldsymbol{x}_\star^p)(\boldsymbol{V}^p)^*}{\|\boldsymbol{V}^p\|^2}\frac{\|\boldsymbol{V}^p\|^2}{\|\boldsymbol{V}\|_F^2}, \boldsymbol{b}_{t-1} - \boldsymbol{b}_\star \right\rangle \\
&= \frac{(\boldsymbol{x}_t - \boldsymbol{x}_\star)^*(\boldsymbol{x}_\star - \boldsymbol{V}\boldsymbol{b}_{t-1})}{\|\boldsymbol{V}\|_F^2} + \left\langle \frac{\boldsymbol{V}^*(\boldsymbol{x}_t - \boldsymbol{x}_\star)}{\|\boldsymbol{V}\|_F^2}, \boldsymbol{b}_{t-1} - \boldsymbol{b}_\star \right\rangle \\
&\overset{(iv)}{=} \left\langle \frac{\boldsymbol{x}_t - \boldsymbol{x}_\star}{\|\boldsymbol{V}\|_F^2}, \boldsymbol{V}(\boldsymbol{b}_\star - \boldsymbol{b}_{t-1}) \right\rangle + \left\langle \frac{\boldsymbol{x}_t - \boldsymbol{x}_\star}{\|\boldsymbol{V}\|_F^2}, \boldsymbol{V}(\boldsymbol{b}_{t-1} - \boldsymbol{b}_\star) \right\rangle \qquad (14) \\
&= 0.
\end{aligned}
$$

Step (i) follows from linearity of inner products, step (ii) simplifies the inner product of two parallel vectors, and step (iii) computes the expectation over all possible choices of rows of $\boldsymbol{V}$. In step (iv), we use the fact that for $k < m, n$, subsystem (3) is always consistent (since $\boldsymbol{V}$ is underdetermined) to make the substitution $\boldsymbol{x}_\star = \boldsymbol{V}\boldsymbol{b}_\star$. ☐

LEMMA 4.5. *In Algorithms* 1 *and* 2, *we can bound the expected norm squared error of* $\boldsymbol{b}_t - \boldsymbol{b}_\star$ *as*

$$\mathbb{E}^{t-1}\|\boldsymbol{b}_t - \boldsymbol{b}_\star\|^2 \le \alpha_V\|\boldsymbol{b}_{t-1} - \boldsymbol{b}_\star\|^2 + \mathbb{E}_x^{t-1}\frac{\|\boldsymbol{x}_t - \boldsymbol{x}_\star\|^2}{\|\boldsymbol{V}\|_F^2}.$$

We investigate the expectation of the norm squared error of $\boldsymbol{b}_t - \boldsymbol{b}_\star$ conditional on the first $t-1$ iterations and over the choice of rows of $\boldsymbol{V}$. We keep $\mathbb{E}_x^{t-1}$ in our bound as this expectation will depend on whether Algorithms 1 or 2 is being used.

Proof.

$$\begin{aligned}
\mathbb{E}^{t-1}\|\boldsymbol{b}_t - \boldsymbol{b}_\star\|^2 &= \mathbb{E}^{t-1}\|\boldsymbol{b}_t - \boldsymbol{b}_\star + \tilde{\boldsymbol{b}}_t - \tilde{\boldsymbol{b}}_t\|^2 \\
&= \mathbb{E}^{t-1}\|\tilde{\boldsymbol{b}}_t - \boldsymbol{b}_\star\|^2 + \mathbb{E}^{t-1}\|\boldsymbol{b}_t - \tilde{\boldsymbol{b}}_t\|^2 + 2\mathbb{E}^{t-1}\left\langle \tilde{\boldsymbol{b}}_t - \boldsymbol{b}_\star, \boldsymbol{b}_t - \tilde{\boldsymbol{b}}_t \right\rangle \\
&\overset{(iii)}{=} \mathbb{E}^{t-1}\|\tilde{\boldsymbol{b}}_t - \boldsymbol{b}_\star\|^2 + \mathbb{E}^{t-1}\|\boldsymbol{b}_t - \tilde{\boldsymbol{b}}_t\|^2 \\
&\overset{(iv)}{=} \mathbb{E}^{t-1}\|\boldsymbol{b}_{t-1} - \boldsymbol{b}_\star\|^2 - \mathbb{E}^{t-1}\|\tilde{\boldsymbol{b}}_t - \boldsymbol{b}_{t-1}\|^2 + \mathbb{E}^{t-1}\|\boldsymbol{b}_t - \tilde{\boldsymbol{b}}_t\|^2 \\
&\overset{(v)}{=} \|\boldsymbol{b}_{t-1} - \boldsymbol{b}_\star\|^2 - \mathbb{E}^{t-1}\left\|\frac{(\boldsymbol{x}_\star^p - \boldsymbol{V}^p\boldsymbol{b}_{t-1})}{\|\boldsymbol{V}^p\|^2}(\boldsymbol{V}^p)^*\right\|^2 \\
&\quad + \mathbb{E}^{t-1}\left\|\frac{(\boldsymbol{x}_t^p - \boldsymbol{x}_\star^p)}{\|\boldsymbol{V}^p\|^2}(\boldsymbol{V}^p)^*\right\|^2 \\
&= \|\boldsymbol{b}_{t-1} - \boldsymbol{b}_\star\|^2 - \mathbb{E}^{t-1}\left[\frac{|\boldsymbol{V}^p\boldsymbol{b}_\star - \boldsymbol{V}^p\boldsymbol{b}_{t-1}|^2}{\|\boldsymbol{V}^p\|^2}\right] + \mathbb{E}^{t-1}\left[\frac{|\boldsymbol{x}_t^p - \boldsymbol{x}_\star^p|^2}{\|\boldsymbol{V}^p\|^2}\right].
\end{aligned}$$

Steps (iii) and (iv) are applications of Lemmas 4.4(a) and 4.4(b), respectively, and step (v) follows from the definition of each term and simplification using the fact that $\boldsymbol{V}\boldsymbol{b}_\star = \boldsymbol{x}_\star$.

Now, we evaluate the conditional expectation on the choices of rows of $\boldsymbol{V}$ to complete the proof:

$$\begin{aligned}
\mathbb{E}^{t-1}\|\boldsymbol{b}_t - \boldsymbol{b}_\star\|^2 &\overset{(vi)}{=} \|\boldsymbol{b}_{t-1} - \boldsymbol{b}_\star\|^2 - \mathbb{E}_b^{t-1}\left[\frac{|\boldsymbol{V}^p\boldsymbol{b}_\star - \boldsymbol{V}^p\boldsymbol{b}_{t-1}|^2}{\|\boldsymbol{V}^p\|^2}\right] \\
&\quad + \mathbb{E}_x^{t-1}\mathbb{E}_b^{t-1}\left[\frac{|\boldsymbol{x}_t^p - \boldsymbol{x}_\star^p|^2}{\|\boldsymbol{V}^p\|^2}\right] \\
&= \|\boldsymbol{b}_{t-1} - \boldsymbol{x}_\star\|^2 - \sum_{p=1}^k \frac{|\boldsymbol{V}^p\boldsymbol{b}_\star - \boldsymbol{V}^p\boldsymbol{b}_{t-1}|^2}{\|\boldsymbol{V}^p\|^2}\frac{\|\boldsymbol{V}^p\|^2}{\|\boldsymbol{V}\|_F^2} \\
&\quad + \mathbb{E}_x^{t-1}\sum_{p=1}^k \frac{|\boldsymbol{x}_t^p - \boldsymbol{x}_\star^p|^2}{\|\boldsymbol{V}^p\|^2}\frac{\|\boldsymbol{V}^p\|^2}{\|\boldsymbol{V}\|_F^2} \\
&= \|\boldsymbol{b}_{t-1} - \boldsymbol{b}_\star\|^2 - \frac{\|\boldsymbol{V}\boldsymbol{b}_\star - \boldsymbol{V}\boldsymbol{b}_{t-1}\|^2}{\|\boldsymbol{V}\|_F^2} + \mathbb{E}_x^{t-1}\left[\frac{\|\boldsymbol{x}_t - \boldsymbol{x}_\star\|^2}{\|\boldsymbol{V}\|_F^2}\right] \\
&\overset{(vii)}{\leq} \|\boldsymbol{b}_{t-1} - \boldsymbol{b}_\star\|^2 - \frac{\sigma_{\min}^2(\boldsymbol{V})\|\boldsymbol{b}_{t-1} - \boldsymbol{b}_\star\|^2}{\|\boldsymbol{V}\|_F^2} + \mathbb{E}_x^{t-1}\left[\frac{\|\boldsymbol{x}_t - \boldsymbol{x}_\star\|^2}{\|\boldsymbol{V}\|_F^2}\right] \\
&= \alpha_V\|\boldsymbol{b}_{t-1} - \boldsymbol{b}_\star\|^2 + \mathbb{E}_x^{t-1}\left[\frac{\|\boldsymbol{x}_t - \boldsymbol{x}_\star\|^2}{\|\boldsymbol{V}\|_F^2}\right].
\end{aligned}$$

In step (vi), we use iterated expectations to split the expected value $\mathbb{E}^{t-1} = \mathbb{E}_x^{t-1}\mathbb{E}_b^{t-1}$. Step (vii) uses the fact that $\|\boldsymbol{V}(\boldsymbol{b}_{t-1} - \boldsymbol{b}_\star)\|^2 \geq \sigma_{\min}^2(\boldsymbol{V})\|\boldsymbol{b}_{t-1} - \boldsymbol{b}_\star\|^2$ since $\boldsymbol{b}_{t-1} - \boldsymbol{b}_\star$ are in the row span of $\boldsymbol{V}$ for all t. We simplify and obtain the desired bound. ∎

4.3. Proof of main result. We now have all the ingredients we need to prove Theorem 4.1, which we now proceed to below.

Proof of Theorem 4.1. The fact that $\boldsymbol{b}_\star = \boldsymbol{\beta}_\star$ was already argued in scenarios S1 and S3(b) in the previous section, so we do not reproduce its argument here. Given this fact, to prove Theorem 4.1, we only need to invoke the statement of Lemma 4.5

and bound the term $\mathbb{E}_x^{t-1}\|\boldsymbol{x}_t - \boldsymbol{x}_\star\|^2$ using Propositions 4.2 or 4.3 depending on whether we are using Algorithms 1 or 2, respectively.

(a) For Algorithm 1, plugging Proposition 4.2 into the statement of Lemma 4.5 yields

$$\mathbb{E}^{t-1}\|\boldsymbol{b}_t - \boldsymbol{b}_\star\|^2 \le \alpha_V \|\boldsymbol{b}_{t-1} - \boldsymbol{b}_\star\|^2 + \alpha_U^t \frac{\|\boldsymbol{x}_\star\|^2}{\|\boldsymbol{V}\|_F^2}. \tag{15}$$

Taking expectations over the randomness from the first $t-1$ iterations and using the law of iterated expectation, we have

$$\begin{aligned}\mathbb{E}\|\boldsymbol{b}_t - \boldsymbol{b}_\star\|^2 &\le \alpha_V^t \|\boldsymbol{b}_\star\|^2 + \alpha_U^t \frac{\|\boldsymbol{x}_\star\|^2}{\|\boldsymbol{V}\|_F^2} \sum_{h=0}^{t-1} \alpha_V^h \\ &\le \alpha_V^t \|\boldsymbol{b}_\star\|^2 + \alpha_U^t \frac{\|\boldsymbol{x}_\star\|^2}{\|\boldsymbol{V}\|_F^2} \frac{1}{1-\alpha_V} \\ &= \alpha_V^t \|\boldsymbol{b}_\star\|^2 + \theta_V \alpha_U^t \|\boldsymbol{x}_\star\|^2.\end{aligned}$$

(b) For Algorithm 2, plugging Proposition 4.3 into the statement of Lemma 4.5 yields

$$\mathbb{E}^{t-1}\|\boldsymbol{b}_t - \boldsymbol{b}_\star\|^2 \le \alpha_V \|\boldsymbol{b}_{t-1} - \boldsymbol{b}_\star\|^2 + (1+2\kappa_U^2)\alpha_U^{\lfloor t/2 \rfloor} \frac{\|\boldsymbol{x}_\star\|^2}{\|\boldsymbol{V}\|_F^2}.$$

Taking expectations over the remaining randomness, we have

$$\begin{aligned}\mathbb{E}\|\boldsymbol{b}_t - \boldsymbol{b}_\star\|^2 &\le \alpha_V^t \|\boldsymbol{b}_\star\|^2 + \alpha_U^{\lfloor t/2 \rfloor}(1+2\kappa_U^2) \frac{\|\boldsymbol{x}_\star\|^2}{\|\boldsymbol{V}\|_F^2} \sum_{h=0}^{t-1} \alpha_V^h \\ &\le \alpha_V^t \|\boldsymbol{b}_\star\|^2 + \alpha_U^{\lfloor t/2 \rfloor}(1+2\kappa_U^2) \frac{\|\boldsymbol{x}_\star\|^2}{\|\boldsymbol{V}\|_F^2} \frac{1}{1-\alpha_V} \\ &= \alpha_V^t \|\boldsymbol{b}_\star\|^2 + \theta_V \alpha_U^{\lfloor t/2 \rfloor}(1+2\kappa_U^2) \|\boldsymbol{x}_\star\|^2.\end{aligned}$$ □

This concludes the proof of the theorem.

5. Experiments.

In this section we discuss experiments done on both simulated and real data using different algorithms in different settings. The naming convention for the remainder of the paper will be to refer to ALG1-ALG2 as an interlaced algorithm where ALG1 is the algorithm iterate used to solve subsystem (2) and ALG2 is the algorithm used to solve subsystem (3). When an algorithm's name is used alone, we imply applying the algorithm on the full system (1).

In Figure 1 we show our first set of experiments. Entries of $\boldsymbol{U}$, $\boldsymbol{V}$, and $\boldsymbol{\beta}$ are drawn from a standard Gaussian distribution. We set $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{V}$ and $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta}$ if $\boldsymbol{X}$ is consistent and $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{r}$, where $\boldsymbol{r} \in null(\boldsymbol{X}^*)$ (computed in MATLAB using `null()` function) if $\boldsymbol{X}$ is inconsistent. In this first set of experiments, $m, n, k \in \{100, 150, 200\}$ depending on the desired size of k with respect to the over or underdeterminedness of $\boldsymbol{X}$. For example, if $k < m, n$ and $\boldsymbol{X}$ is overdetermined, then $k = 100$, $m = 200$, and $n = 150$. The plots show iteration vs ℓ_2-error, $\|\boldsymbol{b}_t - \boldsymbol{\beta}_\star\|^2$, of each method averaged over 40 runs and allowing each algorithm to run 7×10^4 iterations. The layout of Figure 1 is exactly as in Table 3. For each row, we have a different setting for $\boldsymbol{X}$ and for each column, we vary the size of k depending on the size of $\boldsymbol{X}$. Looking at the overall trends, we see that when $k < m, n$ and when $\boldsymbol{X}$ is overdetermined and consistent and $n < k < m$, there is a method that obtains the optimal solution for

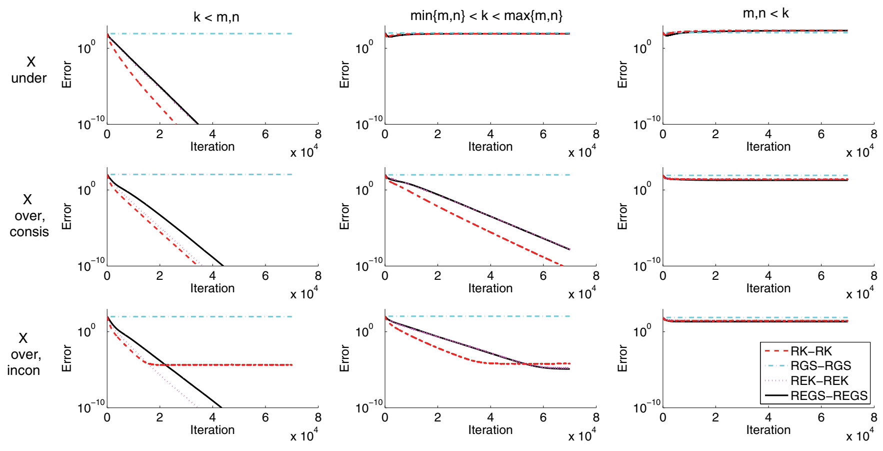

FIG. 1. *This figure shows a summary of convergence for all methods under the variation of possible settings. In the right column, we have that none of the methods convergence. In the middle column, when $\boldsymbol{X}$ is underdetermined or inconsistent none of the methods converge either. In all other variants, the convergence depends on the general behavior of the standard (RK, RGS) algorithms.*

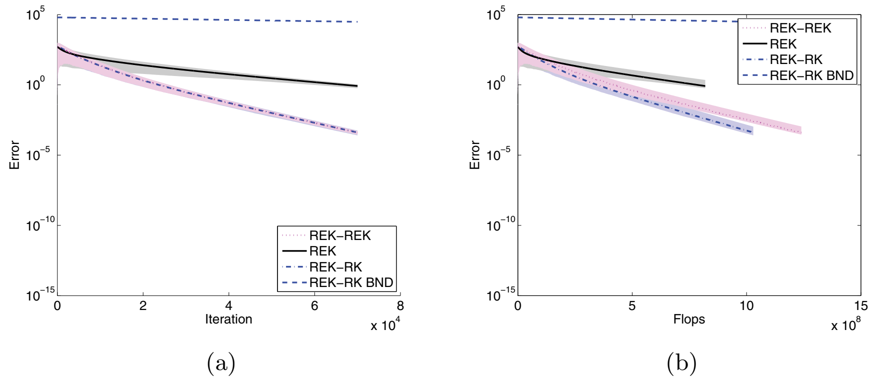

FIG. 2. *When $\boldsymbol{X}$ is overdetermined and inconsistent and $k < m, n$ we propose interlacing iterates of REK and RK to solve subsystems* (2) *and* (3)*. This figure demonstrates the advantage of using REK-RK on an inconsistent system as opposed to REK-REK, the former needing less FLOPS to achieve the same accuracy.*

the system. These results align with the expectations set in Table 3. Looking at each individual subplot, we also find what one would expect according to Table 1. In other words, if $\boldsymbol{U}$ or $\boldsymbol{V}$ is in one of the settings where RK or RGS are expected to fail, then RK-RK or RGS-RGS fail as well.

When $\boldsymbol{X}$ is overdetermined and inconsistent and $k < m, n$ we have that $\boldsymbol{V}$ is underdetermined. In this case, we don't need to interlace iterates of REK and REK together. To work on an underdetermined system, using RK is enough to find the optimal solution of that subsystem. This motivated interlacing iterates of RK with REK. Figure 2 has the same set up as discussed in the previous experiment with the exception of using larger random matrices with $\boldsymbol{X} : 1200 \times 750$ and $k = 500$. In Figure 2(a) we plot iteration vs. ℓ_2-error and in Figure 2(b) we plot FLOPS vs. ℓ_2-error.

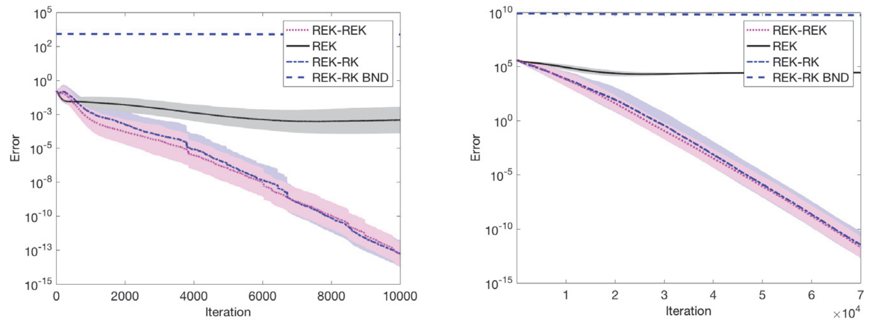

FIG. 3. *In these experiment, we compare the empirical performance of REK, REK-REK, and REK-RK on real world data. Figure* 3(a) *shows the performance of these methods on the wine data set and Figure* 3(b) *shows performance on the bike data set.*

The errors are averaged over 40 runs, with shaded regions representing error within two standard deviations from the mean. Note that Algorithm 2 performs excellently in practice, better than our theoretical upper bound as computed in Theorem 4.1. In this experiment, we see that REK-RK and REK-REK perform comparably in error and that REK-RK is more efficient in FLOPS.

In addition to simulated experiments, we also show the usefulness of these algorithms on real world data sets on wine quality, bike rental data, and Yelp reviews. In all the following experiments, we plot the average ℓ_2-error at the t^{th} iteration over 40 runs and shaded regions representing the ℓ_2-error within two standard deviations. In addition to empirical performance, we also plot the theoretical convergence bound derived in Theorem 4.1 (labeled "BND" in the legends). From these experiments, it is clear that the algorithms perform even better in practice than the worst-case theoretical upper bound. The data sets on wine quality and bike rental data are obtained from the UCI Machine Learning Repository [17]. The wine data set is a sample of $m = 1599$ red wines with $n = 11$ physio-chemical properties of each wine. We choose $k = 5$ and compute $\boldsymbol{U}$ and $\boldsymbol{V}$ using MATLAB's `nnmf()` function for nonnegative matrix factorization. (Recall the motivations from the first section.) Figure 3(a) shows the results from this experiment. The conditioning of $\boldsymbol{X}$, $\boldsymbol{U}$, and $\boldsymbol{V}$ are $\kappa_X^2 = 2.46 \times 10^3$, $\kappa_U^2 = 25.96$, and $\kappa_V^2 = 4.20$ respectively. We plot the ℓ_2-error averaged over 40 runs. Since $\boldsymbol{X}$ has such a large condition number, this impacts the convergence of REK on $\boldsymbol{X}$ negatively as shown by the seemingly horizontal line. (The error is actually decreasing, but incredibly slowly.) We also see that REK-RK and REK-REK are working comparably and significantly faster than REK alone. This can be explained by the better conditioning on $\boldsymbol{U}$ and $\boldsymbol{V}$.

The bike data set contains hourly counts of rental bikes in a bike share system. The data sets contains date as well as weather and seasonal data. There are $m = 17379$ samples and $n = 9$ attributes per sample. We choose $k = 8$ and compute $\boldsymbol{U}$ and $\boldsymbol{V}$ in the same way as with the wine data set. Figure 3(b) shows the results from this experiment. The conditioning of $\boldsymbol{X}$, $\boldsymbol{U}$, and $\boldsymbol{V}$ are $\kappa_X^2 = 94.27$, $\kappa_U^2 = 54.91$, and $\kappa_V^2 = 2.99$, respectively. Similar to Figure 3(a), we see that the convergence of REK suffers from the poorly conditioned matrix $\boldsymbol{X}$. We also see again that REK-REK and REK-RK behave similarly and outperform REK.

To show the advantage of our algorithms on large systems, we create extremely large standard Gaussian matrices $\boldsymbol{U} : 10^6 \times 10^3$ and $\boldsymbol{V} : 10^3 \times 10^4$. These matrices

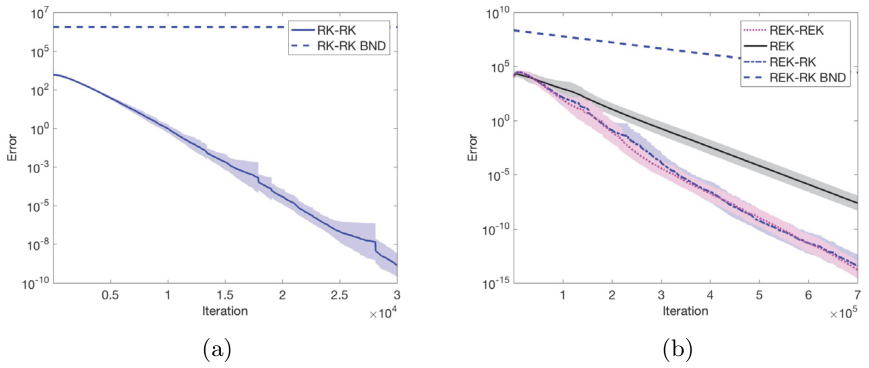

FIG. 4. *We compare the performance of REK, REK-REK, and REK-RK on extremely large datasets. Figure* 4(a) *shows results from an experiment that pushes the limits of memory in MATLAB. Note that in this experiment, we cannot perform RK on the full system as the matrix product requires too much memory and cannot be formed in MATLAB. Figure* 4(b) *shows the performance of our method on the Yelp dataset.*

are so large that the matrix product $\boldsymbol{UV}$ cannot be computed in MATLAB due to memory constraints. These results are shown in Figure 4. We see that without needing to do the matrix computation, we are still able to find the solution to the linear system $\boldsymbol{UV\beta} = \boldsymbol{y}$.

Last, we present the performance of our methods on a large real world data set. We use the Yelp challenge data set [1]. In our setting, we let $\boldsymbol{X} : 10^5 \times 10^4$ be a document term frequency matrix where each row represents a Yelp review and each column represents a word feature. The elements of $\boldsymbol{X}$ contain the frequency at which the word is used in the review. We only use a subset of the amount of data available due to MATLAB memory constraints. Here, $\boldsymbol{y}$ is a vector that represents the number of stars a review received. We choose $k = 5000$. Figure 4(b) shows the results from this experiment using REK, REK-REK, and REK-RK. The conditioning of $\boldsymbol{X}$, $\boldsymbol{U}$, and $\boldsymbol{V}$ are $\kappa_X^2 = 127.3592$, $\kappa_U^2 = 24.274$, and $\kappa_V^2 = 19.096$, respectively. In this large real world data set, we can again see the usefulness of our proposed methods when we are given $\boldsymbol{X} = \boldsymbol{UV}$.

These experiments complement and verify our theoretical findings. In settings which we expect to fail to obtain the least squares or least norm solutions, our experiments show that they do indeed fail. Additionally, where we expect that the optimal solution is obtainable, the experiments show the proposed methods can obtain such solutions and in many instances outperform the original algorithm on the full system. We see that empirically, subsystems are better conditioned than full systems, thus explaining their better performance.

6. Conclusion.

We have proposed two methods interlacing Kaczmarz updates to solve factored systems. For large-scale applications in which the system is stored in factored form for efficiency or the factorization arises naturally, our methods allow one to solve the system without the need to perform the large-scale matrix product first. Our main result proves that our methods provide linear convergence in expectation to the (least-squares or least-norm) solution of (overdetermined or underdetermined) linear systems. Our experiments support these results and show that our methods provide significant computational advantages for factored systems. The interlaced structure of our methods suggests they can be implemented in parallel which would lead to even further computational gains. We leave such details for future work. Additional future work includes the design and analysis of methods that converge to the solution in the settings not covered in this paper, i.e., the gray cells of Table 3. Although its practical implications are not immediately clear to us, these may still be of theoretical interest.

Acknowledgment. The authors thank the Institute of Pure and Applied Mathematics (IPAM), where this collaboration started.

REFERENCES

[1] *Yelp dataset challenge*, https://www.yelp.com/dataset_challenge.

[2] G. Adomavicius and A. Tuzhilin, *Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions*, IEEE Trans. Knowl. Data. Eng., 17 (2005), pp. 734–749.

[3] M. Benzi, *Key moments in the history of numerical analysis*, in Proceedings of the SIAM Conference on Applied Linear Algebra, 2009.

[4] C. L. Byrne, *Applied Iterative Methods*, A K Peters, Wellesley, MA, 2008.

[5] E. J. Candès and B. Recht, *Exact matrix completion via convex optimization*, Found. Comput. Math., 9 (2009), pp. 717–772.

[6] B. Dumitrescu, *On the relation between the randomized extended Kaczmarz algorithm and coordinate descent*, BIT, 55 (2014), pp. 1–11.

[7] J. Goes, T. Zhang, R. Arora, and G. Lerman, *Robust stochastic principal component analysis.*, in Proceeding of AISTATS, 2014, pp. 266–274.

[8] R. Gordon, R. Bender, and G. T. Herman, *Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography*, J. Theoret. Biol., 29 (1970), pp. 471–481.

[9] G. T. Herman, *Fundamentals of Computerized Tomography: Image Reconstruction from Projections*, Springer, New York, 2009.

[10] P. Jain, P. Netrapalli, and S. Sanghavi, *Low-rank matrix completion using alternating minimization*, in Proceedings of the 45th Annual ACM Symposium on Theory of Computing, ACM, 2013, pp. 665–674.

[11] S. Kaczmarz, *Angenäherte auflösung von systemen linearer gleichungen*, Bull. Int. Acad. Polon. Sci. Lett. Ser. A, (1937), pp. 335–357.

[12] R. H. Keshavan, A. Montanari, and S. Oh, *Matrix completion from noisy entries*, J. Mach. Learn. Res., 11 (2010), pp. 2057–2078.

[13] V. Koltchinskii, K. Lounici, and A. B. Tsybakov, *Nuclear-norm penalization and optimal rates for noisy low-rank matrix completion*, Ann. Statist., 39 (2011), pp. 2302–2329.

[14] Y. Koren, R. Bell, C. Volinsky, et al., *Matrix factorization techniques for recommender systems*, Computer, 42 (2009), pp. 30–37.

[15] D. D. Lee and H. S. Seung, *Algorithms for non-negative matrix factorization*, in Peoceedings of Advances in Neural Information Processing Systems, 2001, pp. 556–562.

[16] D. Leventhal and A. S. Lewis, *Randomized methods for linear constraints: Convergence rates and conditioning*, Math. Oper. Res., 35 (2010), pp. 641–654, https://doi.org/10.1287/moor.1100.0456.

[17] M. Lichman, *UCI Machine Learning Repository*, 2013, http://archive.ics.uci.edu/ml.

[18] A. Ma, A. Flenner, D. Needell, and A. G. Percus, *Improving image clustering using sparse text and the wisdom of the crowds*, in Procedings of the 48th Asilomar Conference on Signals, Systems and Computers, IEEE, 2014, pp. 1555–1557.

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

[20] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King, *Recommender systems with social regularization*, in Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, ACM, 2011, pp. 287–296.

[21] F. Natterer, *The Mathematics of Computerized Tomography*, Classics Appl. Math. 32, SIAM, Philadelphia, 2001, https://doi.org/10.1137/1.9780898719284.

[22] D. Needell, *Randomized Kaczmarz solver for noisy linear systems*, BIT, 50 (2010), pp. 395–403, https://doi.org/10.1007/s10543-010-0265-5.

[23] D. Needell, N. Sbrero, and R. Ward, *Stochastic gradient descent and the randomized Kaczmarz algorithm*, Math. Program. Ser. A, 155 (2016), pp. 549–573.

[24] T. Strohmer and R. Vershynin, *Comments on the randomized Kaczmarz method*, J. Fourier Anal. Appl., 15 (2009), pp. 437–440, https://doi.org/10.1007/s00041-009-9082-0.

[25] T. Strohmer and R. Vershynin, *A randomized Kaczmarz algorithm with exponential convergence*, J. Fourier Anal. Appl., 15 (2009), pp. 262–278, https://doi.org/10.1007/s00041-008-9030-4.

[26] G. Takács, I. Pilászy, B. Németh, and D. Tikk, *Investigation of various matrix factorization methods for large recommender systems*, in Proceedings of the International Conference on Data Mining Workshops, IEEE, 2008, pp. 553–562.

[27] K. Verbert, N. Manouselis, X. Ochoa, M. Wolpers, H. Drachsler, I. Bosnic, and E. Duval, *Context-aware recommender systems for learning: A survey and future challenges*, IEEE Trans. Learn. Technol., 5 (2012), pp. 318–335.

[28] J. Wright, A. Ganesh, S. Rao, Y. Peng, and Y. Ma, *Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization*, in Proceedings of the Advances in Neural Information Processing Systems, 2009, pp. 2080–2088.

[29] W. Xu, X. Liu, and Y. Gong, *Document clustering based on non-negative matrix factorization*, in Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2003, pp. 267–273.

[30] A. Zouzias and N. M. Freris, *Randomized extended Kaczmarz for solving least squares*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 773–793.