SIAM J. MATRIX ANAL. APPL.
Vol. 34, No. 2, pp. 773–793

RANDOMIZED EXTENDED KACZMARZ FOR SOLVING LEAST SQUARES*

ANASTASIOS ZOUZIAS† AND NIKOLAOS M. FRERIS‡

Abstract. We present a randomized iterative algorithm that exponentially converges in the mean square to the minimum ℓ_2-norm least squares solution of a given linear system of equations. The expected number of arithmetic operations required to obtain an estimate of given accuracy is proportional to the squared condition number of the system multiplied by the number of nonzero entries of the input matrix. The proposed algorithm is an extension of the randomized Kaczmarz method that was analyzed by Strohmer and Vershynin.

Key words. linear least squares, minimum-length solution, sparse matrix, overdetermined system, underdetermined system, iterative method, random sampling, LAPACK, randomized algorithms

AMS subject classifications. 65F10, 65F20, 65F50

DOI. 10.1137/120889897

1. Introduction.

The Kaczmarz method is an iterative projection algorithm for solving linear systems of equations [Kac37]. Due to its simplicity, the Kaczmarz method has found numerous applications including image reconstruction, distributed computation, and signal processing to name a few [FCM+92, Her80, Nat01, FZ12]; see [Cen81] for more applications. The Kaczmarz method was independently rediscovered in the field of image reconstruction under the name ART (algebraic reconstruction technique) [GBH70]; see also [CZ97, Her80] for additional references. It has also been applied to more general settings; see [Cen81, Table 1] and [Tom55, McC75] for nonlinear versions of the Kaczmarz method.

Let $\mathsf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{b} \in \mathbb{R}^m$. Throughout the paper all vectors are assumed to be column vectors. The Kaczmarz method operates as follows: Initially, it starts with an arbitrary vector $\mathbf{x}^{(0)} \in \mathbb{R}^n$. In each iteration, the Kaczmarz method runs through the rows of A in a cyclic manner[1] and for each selected row, say the ith row $\mathsf{A}^{(i)}$, it orthogonally projects the current estimate vector onto the affine hyperplane defined by the ith constraint of $\mathsf{A}\mathbf{x} = \mathbf{b}$, i.e., $\{\mathbf{x} \mid \langle \mathsf{A}^{(i)},\ \mathbf{x}\rangle = b_i\}$, where $\langle\cdot,\ \cdot\rangle$ is the Euclidean inner product. More precisely, assuming that the i_kth row has been selected at the kth iteration, then the $(k+1)$th estimate vector $\mathbf{x}^{(k+1)}$ is obtained by:

$$\mathbf{x}^{(k+1)} := \mathbf{x}^{(k)} + \lambda_k \frac{b_{i_k} - \left\langle \mathsf{A}^{(i_k)},\ \mathbf{x}^{(k)}\right\rangle}{\left\|\mathsf{A}^{(i_k)}\right\|_2^2}\mathsf{A}^{(i_k)},$$

where $\lambda_k \in \mathbb{R}$ are the so-called relaxation parameters and $\|\cdot\|_2$ denotes the Euclidean norm. The original Kaczmarz method corresponds to $\lambda_k = 1$ for all $k \geq 0$ and all

*Received by the editors September 5, 2012; accepted for publication (in revised form) by I. C. F. Ipsen March 20, 2013; published electronically June 20, 2013.
http://www.siam.org/journals/simax/34-2/88989.html

†Department of Computer Science, University of Toronto, Toronto, ON, Canada (zouzias@cs.toronto.edu). Part of this work was done while the author was visiting the Department of Computer Science at Princeton University.

‡School of Computer and Communication Sciences (IC), École Polytecnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland (nikolaos.freris@epfl.ch). This author would like to acknowledge Qualcomm Inc. for partially supporting this research.

[1]That is, selecting the indices of the rows from the sequence $1, 2, \ldots, m, 1, 2, \ldots$.

other settings of the λ_k's are usually referred to as the *relaxed Kaczmarz method* in the literature [Cen81, Gal03].

Kaczmarz proved that this process converges to the unique solution for square nonsingular matrices [Kac37], but without any attempt to bound the rate of convergence. Bounds on the rate of convergence of the Kaczmarz method are given in [McC75, Ans84] and [Gal03, Theorem 4.4, p. 120]. In addition, an error analysis of the Kaczmarz method under the finite precision model of computation is given in [Kni93, Kni96].

Nevertheless, the relaxed Kaczmarz method converges even if the linear system $\mathsf{A}\mathbf{x} = \mathbf{b}$ is overdetermined ($m > n$) and has no solution. In this case and provided that A has full column rank, the Kaczmarz method converges to the least squares estimate. This was first observed by Whitney and Meany [WM67] who proved that the relaxed Kaczmarz method converges provided that the relaxation parameters are within $[0, 2]$ and $\lambda_k \to 0$; see also [CEG83, Theorem 1], [Tan71, HN90] for additional references.

In the literature there was empirical evidence that selecting the rows nonuniformly at random may be more effective than selecting the rows via Kaczmarz's cyclic manner [HM93, FCM+92]. Towards explaining such an empirical evidence, Strohmer and Vershynin proposed a simple randomized variant of the Kaczmarz method that has exponential convergence *in mean square* [SV09] (assuming that the linear system is solvable). A randomized iterative algorithm that computes a sequence of random vectors $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \ldots$ is said to *converge in mean square* to a vector $\mathbf{x}^*$ if and only if $\mathbb{E}\left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\|_2^2 \to 0$ as $k \to \infty$, where the expectation is taken over the random choices of the algorithm. Along the same lines of [SV09], the authors of [CP12] proved that the randomized Kaczmarz method almost surely converges in certain cases; see also [LL10] for extensions of the randomized Kaczmarz method to linear constraints and [NT12] for the analysis of a randomized block Kaczmarz method. Soon after [SV09], Needell analyzed the behavior of the randomized Kaczmarz method for the case of full column rank linear systems that do not have any solution [Nee10]. Namely, Needell proved that the randomized Kaczmarz estimate vector is (in the limit) within a fixed distance from the least squares solution and also that this distance is proportional to the distance of $\mathbf{b}$ from the column space of A. In other words, Needell proved that the randomized Kaczmarz method is effective when least squares estimation is effective, i.e., the least squares error is negligible.

In this paper we present a randomized iterative least squares solver (Algorithm 3) that exponentially converges in mean square to the minimum Euclidean norm solution of

$$\min_{\mathbf{x}\in\mathbb{R}^n} \|\mathsf{A}\mathbf{x} - \mathbf{b}\|_2 . \tag{1.1}$$

The proposed algorithm is based on [SV09, Nee10] and inspired by [Pop95, Pop98]. More precisely the proposed algorithm can be thought of as a randomized variant of Popa's extended Kaczmarz method [Pop95, Pop98]; therefore we call it the *randomized extended Kaczmarz* (REK).

Organization of the paper. In section 2, we briefly discuss related work on the design of deterministic and randomized algorithms for solving least squares problems. In section 3, we present a randomized iterative algorithm for projecting a vector onto a subspace (represented as the column space of a given matrix) which may be of independent interest. In addition, we discuss the convergence properties of the randomized Kaczmarz algorithm for solvable systems (section 3.2) and recall its

analysis for nonsolvable systems as in [Nee10] (section 3.3). In section 4, we present and analyze the randomized extended Kaczmarz algorithm. Finally, in section 5 we provide a numerical evaluation of the proposed algorithm.

2. Least squares solvers. In this section we give a brief discussion on least squares solvers including deterministic direct and iterative algorithms together with recently proposed randomized algorithms. For a detailed discussion on deterministic methods, the reader is referred to [Bj96]. In addition, we position our contribution in context with prior work.

Deterministic algorithms. In the literature, several methods have been proposed for solving least squares problems of the form (1.1). Here we briefly describe a representative sample of such methods including the use of QR factorization with pivoting, the use of singular value decomposition (SVD), and iterative methods such as Krylov subspace methods applied on the normal equations [Saa03]. LAPACK provides robust implementations of the first two methods; `DGELSY` uses QR factorization with pivoting and `DGELSD` uses SVD [ABD+90]. For the case of iterative methods, LSQR is equivalent to applying the conjugate gradient method on the normal equations [PS82] which is a robust and numerically stable method.

Randomized algorithms. To the best of our knowledge, most randomized algorithms for approximately solving least squares have been proposed in the theoretical computer science literature. These are mainly based on the following generic two step procedure: first randomly (and efficiently) project the linear system into sufficiently many dimensions, and second return the solution of the downsampled linear system as an approximation to the original optimal solution [DMM06, Sar06, CW09, NDT09, MZ11, DMMS11]; see also [CW12]. Concentration of measure arguments imply that the optimal solution of the downsampled system is close to the optimal solution of the original system. The accuracy of the approximate solution using this approach depends on the sample size to achieve relative accuracy ε; the sample size should depend inverse polynomially on ε. This makes these approaches unsuitable for the high-precision regime of error that is considered here.

A different approach is the so-called randomized preconditioning method; see [RT08, AMT10]. The authors of [AMT10] implemented Blendenpik, a high-precision least squares solver. Blendenpik consists of two steps: In the first step, the input matrix is randomly projected and an effective preconditioning matrix is extracted from the projected matrix. In the second step, an iterative least squares solver such as the LSQR algorithm of Paige and Saunders [PS82] is applied to the preconditioned system. Blendenpik is effective for both overdetermined and underdetermined problems.

A parallel iterative least squares solver based on normal random projections called `LSRN` was recently implemented by Meng, Saunders, and Mahoney [MSM11]. `LSRN` consists of two phases. In the first preconditioning phase, the original system is projected using random normal projection from which a preconditioner is extracted. In the second step, an iterative method such as LSQR or the Chebyshev semi-iterative method [GV61] is applied on the preconditioned system. This approach is also effective for overdetermined and underdetermined least squares problems assuming the existence of a parallel computational environment.

2.1. Relation with our contribution. In section 5, we compare the randomized extended Kaczmarz algorithm against `DGELSY`, `DGELSD`, and Blendenpik. `LSRN` [MSM11] did not perform well under a setup in which no parallelization is allowed, so we do not include `LSRN`'s evaluation. The numerical evaluation of section 5 indicates that the randomized extended Kaczmarz is effective in the case of sparse,

well-conditioned, and highly rectangular (both overdetermined and underdetermined) least squares problems; see Figure 5.1. Moreover, the randomized extended Kaczmarz algorithm also has comparable performance with LAPACK's routine for the dense random input matrices; see Figure 5.2. On the other hand, a preconditioned version of the proposed algorithm does not perform well under the case of ill-conditioned matrices; see Figure 5.3.

3. Background.

Preliminaries and notation. For an integer $m \geq 1$, let $[m] := \{1, \ldots, m\}$. Throughout the paper all vectors are assumed to be column vectors. We denote the rows and columns of A by $\mathsf{A}^{(1)}, \ldots, \mathsf{A}^{(m)}$ and $\mathsf{A}_{(1)}, \ldots, \mathsf{A}_{(n)}$, respectively (both viewed as column vectors). $\mathcal{R}(\mathsf{A})$ denotes the column space of A, i.e., $\mathcal{R}(\mathsf{A}) := \{\mathsf{A}\mathbf{x} \mid \mathbf{x} \in \mathbb{R}^n\}$ and $\mathcal{R}(\mathsf{A})^\perp$ denotes the orthogonal complement of $\mathcal{R}(\mathsf{A})$. Given any $\mathbf{b} \in \mathbb{R}^m$, we can uniquely write it as $\mathbf{b}_{\mathcal{R}(\mathsf{A})} + \mathbf{b}_{\mathcal{R}(\mathsf{A})^\perp}$, where $\mathbf{b}_{\mathcal{R}(\mathsf{A})}$ is the projection of $\mathbf{b}$ onto $\mathcal{R}(\mathsf{A})$. We use $\|\mathsf{A}\|_{\mathrm{F}} := \sqrt{\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2}$ and $\|\mathsf{A}\|_2 := \max_{\mathbf{x}\neq\mathbf{0}} \|\mathsf{A}\mathbf{x}\|_2 / \|\mathbf{x}\|_2$ to denote the Frobenius norm and spectral norm, respectively. Let $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\mathbf{rank}(\mathsf{A})}$ be the nonzero singular values of A. We will usually refer to σ_1 and $\sigma_{\mathbf{rank}(\mathsf{A})}$ as $\sigma_{\max}$ and $\sigma_{\min}$, respectively. The Moore–Penrose pseudoinverse of A is denoted by $\mathsf{A}^\dagger$ [GL96]. Recall that $\left\|\mathsf{A}^\dagger\right\|_2 = 1/\sigma_{\min}$. For any nonzero real matrix A, we define

$$\kappa_{\mathrm{F}}^2(\mathsf{A}) := \|\mathsf{A}\|_{\mathrm{F}}^2 \left\|\mathsf{A}^\dagger\right\|_2^2. \tag{3.1}$$

Related to κ_{F} is the scaled squared condition number introduced by Demmel in [Dem88]; see also [SV09]. It is easy to check that the above parameter $\kappa_{\mathrm{F}}^2(\mathsf{A})$ is related to the condition number of A, $\kappa^2(\mathsf{A}) := \sigma_{\max}^2/\sigma_{\min}^2$, via the inequalities: $\kappa^2(\mathsf{A}) \leq \kappa_{\mathrm{F}}^2(\mathsf{A}) \leq \mathbf{rank}(\mathsf{A}) \cdot \kappa^2(\mathsf{A})$. We denote by $\mathbf{nnz}(\cdot)$ the number of nonzero entries of its argument matrix. We define the *average row sparsity* and *average column sparsity* of A by $\mathrm{R}_{\mathrm{avg}}$ and $\mathrm{C}_{\mathrm{avg}}$, respectively, as follows:

$$\mathrm{R}_{\mathrm{avg}} := \sum_{i=1}^m q_i \mathbf{nnz}\left(\mathsf{A}^{(i)}\right) \quad \text{and} \quad \mathrm{C}_{\mathrm{avg}} := \sum_{j=1}^n p_j \mathbf{nnz}\left(\mathsf{A}_{(j)}\right),$$

where $q_i := \left\|\mathsf{A}^{(i)}\right\|_2^2 / \|\mathsf{A}\|_{\mathrm{F}}^2$ for every $i \in [m]$ and $p_j := \left\|\mathsf{A}_{(j)}\right\|_2^2 / \|\mathsf{A}\|_{\mathrm{F}}^2$ for every $i \in [n]$. The following fact will be used extensively in the paper.

FACT 3.1. *Let* A *be any nonzero real* $m \times n$ *matrix and* $\mathbf{b} \in \mathbb{R}^m$. *Denote by* $\mathbf{x}_{LS} := \mathsf{A}^\dagger\mathbf{b}$. *Then* $\mathbf{x}_{LS} = \mathsf{A}^\dagger\mathbf{b}_{\mathcal{R}(\mathsf{A})}$.

We frequently use the inequality $1 - t \leq \exp(-t)$ for every $t \leq 1$. We conclude this section by collecting a few basic facts from probability theory that will be frequently used. For any random variable X, we denote its expectation by $\mathbb{E}[X]$ or $\mathbb{E}\,X$. If X is a nonnegative random variable, Markov's inequality states that $\mathbb{P}(X > t) \leq t^{-1}\,\mathbb{E}[X]$ for $t > 0$. Let X and Y be two random variables, then $\mathbb{E}[X+Y] = \mathbb{E}[X] + \mathbb{E}[Y]$. We will refer to this fact as *linearity of expectation*. Let $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_l$ be a set of events defined over some probability space holding with probabilities $p_1, p_2, \ldots p_l$, respectively, then $\mathbb{P}(\mathcal{E}_1 \cup \mathcal{E}_2 \cup \cdots \cup \mathcal{E}_l) \leq \sum_{i=1}^l p_i$. We refer to this fact as *union bound*.

3.1. Randomized approximate orthogonal projection.

In this section we present a randomized iterative algorithm (Algorithm 1) that, given any vector $\mathbf{b} \in \mathbb{R}^m$ and a linear subspace of $\mathbb{R}^m$ represented as the column space of a given matrix A,

ALGORITHM 1. RANDOMIZED ORTHOGONAL PROJECTION.

procedure $(\mathsf{A}, \mathbf{b}, T)$ ▷ $\mathsf{A} \in \mathbb{R}^{m\times n}, \mathbf{b} \in \mathbb{R}^m$, $T \in \mathbb{N}$
  Initialize $\mathbf{z}^{(0)} = \mathbf{b}$
  for $k = 0, 1, 2, \ldots, T-1$ **do**
    Pick $j_k \in [n]$ with probability $p_j := \left\|\mathsf{A}_{(j)}\right\|_2^2 / \left\|\mathsf{A}\right\|_{\mathrm{F}}^2$, $j \in [n]$
    Set $\mathbf{z}^{(k+1)} = \left(\mathbf{I}_m - \frac{\mathsf{A}_{(j_k)}\mathsf{A}_{(j_k)}^{\top}}{\left\|\mathsf{A}_{(j_k)}\right\|_2^2}\right)\mathbf{z}^{(k)}$
  end for
  Output $\mathbf{z}^{(T)}$
end procedure

approximately computes the orthogonal projection of $\mathbf{b}$ onto the column space of A (denoted $\mathbf{b}_{\mathcal{R}(\mathsf{A})} = \mathsf{A}\mathsf{A}^{\dagger}\mathbf{b}$); see [CRT11] for a different approach.

Algorithm 1 is iterative. Initially, it starts with $\mathbf{z}^{(0)} = \mathbf{b}$. At the kth iteration, the algorithm randomly selects a column $\mathsf{A}_{(j)}$ of A for some j, and updates $\mathbf{z}^{(k)}$ by projecting it onto the orthogonal complement of the linear span of $\mathsf{A}_{(j)}$. The result is that by randomly selecting the columns of A with probability proportional to their squared norms yields exponential convergence in the mean square to $\mathbf{b}_{\mathcal{R}(\mathsf{A})^{\perp}}$. After T iterations, the algorithm outputs $\mathbf{z}^{(T)}$ whence by orthogonality $\mathbf{b} - \mathbf{z}^{(T)}$ serves as an approximation for $\mathbf{b}_{\mathcal{R}(\mathsf{A})}$. The next theorem bounds the expected rate of convergence for Algorithm 1.

THEOREM 3.2. *Let $\mathsf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{R}^m$, and $T > 1$ be the input to Algorithm* 1. *Fix any integer $k > 0$. In exact arithmetic, after k iterations of Algorithm* 1 *it holds that*

$$\mathbb{E}\left\|\mathbf{z}^{(k)} - \mathbf{b}_{\mathcal{R}(\mathsf{A})^{\perp}}\right\|_2^2 \le \left(1 - \frac{1}{\kappa_F^2(\mathsf{A})}\right)^k \left\|\mathbf{b}_{\mathcal{R}(\mathsf{A})}\right\|_2^2.$$

Moreover, each iteration of Algorithm 1 *requires in expectation (over the random choices of the algorithm) at most $5C_{avg}$ arithmetic operations.*

Remark 1. A possible stopping criterion for Algorithm 1 is to regularly check $\frac{\|\mathsf{A}^{\top}\mathbf{z}^{(k)}\|}{\|\mathsf{A}\|_{\mathrm{F}}\|\mathbf{z}^{(k)}\|} \le \varepsilon$ for some given accuracy $\varepsilon > 0$. It is easy to see that whenever this criterion is satisfied, it holds that $\|\mathbf{b}_{\mathcal{R}(\mathsf{A})^{\perp}} - \mathbf{z}^{(k)}\|_2 / \|\mathbf{z}^{(k)}\|_2 \le \varepsilon\kappa_{\mathrm{F}}(\mathsf{A})$, i.e., $\mathbf{b} - \mathbf{z}^{(k)} \approx \mathbf{b}_{\mathcal{R}(\mathsf{A})}$.

We devote the rest of this subsection to prove Theorem 3.2. Define $\mathsf{P}(j) := \mathbf{I}_m - \frac{\mathsf{A}_{(j)}\mathsf{A}_{(j)}^{\top}}{\|\mathsf{A}_{(j)}\|_2^2}$ for every $j \in [n]$. Observe that $\mathsf{P}(j)\mathsf{P}(j) = \mathsf{P}(j)$, i.e., $\mathsf{P}(j)$ is a projection matrix. Let X be a random variable over $\{1, 2, \ldots, n\}$ that picks index j with probability $\|\mathsf{A}_{(j)}\|_2^2 / \|\mathsf{A}\|_{\mathrm{F}}^2$. It holds that $\mathbb{E}[\mathsf{P}(X)] = \mathbf{I}_m - \mathsf{A}\mathsf{A}^{\top} / \|\mathsf{A}\|_{\mathrm{F}}^2$. Later we will make use of the following fact.

FACT 3.3. *For every vector $\mathbf{u}$ in the column space of A, it holds $\|(\mathbf{I}_m - \frac{\mathsf{A}\mathsf{A}^{\top}}{\|\mathsf{A}\|_{\mathrm{F}}^2})\mathbf{u}\|_2 \le (1 - \frac{\sigma_{\min}^2}{\|\mathsf{A}\|_{\mathrm{F}}^2})\|\mathbf{u}\|_2$.*

Define $\mathbf{e}^{(k)} := \mathbf{z}^{(k)} - \mathbf{b}_{\mathcal{R}(\mathsf{A})^{\perp}}$ for every $k \ge 0$. A direct calculation implies that

$$\mathbf{e}^{(k)} = \mathsf{P}(j_k)\mathbf{e}^{(k-1)}.$$

Indeed, $\mathbf{e}^{(k)} = \mathbf{z}^{(k)} - \mathbf{b}_{\mathcal{R}(\mathsf{A})^{\perp}} = \mathsf{P}(j_k)\mathbf{z}^{(k-1)} - \mathbf{b}_{\mathcal{R}(\mathsf{A})^{\perp}} = \mathsf{P}(j_k)(\mathbf{e}^{(k-1)} + \mathbf{b}_{\mathcal{R}(\mathsf{A})^{\perp}}) - \mathbf{b}_{\mathcal{R}(\mathsf{A})^{\perp}} = \mathsf{P}(j_k)\mathbf{e}^{(k-1)}$ using the definitions of $\mathbf{e}^{(k)}$, $\mathbf{z}^{(k)}$, $\mathbf{e}^{(k-1)}$, and the fact that $\mathsf{P}(j_k)\mathbf{b}_{\mathcal{R}(\mathsf{A})^{\perp}} = \mathbf{b}_{\mathcal{R}(\mathsf{A})^{\perp}}$ for any $j_k \in [n]$. Moreover, it is easy to see that for every

$k \geq 0$, $\mathbf{e}^{(k)}$ is in the column space of A, since $\mathbf{e}^{(0)} = \mathbf{b} - \mathbf{b}_{\mathcal{R}(\mathsf{A})^\perp} = \mathbf{b}_{\mathcal{R}(\mathsf{A})} \in \mathcal{R}(\mathsf{A})$, $\mathbf{e}^{(k)} = \mathsf{P}(j_k)\mathbf{e}^{(k-1)}$, and in addition $\mathsf{P}(j_k)$ is a projection matrix for every $j_k \in [n]$.

Let $X_1, X_2, \ldots$ be a sequence of independently and identically distributed (i.i.d.) random variables distributed as X. For ease of notation, we denote by $\mathbb{E}_{k-1}[\cdot] = \mathbb{E}_{X_k}[\cdot \mid X_1, X_2, \ldots, X_{k-1}]$, the conditional expectation conditioned on the first $(k-1)$ iterations of the algorithm. It follows that

$$\begin{aligned}
\mathbb{E}_{k-1}\left\|\mathbf{e}^{(k)}\right\|_2^2 &= \mathbb{E}_{k-1}\left\|\mathsf{P}(X_k)\mathbf{e}^{(k-1)}\right\|_2^2 \;=\; \mathbb{E}_{k-1}\left\langle \mathsf{P}(X_k)\mathbf{e}^{(k-1)},\; \mathsf{P}(X_k)\mathbf{e}^{(k-1)}\right\rangle \\
&= \mathbb{E}_{k-1}\left\langle \mathbf{e}^{(k-1)},\; \mathsf{P}(X_k)\mathsf{P}(X_k)\mathbf{e}^{(k-1)}\right\rangle \;=\; \left\langle \mathbf{e}^{(k-1)},\; \mathbb{E}_{k-1}[\mathsf{P}(X_k)]\mathbf{e}^{(k-1)}\right\rangle \\
&\leq \left\|\mathbf{e}^{(k-1)}\right\|_2 \left\|\left(\mathbf{I}_m - \frac{\mathsf{A}\mathsf{A}^\top}{\|\mathsf{A}\|_{\mathrm{F}}^2}\right)\mathbf{e}^{(k-1)}\right\|_2 \;\leq\; \left(1 - \frac{\sigma_{\min}^2}{\|\mathsf{A}\|_{\mathrm{F}}^2}\right)\left\|\mathbf{e}^{(k-1)}\right\|_2^2,
\end{aligned}$$

where we used linearity of expectation, the fact that $\mathsf{P}(\cdot)$ is a projection matrix, the Cauchy–Schwarz inequality, and Fact 3.3. Repeating the same argument $k-1$ times we get

$$\mathbb{E}\left\|\mathbf{e}^{(k)}\right\|_2^2 \leq \left(1 - \frac{1}{\kappa_{\mathrm{F}}^2(\mathsf{A})}\right)^k \left\|\mathbf{e}^{(0)}\right\|_2^2.$$

Note that $\mathbf{e}^{(0)} = \mathbf{b} - \mathbf{b}_{\mathcal{R}(\mathsf{A})^\perp} = \mathbf{b}_{\mathcal{R}(\mathsf{A})}$ to conclude.

Step 5 can be rewritten as $\mathbf{z}^{(k+1)} = \mathbf{z}^{(k)} - (\langle \mathsf{A}_{(j_k)},\; \mathbf{z}^{(k)}\rangle / \|\mathsf{A}_{(j_k)}\|_2^2)\mathsf{A}_{(j_k)}$. At every iteration, the inner product and the update from $\mathbf{z}^{(k)}$ to $\mathbf{z}^{(k+1)}$ require at most $5\mathbf{nnz}(\mathsf{A}_{(j_k)})$ operations for some $j_k \in [n]$; hence in expectation each iteration requires at most $\sum_{j=1}^n 5p_j\mathbf{nnz}(\mathsf{A}_{(j)}) = 5\mathrm{C}_{\mathrm{avg}}$ operations.

3.2. Randomized Kaczmarz. Strohmer and Vershynin proposed the following randomized variant of the Kaczmarz algorithm (Algorithm 2); see [SV09] for more details. The following theorem is a restatement of the main result of [SV09] without imposing the full column rank assumption.

ALGORITHM 2. RANDOMIZED KACZMARZ [SV09].

procedure (A, $\mathbf{b}$, T) ▷ $\mathsf{A} \in \mathbb{R}^{m\times n}, \mathbf{b} \in \mathbb{R}^m$
  Set $\mathbf{x}^{(0)}$ to be any vector in the row space of A
  for $k = 0, 1, 2, \ldots, T-1$ **do**
    Pick $i_k \in [m]$ with probability $q_i := \left\|\mathsf{A}^{(i)}\right\|_2^2 / \|\mathsf{A}\|_{\mathrm{F}}^2, i \in [m]$
    Set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \frac{b_{i_k} - \langle \mathbf{x}^{(k)},\; \mathsf{A}^{(i_k)}\rangle}{\left\|\mathsf{A}^{(i_k)}\right\|_2^2}\mathsf{A}^{(i_k)}$
  end for
  Output $\mathbf{x}^{(T)}$
end procedure

THEOREM 3.4. *Let $\mathsf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{R}^m$, and $T > 1$ be the input to Algorithm 2. Assume that $\mathsf{A}\mathbf{x} = \mathbf{b}$ has a solution and denote $\mathbf{x}_{LS} := \mathsf{A}^\dagger\mathbf{b}$. In exact arithmetic, Algorithm 2 converges to $\mathbf{x}_{LS}$ in mean square:*

$$\mathbb{E}\left\|\mathbf{x}^{(k)} - \mathbf{x}_{LS}\right\|_2^2 \leq \left(1 - \frac{1}{\kappa_F^2(\mathsf{A})}\right)^k \left\|\mathbf{x}^{(0)} - \mathbf{x}_{LS}\right\|_2^2 \quad \forall\; k > 0. \tag{3.2}$$

Remark 2. The above theorem has been proved in [SV09] for the case of full column rank. Also, the rate of expected convergence in [SV09] is $1 - 1/\widetilde{\kappa}^2(\mathsf{A})$, where

$\widetilde{\kappa}^2(\mathsf{A}) := \|\mathsf{A}\|_{\mathrm{F}}^2 / \sigma_{\min(m,n)}(\mathsf{A}^\top \mathsf{A})$. Notice that if $\mathbf{rank}(\mathsf{A}) < n$, then $\widetilde{\kappa}^2(\mathsf{A})$ is infinite whereas $\kappa_{\mathrm{F}}^2(\mathsf{A})$ is bounded.

We devote the rest of this subsection to prove Theorem 3.4 following [SV09]. The proof is based on the following two elementary lemmas which both appeared in [SV09]. However, in our setting, the second lemma is not identical to that in [SV09]. We defer their proofs to the Appendix.

LEMMA 3.5 (orthogonality). *Assume that* $\mathsf{A}\mathbf{x} = \mathbf{b}$ *has a solution and use the notation of Algorithm* 2, *then* $\mathbf{x}^{(k+1)} - \mathbf{x}_{LS}$ *is perpendicular to* $\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$ *for any* $k \geq 0$. *In particular, in exact arithmetic it holds that* $\|\mathbf{x}^{(k+1)} - \mathbf{x}_{LS}\|_2^2 = \|\mathbf{x}^{(k)} - \mathbf{x}_{LS}\|_2^2 - \|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|_2^2$.

The above lemma provides a formula for the error at each iteration. Ideally, we seek to minimize the error at each iteration which is equivalent to maximizing $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|_2$ over the choice of the row projections of the algorithm. The next lemma suggests that randomly picking the rows of A reduces the error in mean square.

LEMMA 3.6 (expected error reduction). *Assume that* $\mathsf{A}\mathbf{x} = \mathbf{b}$ *has a solution. Let* Z *be a random variable over* $[m]$ *with distribution* $\mathbb{P}(Z = i) = \frac{\|\mathsf{A}^{(i)}\|_2^2}{\|\mathsf{A}\|_{\mathrm{F}}^2}$ *and assume that* $\mathbf{x}^{(k)}$ *is a vector in the row space of* A. *If* $\mathbf{x}^{(k+1)} := \mathbf{x}^{(k)} + \frac{b_Z - \langle \mathbf{x}^{(k)},\, \mathsf{A}^{(Z)} \rangle}{\|\mathsf{A}^{(Z)}\|_2^2} \mathsf{A}^{(Z)}$ *(in exact arithmetic), then*

$$\mathbb{E}_Z \left\| \mathbf{x}^{(k+1)} - \mathbf{x}_{LS} \right\|_2^2 \leq \left(1 - \frac{1}{\kappa_F^2(\mathsf{A})}\right) \left\| \mathbf{x}^{(k)} - \mathbf{x}_{LS} \right\|_2^2. \tag{3.3}$$

Theorem 3.4 follows by iterating Lemma 3.6 which implies that

$$\mathbb{E} \left\| \mathbf{x}^{(k+1)} - \mathbf{x}_{\mathrm{LS}} \right\|_2^2 \leq \left(1 - \frac{1}{\kappa_{\mathrm{F}}^2(\mathsf{A})}\right)^k \left\| \mathbf{x}^{(0)} - \mathbf{x}_{\mathrm{LS}} \right\|_2^2.$$

3.3. Randomized Kaczmarz applied to noisy linear systems. The analysis of Strohmer and Vershynin is based on the restrictive assumption that the linear system has a solution. Needell went a step further and analyzed the more general setting in which the linear system does not have any solution and A has full column rank [Nee10]. In this setting, it turns out that the randomized Kaczmarz algorithm computes an estimate vector that is within a fixed distance from the solution; the distance is proportional to the norm of the "noise vector" multiplied by $\kappa_{\mathrm{F}}^2(\mathsf{A})$ [Nee10]. The following theorem is a restatement of the main result in [Nee10] with two modifications: the full column rank assumption on the input matrix is dropped and the additive term γ of Theorem 2.1 in [Nee10] is improved to $\|\mathbf{w}\|_2^2 / \|\mathsf{A}\|_{\mathrm{F}}^2$. The only technical difference here from [Nee10] is that the full column rank assumption is not necessary, so we defer the proof to the appendix for completeness.

THEOREM 3.7. *Assume that the system* $\mathsf{A}\mathbf{x} = \mathbf{y}$ *has a solution for some* $\mathbf{y} \in \mathbb{R}^m$. *Denote it by* $\mathbf{x}^* := \mathsf{A}^\dagger \mathbf{y}$. *Let* $\hat{\mathbf{x}}^{(k)}$ *denote the* k*th iterate of the randomized Kaczmarz algorithm applied to the linear system* $\mathsf{A}\mathbf{x} = \mathbf{b}$ *with* $\mathbf{b} := \mathbf{y} + \mathbf{w}$ *for any fixed* $\mathbf{w} \in \mathbb{R}^m$, *i.e., run Algorithm* 2 *with input* $(\mathsf{A}, \mathbf{b})$. *In exact arithmetic, it follows that*

$$\mathbb{E}_{k-1} \left\| \hat{\mathbf{x}}^{(k)} - \mathbf{x}^* \right\|_2^2 \leq \left(1 - \frac{1}{\kappa_F^2(\mathsf{A})}\right) \left\| \hat{\mathbf{x}}^{(k-1)} - \mathbf{x}^* \right\|_2^2 + \frac{\|\mathbf{w}\|_2^2}{\|\mathsf{A}\|_{\mathrm{F}}^2}. \tag{3.4}$$

In particular,

$$\mathbb{E} \left\| \hat{\mathbf{x}}^{(k)} - \mathbf{x}^* \right\|_2^2 \leq \left(1 - \frac{1}{\kappa_F^2(\mathsf{A})}\right)^k \left\| \mathbf{x}^{(0)} - \mathbf{x}^* \right\|_2^2 + \frac{\|\mathbf{w}\|_2^2}{\sigma_{\min}^2}.$$

4. Randomized extended Kaczmarz. Given a least squares problem, Theorem 3.7 with $\mathbf{w} = \mathbf{b}_{\mathcal{R}(\mathsf{A})^\perp}$ (and $y = \mathbf{b}_{\mathcal{R}(\mathsf{A})}$) tells us that the randomized Kaczmarz algorithm works well for least squares problems whose least squares error is very close to zero, i.e., $\|\mathbf{w}\|_2 \approx 0$. Roughly speaking, in this case the randomized Kaczmarz algorithm approaches the minimum ℓ_2-norm least squares solution up to an additive error that depends on the distance between $\mathbf{b}$ and the column space of A.

In the present paper, the main observation is that it is possible to efficiently reduce the norm of the "noisy" part of $\mathbf{b}$, $\mathbf{b}_{\mathcal{R}(\mathsf{A})^\perp}$ (using Algorithm 1) and then apply the randomized Kaczmarz algorithm on a new linear system whose right-hand side vector is now arbitrarily close to the column space of A, i.e., $\mathsf{A}\mathbf{x} \approx \mathbf{b}_{\mathcal{R}(\mathsf{A})}$. This idea together with the observation that the least squares solution of the latter linear system is equal (in the limit) to the least squares solution of the original system (see Fact 3.1) implies a randomized algorithm for solving least squares.

Next we present the randomized extended Kaczmarz algorithm which is a specific combination of the randomized orthogonal projection algorithm together with the randomized Kaczmarz algorithm.

ALGORITHM 3. RANDOMIZED EXTENDED KACZMARZ (REK).

1: **procedure** (A, $\mathbf{b}$, ε) ▷ $\mathsf{A} \in \mathbb{R}^{m\times n}, \mathbf{b} \in \mathbb{R}^m$, $\varepsilon > 0$
2: Initialize $\mathbf{x}^{(0)} = \mathbf{0}$ and $\mathbf{z}^{(0)} = \mathbf{b}$
3: **for** $k = 0, 1, 2, \dots$ **do**
4: Pick $i_k \in [m]$ with probability $q_i := \left\|\mathsf{A}^{(i)}\right\|_2^2 / \|\mathsf{A}\|_{\mathrm{F}}^2, i \in [m]$
5: Pick $j_k \in [n]$ with probability $p_j := \left\|\mathsf{A}_{(j)}\right\|_2^2 / \|\mathsf{A}\|_{\mathrm{F}}^2,\ j \in [n]$
6: Set $\mathbf{z}^{(k+1)} = \mathbf{z}^{(k)} - \frac{\langle \mathsf{A}_{(j_k)},\, \mathbf{z}^{(k)}\rangle}{\left\|\mathsf{A}_{(j_k)}\right\|_2^2}\mathsf{A}_{(j_k)}$
7: Set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \frac{b_{i_k} - z_{i_k}^{(k)} - \langle \mathbf{x}^{(k)},\, \mathsf{A}^{(i_k)}\rangle}{\left\|\mathsf{A}^{(i_k)}\right\|_2^2}\mathsf{A}^{(i_k)}$
8: Check every $8\min(m, n)$ iterations and terminate if it holds:

$$\frac{\left\|\mathsf{A}\mathbf{x}^{(k)} - (\mathbf{b} - \mathbf{z}^{(k)})\right\|_2}{\|\mathsf{A}\|_{\mathrm{F}}\left\|\mathbf{x}^{(k)}\right\|_2} \le \varepsilon \quad \text{and} \quad \frac{\left\|\mathsf{A}^\top\mathbf{z}^{(k)}\right\|_2}{\|\mathsf{A}\|_{\mathrm{F}}^2\left\|\mathbf{x}^{(k)}\right\|_2} \le \varepsilon.$$

9: **end for**
10: Output $\mathbf{x}^{(k)}$
11: **end procedure**

4.1. The algorithm. We describe a randomized algorithm that converges in mean square to the minimum ℓ_2-norm solution vector $\mathbf{x}_{\mathrm{LS}}$ (Algorithm 3). The proposed algorithm consists of two components. The first component consisting of steps 5 and 6 is responsible for implicitly maintaining an approximation to $\mathbf{b}_{\mathcal{R}(\mathsf{A})}$ formed by $\mathbf{b} - \mathbf{z}^{(k)}$. The second component, consisting of steps 4 and 7, applies the randomized Kaczmarz algorithm with input A and the current approximation $\mathbf{b} - \mathbf{z}^{(k)}$ of $\mathbf{b}_{\mathcal{R}(\mathsf{A})}$, i.e., applies one step of the randomized Kaczmarz on the system $\mathsf{A}\mathbf{x} = \mathbf{b} - \mathbf{z}^{(k)}$. Since $\mathbf{b} - \mathbf{z}^{(k)}$ converges to $\mathbf{b}_{\mathcal{R}(\mathsf{A})}$, $\mathbf{x}^{(k)}$ will eventually converge to the minimum Euclidean norm solution of $\mathsf{A}\mathbf{x} = \mathbf{b}_{\mathcal{R}(\mathsf{A})}$ which equals $\mathbf{x}_{\mathrm{LS}} = \mathsf{A}^\dagger\mathbf{b}$ (see Fact 3.1).

The stopping criterion of step 8 is decided based on the following analysis. Assume that the termination criteria are met for some $k > 0$. Let $\mathbf{z}^{(k)} = \mathbf{b}_{\mathcal{R}(\mathsf{A})^\perp} + \mathbf{w}^{(k)}$ for some $\mathbf{w}^{(k)} \in \mathcal{R}(\mathsf{A})$ (which holds by the definition of $\mathbf{z}^{(k)}$). Then,

$$\left\|\mathsf{A}^\top\mathbf{z}^{(k)}\right\|_2 = \left\|\mathsf{A}^\top(\mathbf{b}_{\mathcal{R}(\mathsf{A})^\perp} + \mathbf{w}^{(k)})\right\|_2 = \left\|\mathsf{A}^\top\mathbf{w}\right\|_2 \ge \sigma_{\min}\left\|\mathbf{z}^{(k)} - \mathbf{b}_{\mathcal{R}(\mathsf{A})^\perp}\right\|_2.$$

By rearranging terms and using the second part of the termination criterion, it follows that $\|\mathbf{z}^{(k)} - \mathbf{b}_{\mathcal{R}(\mathsf{A})^\perp}\|_2 \leq \varepsilon \frac{\|\mathsf{A}\|_{\mathrm{F}}^2}{\sigma_{\min}} \|\mathbf{x}^{(k)}\|_2$. Now,

$$\begin{aligned}\left\|\mathsf{A}(\mathbf{x}^{(k)} - \mathbf{x}_{\mathrm{LS}})\right\|_2 &\leq \left\|\mathsf{A}\mathbf{x}^{(k)} - (\mathbf{b} - \mathbf{z}^{(k)})\right\|_2 + \left\|\mathbf{b} - \mathbf{z}^{(k)} - \mathsf{A}\mathbf{x}_{\mathrm{LS}}\right\|_2 \\ &\leq \varepsilon \left\|\mathsf{A}\right\|_{\mathrm{F}} \left\|\mathbf{x}^{(k)}\right\|_2 + \left\|\mathbf{b}_{\mathcal{R}(\mathsf{A})^\perp} - \mathbf{z}^{(k)}\right\|_2 \\ &\leq \varepsilon \left\|\mathsf{A}\right\|_{\mathrm{F}} \left\|\mathbf{x}^{(k)}\right\|_2 + \varepsilon \frac{\|\mathsf{A}\|_{\mathrm{F}}^2}{\sigma_{\min}} \left\|\mathbf{x}^{(k)}\right\|_2,\end{aligned}$$

where we used the triangle inequality, the first part of the termination rule together with $\mathbf{b}_{\mathcal{R}(\mathsf{A})} = \mathsf{A}\mathbf{x}_{\mathrm{LS}}$, and the above discussion. Now, since $\mathbf{x}^{(k)}, \mathbf{x}_{\mathrm{LS}} \in \mathcal{R}(\mathsf{A}^\top)$, it follows that

$$\frac{\left\|\mathbf{x}^{(k)} - \mathbf{x}_{\mathrm{LS}}\right\|_2}{\left\|\mathbf{x}^{(k)}\right\|_2} \leq \varepsilon \kappa_{\mathrm{F}}(\mathsf{A})(1 + \kappa_{\mathrm{F}}(\mathsf{A})) \tag{4.1}$$

using that $\|\mathsf{A}(\mathbf{x}^{(k)} - \mathbf{x}_{\mathrm{LS}})\|_2 \geq \sigma_{\min} \|\mathbf{x}^{(k)} - \mathbf{x}_{\mathrm{LS}}\|_2$. Equation (4.1) demonstrates that the forward error of REK after termination is bounded.

4.2. Rate of convergence. The following theorem bounds the expected rate of convergence of Algorithm 3.

THEOREM 4.1. *After $T > 1$ iterations, in exact arithmetic, Algorithm* 3 *with input* A *(possibly rank deficient) and* $\mathbf{b}$ *computes a vector* $\mathbf{x}^{(T)}$ *such that*

$$\mathbb{E}\left\|\mathbf{x}^{(T)} - \mathbf{x}_{LS}\right\|_2^2 \leq \left(1 - \frac{1}{\kappa_F^2(\mathsf{A})}\right)^{\lfloor T/2 \rfloor} \left(1 + 2\kappa^2\left(\mathsf{A}\right)\right) \left\|\mathbf{x}_{LS}\right\|_2^2.$$

Proof. For the sake of notation, set $\alpha = 1 - 1/\kappa_{\mathrm{F}}^2(\mathsf{A})$ and denote by $\mathbb{E}_k[\cdot] := \mathbb{E}[\cdot \mid i_0, j_0, i_1, j_1, \ldots, i_k, j_k]$, the conditional expectation with respect to the first k iterations of Algorithm 3. Observe that steps 5 and 6 are independent of steps 4 and 7 of Algorithm 3, so Theorem 3.2 implies that for every $l \geq 0$

$$\mathbb{E}\left\|\mathbf{z}^{(l)} - \mathbf{b}_{\mathcal{R}(\mathsf{A})^\perp}\right\|_2^2 \leq \alpha^l \left\|\mathbf{b}_{\mathcal{R}(\mathsf{A})}\right\|_2^2 \leq \left\|\mathbf{b}_{\mathcal{R}(\mathsf{A})}\right\|_2^2. \tag{4.2}$$

Fix a parameter $k^* := \lfloor T/2 \rfloor$. After the k^*th iteration of Algorithm 3, it follows from Theorem 3.7 (inequality (3.4)) that

$$\mathbb{E}_{(k^*-1)}\left\|\mathbf{x}^{(k^*)} - \mathbf{x}_{\mathrm{LS}}\right\|_2^2 \leq \alpha \left\|\mathbf{x}^{(k^*-1)} - \mathbf{x}_{\mathrm{LS}}\right\|_2^2 + \frac{\left\|\mathbf{b}_{\mathcal{R}(\mathsf{A})^\perp} - \mathbf{z}^{(k^*-1)}\right\|_2^2}{\|\mathsf{A}\|_{\mathrm{F}}^2}.$$

Indeed, the randomized Kaczmarz algorithm is executed with input $(\mathsf{A}, \mathbf{b} - \mathbf{z}^{(k^*-1)})$ and current estimate vector $\mathbf{x}^{(k^*-1)}$. Set $\mathbf{y} = \mathbf{b}_{\mathcal{R}(\mathsf{A})}$ and $\mathbf{w} = \mathbf{b}_{\mathcal{R}(\mathsf{A})^\perp} - \mathbf{z}^{(k^*-1)}$ in Theorem 3.7 and recall that $\mathbf{x}_{\mathrm{LS}} = \mathsf{A}^\dagger \mathbf{b} = \mathsf{A}^\dagger \mathbf{b}_{\mathcal{R}(\mathsf{A})} = \mathsf{A}^\dagger \mathbf{y}$.

Now, averaging the above inequality over the random variables $i_1, j_1, i_2, j_2, \ldots, i_{k^*-1}, j_{k^*-1}$ and using linearity of expectation, it holds that

(4.3)

$$\begin{aligned}
&\mathbb{E}\left\|\mathbf{x}^{(k^*)} - \mathbf{x}_{\text{LS}}\right\|_2^2 \\
&\quad\leq \alpha\,\mathbb{E}\left\|\mathbf{x}^{(k^*-1)} - \mathbf{x}_{\text{LS}}\right\|_2^2 + \frac{\mathbb{E}\left\|\mathbf{b}_{\mathcal{R}(\mathsf{A})^\perp} - \mathbf{z}^{(k^*-1)}\right\|_2^2}{\|\mathsf{A}\|_{\text{F}}^2} \\
&\quad\leq \alpha\,\mathbb{E}\left\|\mathbf{x}^{(k^*-1)} - \mathbf{x}_{\text{LS}}\right\|_2^2 + \frac{\left\|\mathbf{b}_{\mathcal{R}(\mathsf{A})}\right\|_2^2}{\|\mathsf{A}\|_{\text{F}}^2} \quad \text{by inequality (4.2)} \\
&\quad\leq \cdots \leq \alpha^{k^*}\left\|\mathbf{x}^{(0)} - \mathbf{x}_{\text{LS}}\right\|_2^2 + \sum_{l=0}^{k^*-2}\alpha^l\frac{\left\|\mathbf{b}_{\mathcal{R}(\mathsf{A})}\right\|_2^2}{\|\mathsf{A}\|_{\text{F}}^2} \quad \text{(repeat the above } k^*-1 \text{ times)} \\
&\quad\leq \|\mathbf{x}_{\text{LS}}\|_2^2 + \sum_{l=0}^{\infty}\alpha^l\frac{\left\|\mathbf{b}_{\mathcal{R}(\mathsf{A})}\right\|_2^2}{\|\mathsf{A}\|_{\text{F}}^2} \quad \text{since } \alpha<1 \text{ and } \mathbf{x}^{(0)} = \mathbf{0}.
\end{aligned}$$

Simplifying the right-hand side using the fact that $\sum_{l=0}^{\infty}\alpha^l = \frac{1}{1-\alpha} = \kappa_{\text{F}}^2(\mathsf{A})$, it follows that

$$\mathbb{E}\left\|\mathbf{x}^{(k^*)} - \mathbf{x}_{\text{LS}}\right\|_2^2 \leq \|\mathbf{x}_{\text{LS}}\|_2^2 + \left\|\mathbf{b}_{\mathcal{R}(\mathsf{A})}\right\|_2^2/\sigma_{\min}^2. \tag{4.4}$$

Moreover, observe that for every $l \geq 0$

$$\mathbb{E}\left\|\mathbf{b}_{\mathcal{R}(\mathsf{A})^\perp} - \mathbf{z}^{(l+k^*)}\right\|_2^2 \leq \alpha^{l+k^*}\left\|\mathbf{b}_{\mathcal{R}(\mathsf{A})}\right\|_2^2 \leq \alpha^{k^*}\left\|\mathbf{b}_{\mathcal{R}(\mathsf{A})}\right\|_2^2. \tag{4.5}$$

Now for any $k > 0$, similar considerations as inequality (4.3) imply that

$$\begin{aligned}
&\mathbb{E}\left\|\mathbf{x}^{(k+k^*)} - \mathbf{x}_{\text{LS}}\right\|_2^2 \\
&\quad\leq \alpha\,\mathbb{E}\left\|\mathbf{x}^{(k+k^*-1)} - \mathbf{x}_{\text{LS}}\right\|_2^2 + \frac{\mathbb{E}\left\|\mathbf{b}_{\mathcal{R}(\mathsf{A})^\perp} - \mathbf{z}^{(k-1+k^*)}\right\|_2^2}{\|\mathsf{A}\|_{\text{F}}^2} \\
&\quad\leq \cdots \leq \alpha^k\,\mathbb{E}\left\|\mathbf{x}^{(k^*)} - \mathbf{x}_{\text{LS}}\right\|_2^2 + \sum_{l=0}^{k-1}\alpha^{(k-1)-l}\frac{\mathbb{E}\left\|\mathbf{b}_{\mathcal{R}(\mathsf{A})^\perp} - \mathbf{z}^{(l+k^*)}\right\|_2^2}{\|\mathsf{A}\|_{\text{F}}^2} \quad \text{(by induction)} \\
&\quad\leq \alpha^k\,\mathbb{E}\left\|\mathbf{x}^{(k^*)} - \mathbf{x}_{\text{LS}}\right\|_2^2 + \frac{\alpha^{k^*}\left\|\mathbf{b}_{\mathcal{R}(\mathsf{A})}\right\|_2^2}{\|\mathsf{A}\|_{\text{F}}^2}\sum_{l=0}^{k-1}\alpha^l \quad \text{(by inequality (4.5))} \\
&\quad\leq \alpha^k\left(\|\mathbf{x}_{\text{LS}}\|_2^2 + \left\|\mathbf{b}_{\mathcal{R}(\mathsf{A})}\right\|_2^2/\sigma_{\min}^2\right) + \alpha^{k^*}\left\|\mathbf{b}_{\mathcal{R}(\mathsf{A})}\right\|_2^2/\sigma_{\min}^2 \quad \text{(by inequality (4.4))} \\
&\quad= \alpha^k\|\mathbf{x}_{\text{LS}}\|_2^2 + (\alpha^k + \alpha^{k^*})\left\|\mathbf{b}_{\mathcal{R}(\mathsf{A})}\right\|_2^2/\sigma_{\min}^2 \\
&\quad\leq \alpha^k\|\mathbf{x}_{\text{LS}}\|_2^2 + (\alpha^k + \alpha^{k^*})\kappa^2(\mathsf{A})\|\mathbf{x}_{\text{LS}}\|_2^2 \quad \text{since } \left\|\mathbf{b}_{\mathcal{R}(\mathsf{A})}\right\|_2 \leq \sigma_{\max}\|\mathbf{x}_{\text{LS}}\|_2 \\
&\quad\leq \alpha^{k^*}(1 + 2\kappa^2(\mathsf{A}))\|\mathbf{x}_{\text{LS}}\|_2^2.
\end{aligned}$$

To derive the last inequality, consider two cases. If T is even, set $k = k^*$, otherwise set $k = k^* + 1$. In both cases, $(\alpha^k + \alpha^{k^*}) \leq 2\alpha^{k^*}$. ∎

4.3. Theoretical bounds on time complexity. In this section, we discuss the running time complexity of the REK (Algorithm 3). Recall that REK is a Las Vegas randomized algorithm, i.e., the algorithm always outputs an "approximately correct" least squares estimate (satisfying (4.1)) and its running time is a random variable. Given any fixed accuracy parameter $\varepsilon > 0$ and any fixed failure probability $0 < \delta < 1$ we bound the number of iterations required by the algorithm to terminate with probability at least $1 - \delta$.

LEMMA 4.2. *Fix an accuracy parameter* $0 < \varepsilon < 2$ *and failure probability* $0 < \delta < 1$. *In exact arithmetic, Algorithm* 3 *terminates after at most*

$$T^* := 2\kappa_F^2(\mathsf{A}) \ln\left(\frac{32(1+2\kappa^2(\mathsf{A}))}{\delta\varepsilon^2}\right)$$

iterations with probability at least $1 - \delta$.

Proof. Denote $\alpha := 1 - 1/\kappa_{\mathrm{F}}^2(\mathsf{A})$ for notational convenience. It suffices to prove that with probability at least $1 - \delta$ the conditions of step 8 of Algorithm 3 are met. Instead of proving this, we will show that

1. with probability at least $1 - \delta/2$, $\|(\mathbf{b} - \mathbf{z}^{(T^*)}) - \mathbf{b}_{\mathcal{R}(\mathsf{A})}\|_2 \le \varepsilon \|\mathbf{b}_{\mathcal{R}(\mathsf{A})}\|_2/4$;
2. with probability at least $1 - \delta/2$, $\|\mathbf{x}^{(T^*)} - \mathbf{x}_{\mathrm{LS}}\|_2 \le \varepsilon \|\mathbf{x}_{\mathrm{LS}}\|_2/4$.

Later we prove that items (1) and (2) imply the lemma. First we prove item (1). By the definition of the algorithm,

$$\begin{aligned}
&\mathbb{P}\left(\left\|(\mathbf{b} - \mathbf{z}^{(T^*)}) - \mathbf{b}_{\mathcal{R}(\mathsf{A})}\right\|_2 \ge \varepsilon \left\|\mathbf{b}_{\mathcal{R}(\mathsf{A})}\right\|_2/4\right) \\
&\quad = \mathbb{P}\left(\left\|\mathbf{b}_{\mathcal{R}(\mathsf{A})^\perp} - \mathbf{z}^{(T^*)}\right\|_2^2 \ge \varepsilon^2 \left\|\mathbf{b}_{\mathcal{R}(\mathsf{A})}\right\|_2^2/16\right) \\
&\quad \le \frac{16\,\mathbb{E}\left\|\mathbf{z}^{(T^*)} - \mathbf{b}_{\mathcal{R}(\mathsf{A})^\perp}\right\|_2^2}{\varepsilon^2 \left\|\mathbf{b}_{\mathcal{R}(\mathsf{A})}\right\|_2^2} \\
&\quad \le 16\alpha^{T^*}/\varepsilon^2 \le \delta/2;
\end{aligned}$$

the first equality follows since $\mathbf{b} - \mathbf{b}_{\mathcal{R}(\mathsf{A})} = \mathbf{b}_{\mathcal{R}(\mathsf{A})^\perp}$, the second inequality is Markov's inequality, the third inequality follows by Theorem 3.2, and the last inequality follows since $T^* \ge \kappa_{\mathrm{F}}^2(\mathsf{A}) \ln(\frac{32}{\delta\varepsilon^2})$.

Now, we prove item (2):

$$\begin{aligned}
\mathbb{P}\left(\left\|\mathbf{x}^{(T^*)} - \mathbf{x}_{\mathrm{LS}}\right\|_2 \le \varepsilon \left\|\mathbf{x}_{\mathrm{LS}}\right\|_2/4\right) &\le \frac{16\,\mathbb{E}\left\|\mathbf{x}^{(T^*)} - \mathbf{x}_{\mathrm{LS}}\right\|_2^2}{\varepsilon^2 \left\|\mathbf{x}_{\mathrm{LS}}\right\|_2^2} \\
&\le 16\alpha^{\lfloor T^*/2 \rfloor}(1 + 2\kappa^2(\mathsf{A}))/\varepsilon^2 \le \delta/2;
\end{aligned}$$

the first inequality is Markov's inequality, the second inequality follows by Theorem 4.1, and the last inequality follows because $T^* \ge 2\kappa_{\mathrm{F}}^2(\mathsf{A}) \ln(\frac{32(1+2\kappa^2(\mathsf{A}))}{\delta\varepsilon^2})$.

A union bound on the complement of the above two events (items (1) and (2)) implies that both events happen with probability at least $1 - \delta$. Now we show that conditioning on items (1) and (2), it follows that REK terminates after T^* iterations with probability at least $1 - \delta$, i.e.,

$$\frac{\left\|\mathsf{A}\mathbf{x}^{(T^*)} - (\mathbf{b} - \mathbf{z}^{(T^*)})\right\|_2}{\|\mathsf{A}\|_{\mathrm{F}} \left\|\mathbf{x}^{(T^*)}\right\|_2} \le \varepsilon \quad \text{and} \quad \frac{\left\|\mathsf{A}^\top \mathbf{z}^{(k)}\right\|_2}{\|\mathsf{A}\|_{\mathrm{F}}^2 \left\|\mathbf{x}^{(k)}\right\|_2} \le \varepsilon.$$

We start with the first condition. Using the triangle inequality and item 2, it follows that

$$\left\|\mathbf{x}^{(T^*)}\right\|_2 \geq \|\mathbf{x}_{\text{LS}}\|_2 - \left\|\mathbf{x}_{\text{LS}} - \mathbf{x}^{(T^*)}\right\|_2 \geq (1-\varepsilon/4)\,\|\mathbf{x}_{\text{LS}}\|_2\,. \tag{4.6}$$

Now,

$$\begin{aligned}
\left\|\mathsf{A}\mathbf{x}^{(T^*)} - (\mathbf{b} - \mathbf{z}^{(T^*)})\right\|_2 &\leq \left\|\mathsf{A}\mathbf{x}^{(T^*)} - \mathbf{b}_{\mathcal{R}(\mathsf{A})}\right\|_2 + \left\|(\mathbf{b} - \mathbf{z}^{(T^*)}) - \mathbf{b}_{\mathcal{R}(\mathsf{A})}\right\|_2 \\
&\leq \left\|\mathsf{A}(\mathbf{x}^{(T^*)} - \mathbf{x}_{\text{LS}})\right\|_2 + \varepsilon\left\|\mathbf{b}_{\mathcal{R}(\mathsf{A})}\right\|_2/4 \\
&\leq \sigma_{\max}\left\|\mathbf{x}^{(T^*)} - \mathbf{x}_{\text{LS}}\right\|_2 + \varepsilon\,\|\mathsf{A}\mathbf{x}_{\text{LS}}\|_2/4 \\
&\leq \varepsilon\sigma_{\max}\,\|\mathbf{x}_{\text{LS}}\|_2/2 \\
&\leq \frac{\varepsilon/2}{1-\varepsilon/4}\left\|\mathbf{x}^{(T^*)}\right\|_2 \leq \varepsilon\left\|\mathbf{x}^{(T^*)}\right\|_2,
\end{aligned}$$

where the first inequality is the triangle inequality, the second inequality follows by item 1 and $\mathbf{b}_{\mathcal{R}(\mathsf{A})} = \mathsf{A}\mathbf{x}_{\text{LS}}$, the third and forth inequalities follow by item 2, the fifth inequality holds by inequality (4.6), and the last inequality follows since $\varepsilon < 2$. The second condition follows since

$$\begin{aligned}
\left\|\mathsf{A}^\top\mathbf{z}^{(T^*)}\right\|_2 &= \left\|\mathsf{A}^\top(\mathbf{b}_{\mathcal{R}(\mathsf{A})^\perp} - \mathbf{z}^{(T^*)})\right\|_2 \leq \sigma_{\max}\left\|\mathbf{b}_{\mathcal{R}(\mathsf{A})^\perp} - \mathbf{z}^{(T^*)}\right\|_2 \\
&\leq \varepsilon\sigma_{\max}\left\|\mathbf{b}_{\mathcal{R}(\mathsf{A})}\right\|_2/4 \leq \varepsilon\sigma_{\max}^2\,\|\mathbf{x}_{\text{LS}}\|_2/4 \\
&\leq \frac{\varepsilon/4}{1-\varepsilon/4}\sigma_{\max}^2\left\|\mathbf{x}^{(T^*)}\right\|_2 \leq \varepsilon\,\|\mathsf{A}\|_{\text{F}}^2\left\|\mathbf{x}^{(T^*)}\right\|_2;
\end{aligned}$$

the first equation follows by orthogonality, the second inequality assuming item (2), the third inequality follows since $\mathbf{b}_{\mathcal{R}(\mathsf{A})} = \mathsf{A}\mathbf{x}_{\text{LS}}$, the forth inequality follows by (4.6), and the final inequality since $\varepsilon < 2$. ∎

Lemma 4.2 bounds the number of iterations with probability at least $1-\delta$; next we bound the total number of arithmetic operations in the worst case (4.7) and in expectation (4.8). Let's calculate the computational cost of REK in terms of floating-point operations (flops) per iteration. For the sake of simplicity, we ignore the additional (negligible) computational overhead required to perform the sampling operations (see section 5 for more details) and checking for convergence.

Each iteration of Algorithm 3 requires four level-1 BLAS operations (two *DDOT* operations of size m and n, respectively, and two *DAXPY* operations of size n and m, respectively) and an additional four flops; in total, $4(m+n)+2$ flops per iteration.

Therefore by Lemma 4.2, with probability at least $1-\delta$, REK requires at most

$$5(m+n)\cdot T^* \leq 10(m+n)\mathbf{rank}\,(\mathsf{A})\,\kappa^2\,(\mathsf{A})\ln\left(\frac{32(1+2\kappa^2\,(\mathsf{A}))}{\delta\varepsilon^2}\right) \tag{4.7}$$

arithmetic operations (using that $\kappa_{\text{F}}^2(\mathsf{A}) \leq \mathbf{rank}\,(\mathsf{A})\,\kappa^2\,(\mathsf{A})$).

Next, we bound the *expected running time* of REK for achieving the above guarantees for any fixed ε and δ. Obviously, the expected running time is at most the quantity in (4.7). However, as we will see shortly the expected running time is proportional to $\mathbf{nnz}\,(\mathsf{A})\,\kappa^2\,(\mathsf{A})$ instead of $(m+n)\mathbf{rank}\,(\mathsf{A})\,\kappa^2\,(\mathsf{A})$.

Exploiting the (possible) sparsity of A, we first show that each iteration of Algorithm 3 requires at most $5(\text{C}_{\text{avg}} + \text{R}_{\text{avg}})$ operations in expectation. For simplicity of presentation, we assume that we have stored A in compressed column sparse format *and* compressed row sparse format [BBC+87].

Indeed, fix any $i_k \in [m]$ and $j_k \in [n]$ at some iteration k of Algorithm 3. Since A is stored both in compressed column and compressed sparse format, steps 7 and 8 can be implemented in $5\mathbf{nnz}(\mathsf{A}_{(j_k)})$ and $5\mathbf{nnz}(\mathsf{A}^{(i_k)})$, respectively.

By the linearity of expectation and the definitions of $\mathrm{C}_{\mathrm{avg}}$ and $\mathrm{R}_{\mathrm{avg}}$, the expected running time after T^* iterations is at most $5T^*(\mathrm{C}_{\mathrm{avg}} + \mathrm{R}_{\mathrm{avg}})$. It holds that (recall that $p_j = \|\mathsf{A}_{(j)}\|_2^2 / \|\mathsf{A}\|_{\mathrm{F}}^2$)

$$\begin{aligned}
\mathrm{C}_{\mathrm{avg}} T^* &= \frac{2}{\|\mathsf{A}\|_{\mathrm{F}}^2} \left(\sum_{j=1}^{n} \left\|\mathsf{A}_{(j)}\right\|_2^2 \mathbf{nnz}\left(\mathsf{A}_{(j)}\right) \right) \frac{\|\mathsf{A}\|_{\mathrm{F}}^2}{\sigma_{\min}^2} \ln\left(\frac{32(1+2\kappa^2(\mathsf{A}))}{\delta\varepsilon^2} \right) \\
&= 2\frac{\sum_{j=1}^{n} \left\|\mathsf{A}_{(j)}\right\|_2^2 \mathbf{nnz}\left(\mathsf{A}_{(j)}\right)}{\sigma_{\min}^2} \ln\left(\frac{32(1+2\kappa^2(\mathsf{A}))}{\delta\varepsilon^2} \right) \\
&\leq 2\sum_{j=1}^{n} \mathbf{nnz}\left(\mathsf{A}_{(j)}\right) \frac{\max_{j\in[n]} \left\|\mathsf{A}_{(j)}\right\|_2^2}{\sigma_{\min}^2} \ln\left(\frac{32(1+2\kappa^2(\mathsf{A}))}{\delta\varepsilon^2} \right) \\
&\leq 2\mathbf{nnz}(\mathsf{A})\,\kappa^2(\mathsf{A}) \ln\left(\frac{32(1+2\kappa^2(\mathsf{A}))}{\delta\varepsilon^2} \right)
\end{aligned}$$

using the definition of $\mathrm{C}_{\mathrm{avg}}$ and T^* in the first equality and the fact that $\max_{j\in[n]} \|\mathsf{A}_{(j)}\|_2^2 \leq \sigma_{\max}^2$ and $\sum_{j=1}^{n} \mathbf{nnz}(\mathsf{A}_{(j)}) = \mathbf{nnz}(\mathsf{A})$ in the first and second inequalities. A similar argument shows that $\mathrm{R}_{\mathrm{avg}} T^* \leq 2\mathbf{nnz}(\mathsf{A})\kappa^2(\mathsf{A}) \ln(\frac{32(1+2\kappa^2(\mathsf{A}))}{\delta\varepsilon^2})$ using the inequality $\max_{i\in[m]} \|\mathsf{A}^{(i)}\|_2^2 \leq \sigma_{\max}^2$.

Hence by Lemma 4.2, with probability at least $1-\delta$, the expected number of arithmetic operations of REK is at most

$$20\mathbf{nnz}(\mathsf{A})\,\kappa^2(\mathsf{A}) \ln\left(\frac{32(1+2\kappa^2(\mathsf{A}))}{\delta\varepsilon^2} \right). \tag{4.8}$$

In other words, the expected running time is much lower than the worst case displayed in (4.7) and is proportional to $\mathbf{nnz}(\mathsf{A})$ times the squared condition number of A as advertised in the abstract.

5. Implementation and experimental results.

5.1. Implementation. The proposed algorithm has been implemented entirely in C. We provide three implementations of Algorithm 3: `REK-C`, `REK-BLAS`, and `REK-BLAS-PRECOND`. `REK-C` corresponds to a direct translation of Algorithm 3 to C code. `REK-BLAS` is an implementation of REK with two additional technical features. First, `REK-BLAS` uses level-1 BLAS routines for all operations of Algorithm 3 and second `REK-BLAS` additionally stores explicitly the transpose of A for more efficient memory access of both the rows and columns of A using BLAS. `REK-BLAS-PRECOND` is an implementation of `REK-BLAS` that additionally supports upper triangular preconditioning; we used Blendenpik's preconditioning code to ensure a fair comparison with Blendenpik (see section 5.2). In the implementations of `REK-C` and `REK-BLAS` we check for convergence every $8\min(m,n)$ iterations. We experimentally observed that checking for convergence should be performed after $\mathcal{O}(\min(m,n))$ iterations; the constant 8 is fine-tuned for our experiments and might be suboptimal in general.

Moreover, all implementations include efficient code that handles sparse input matrices using the compressed column (and row) sparse matrix format [BBC+87].

Sampling from nonuniform distributions. The sampling operations of Algorithm 3 (steps 4 and 5) are implemented using the so-called "alias method" for generating samples from any given discrete distribution [Wal77, Vos91]. The alias method, assuming

access to a uniform random variable on $[0, 1]$ in constant time and linear time preprocessing, generates one sample of the given distribution in constant time [Vos91]. We use an implementation of Smith that is described in [Smi02] and C's *drand*48() to get uniform samples from $[0, 1]$.

5.2. Experimental results. We report our experimental results in this section. We compared the REK (`REK-C`, `REK-BLAS`, `REK-BLAS-PRECOND`) algorithm to LAPACK's `DGELSY` and `DGELSD` least squares solvers, Blendenpik[2] (version 1.3, [AMT10]) and the backslash operator in MATLAB. `LSRN` [MSM11] did not perform well under a setup in which no parallelization is allowed as the one used here, so we do not include `LSRN`'s performance. `DGELSY` uses QR factorization with pivoting and `DGELSD` uses the SVD. We use MATLAB version 7.9.0.529 (R2009b). In addition, we use the MATLAB packages BLAS and LAPACK and we call LAPACK's functions from MATLAB using its CMEX technology which allows us to measure only LAPACK's elapsed time. We should highlight that MATLAB is used as a scripting language and no MATLAB-related overheads have been taken into consideration. Blendenpik requires the FFTW library;[3] we used FFTW-3.3.3. To match the accuracy of LAPACK's direct solvers, we fixed ε in Algorithm 3 to be 10e-14. Moreover, during our experiments we ensured that the residual error of all the competing algorithms were of the same order of magnitude.

We used a Pentium(R) Dual-Core E5300 (2.60 GHz) equipped with 5 GB of RAM and compiled our source code using GCC-4.7.2 under Linux operating system. All running times displayed below are measured using the *ftime* Linux system call by taking the average of the running time of 10 independent executions.

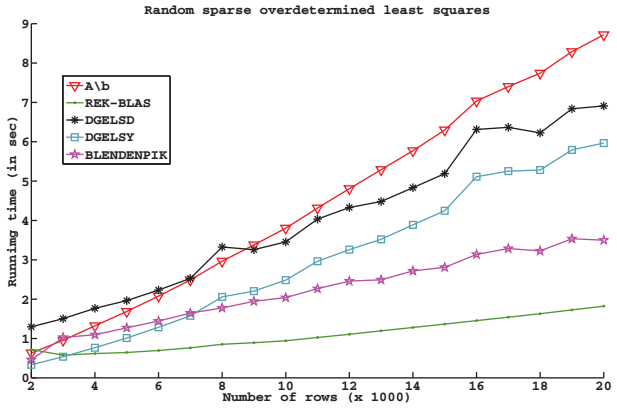
We tested our algorithm under three different distributions of random input matrices (sparse, dense, and ill-conditioned) using strongly rectangular settings of least squares instances. Numerical experiments indicate that our algorithm does not converge to $\mathbf{x}_{\text{LS}}$ in practice whenever A is rank deficient, so we do not include any relevant experiments. In all cases we normalized the column norms of the input matrices to unity and generated the right-hand side vector $\mathbf{b}$ having i.i.d. Gaussian entries of variance one, which ensures that $\mathbf{b} \notin \mathcal{R}(\mathsf{A})$ almost surely for the case of overdetermined linear systems.

Sparse least squares. We tested our algorithm in the overdetermined setting of random sparse $m \times n$ matrices with $n = 800$ and $m = 2000, 3000, \ldots, 20000$ and density 0.25. We also tested `REK-BLAS` in the underdetermined case where $m = 800$ and $n = 2000, 3000, \ldots, 20000$. In both cases, the density of the sparse matrices was set to 0.25 (for even sparser matrices `REK-BLAS` performed even better compared to all other mentioned methods). To generate these sparse matrix ensembles, we used the MATLAB *sprandn* function with variance one. The results are depicted in Figure 5.1. Both plots demonstrate that `REK-BLAS` is superior on both the underdetermined (Figure 5.1(b)) and overdetermined cases (Figure 5.1(a)). It is interesting that `REK-BLAS` performs well in the underdetermined case.

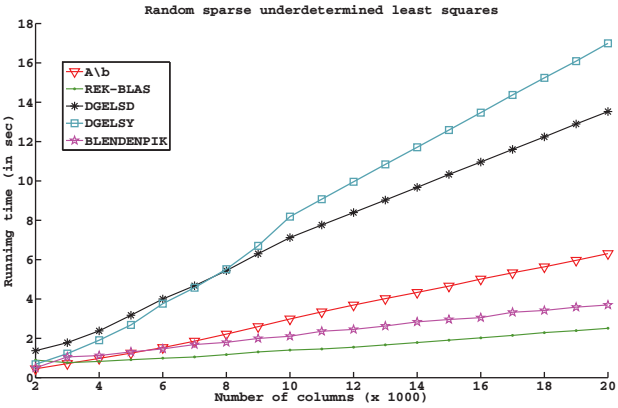
Dense and well-conditioned least squares. In this scenario, we used random overdetermined dense $m \times n$ matrices with many more rows than columns, i.e., we set $n = 500$ and $m = 1000, 2000, \ldots, 20000$. We also tested `REK-BLAS` on the underdetermined case where $m = 500$ and $n = 1000, 2000, \ldots, 20000$. We generated this set of matrices using the MATLAB *randn* function with variance ten. We depict the results in

[2] Available at http://www.mathworks.com/matlabcentral/fileexchange/25241-blendenpik (accessed on 10 December 2012). Blendenpik's default settings were used.

[3] http://www.fftw.org/



(a) Random sparse matrices having 800 columns and density 0.25.



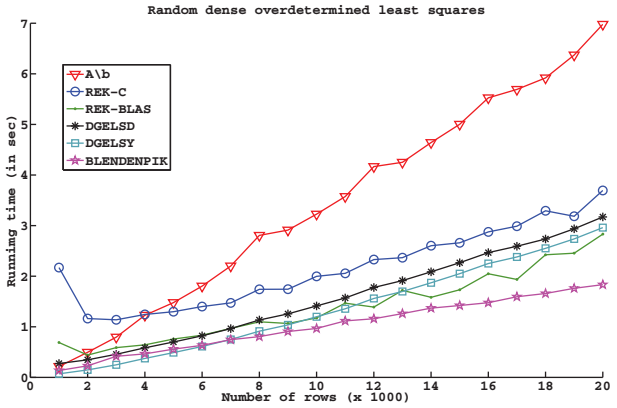
(b) Random sparse matrices having 800 rows and density 0.25.

FIG. 5.1. *Figures depict the running time (in seconds) versus increasing number of rows/columns (scaled by* 1000*) for the case of random sparse overdetermined* (a) *and underdetermined* (b) *least squares problems.*

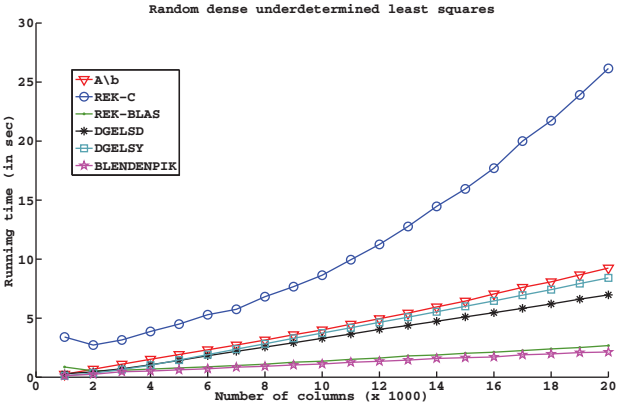
Figure 5.2. In the overdetermined case (Figure 5.2(a)), `REK-BLAS` is marginally superior compared to LAPACK's routines whereas `REK-C` (as a naive implementation of Algorithm 3) is inferior. Blendenpik is the winner in this case. Interestingly, `REK-BLAS` almost matches the performance of Blendenpik in the underdetermined case; see Figure 5.2(b).

Dense and ill-conditioned least squares. In this setting, we tested all algorithms under a particular case of random ill-conditioned dense matrices with $n = 500$ and $m = 1000, 2000, \ldots, 20000$. Namely, we used Higham's *randSVD* function for generating these matrices [Hig89, Hig96]. More precisely, we set the condition number of these matrices to be $10e6$, set the top singular value to one, and the rest to 10e-6. The results are displayed in Figure 5.3. Unfortunately, in the ill-conditioned setting `REK-BLAS-PRECOND` is inferior compared to LAPACK's routines and Blendenpik. We also verified the results of [AMT10] that Blendenpik is superior compared to LAPACK's least squares solvers in this setting.

Moreover, we should highlight that the REK algorithm did not perform well in the setting of random low-rank overdetermined linear systems, although Theorem 4.1



(a) Random dense matrices having 500 columns.



(b) Random dense matrices having 500 rows.

FIG. 5.2. *Figures depict the running time (in seconds) versus increasing number of rows/columns (scaled by* 1000*) for the case of random dense overdetermined* (a) *and underdetermined* (b) *least squares problems.*

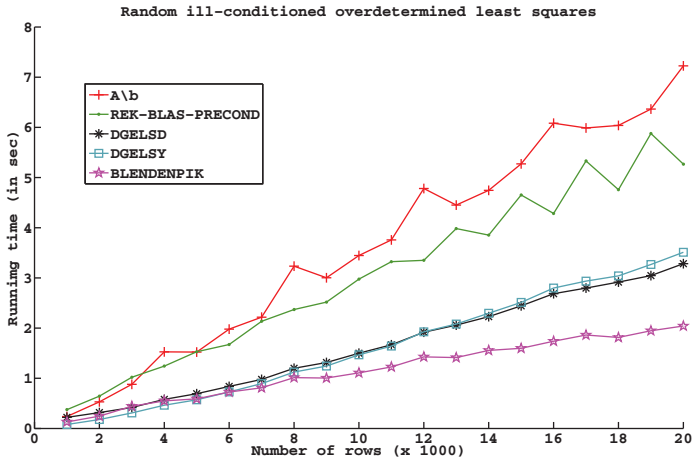


FIG. 5.3. *Running time (in seconds) versus number of rows (scaled by* 1000*) for the case of random dense and ill-conditioned input matrices having* 500 *columns and condition number* 10*e*6.

TABLE 5.1

Real-world problems and corresponding running times in seconds. All matrices are from the University of Florida Sparse Matrix Collection [DH11]. *OOM corresponds to "out of memory error message."*

Matrix properties					Runtime in seconds		
Matrix	m	n	condest(A)	**nnz** (A)	A**b**	DGELSY	REK-BLAS
GL7d13	47271	8899	Inf	356232	235	OOM	7.49
relat8	345688	12347	Inf	1334038	OOM	OOM	48.25
ESOC	327062	37830	Inf	6019939	153	OOM	>3600
12month1	12471	872622	Inf	22624727	3.3	OOM	>3600
stat96v3	33841	1113780	1.8e+08	3317736	1.6	OOM	2120.42
lp_nug30	52260	379350	N/A	1567800	OOM	OOM	3.678

provides bounds on the performance of the REK algorithm for rank-deficient linear systems *in exact arithmetic*.

Real-world least squares problems. Finally, we test all algorithms on a few real-world large scale problems. All problem instances (matrices) depicted in Table 5.1 are from the University of Florida Sparse Matrix Collection [DH11] which is publicly available online.[4]

GL7d13 is a particular differential of the Voronoi complex of various perfect forms; relat8 originated from a combinatorial problem; ESOC is a least squares problem which originated from orbit estimates; the 12month1 matrix contains the users from digg.com within a 12 month period (each row is a user and each column is a web page); stat96v3 and lp_nug30 originated from linear programming problems.

Blendenpik did not return any results on all of the above examples (runs out of memory or terminates unexpectedly) and DGELSY runs out of memory. The REK-BLAS algorithm is superior compared to the backslash operator in MATLAB on the overdetermined test cases of GL7d13 and relat8 and the underdetermined case of lp_nug30. On the other hand, the backslash operator in MATLAB is the winner on the cases of ESOC, 12month1, and stat96v3 with an unpredictable performance; this backslash operator probably relies on analyzing the sparsity patterns of the input matrix; see also [MSM11].

6. Appendix. We present the proof of known facts from previous works for completeness.

Proof of Lemma 3.5. It suffices to show that $\langle \mathbf{x}^{(k+1)} - \mathbf{x}_{\mathrm{LS}},\ \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)} \rangle = 0$. For notational convenience, let $\alpha_i := \frac{b_i - \langle \mathbf{x}^{(k)},\ \mathsf{A}^{(i)} \rangle}{\|\mathsf{A}^{(i)}\|_2^2}$ for every $i \in [m]$. Assume that $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_{i_k} \mathsf{A}^{(i_k)}$ for some arbitrary $i_k \in [m]$. Then,

$$\begin{aligned} \left\langle \mathbf{x}^{(k+1)} - \mathbf{x}_{\mathrm{LS}},\ \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)} \right\rangle &= \left\langle \mathbf{x}^{(k+1)} - \mathbf{x}_{\mathrm{LS}},\ \alpha_{i_k} \mathsf{A}^{(i_k)} \right\rangle \\ &= \alpha_{i_k} \left(\left\langle \mathbf{x}^{(k+1)},\ \mathsf{A}^{(i_k)} \right\rangle - b_{i_k} \right) \end{aligned}$$

using the definition of $\mathbf{x}^{(k+1)}$, and the fact that $\langle \mathbf{x}_{\mathrm{LS}},\ \mathsf{A}^{(i_k)} \rangle = b_{i_k}$ since $\mathbf{x}_{\mathrm{LS}}$ is a solution to $\mathsf{A}\mathbf{x} = \mathbf{b}$. Now, by the definition of α_{i_k}, $\langle \mathbf{x}^{(k+1)},\ \mathsf{A}^{(i_k)} \rangle = \langle \mathbf{x}^{(k)},\ \mathsf{A}^{(i_k)} \rangle + \alpha_{i_k} \|\mathsf{A}^{(i_k)}\|_2^2 = \langle \mathbf{x}^{(k)},\ \mathsf{A}^{(i_k)} \rangle + b_{i_k} - \langle \mathbf{x}^{(k)},\ \mathsf{A}^{(i_k)} \rangle = b_{i_k}$. ∎

[4] http://www.cise.ufl.edu/research/sparse/matrices/ (Accessed in February 2013).

Proof of Lemma 3.6. In light of Lemma 3.5, it suffices to show that $\mathbb{E}_Z \|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|_2^2 \geq \frac{1}{\kappa_{\mathrm{F}}^2(\mathsf{A})}\|\mathbf{x}^{(k)} - \mathbf{x}_{\mathrm{LS}}\|_2^2$. By the definition of $\mathbf{x}^{(k+1)}$, it follows that

$$\begin{aligned}\mathbb{E}_Z \left\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right\|_2^2 &= \mathbb{E}_Z \left[\left(\frac{b_Z - \left\langle \mathbf{x}^{(k)},\ \mathsf{A}^{(Z)}\right\rangle}{\left\|\mathsf{A}^{(Z)}\right\|_2^2}\right)^2 \left\|\mathsf{A}^{(Z)}\right\|_2^2\right] \\ &= \mathbb{E}_Z \frac{\left\langle \mathbf{x}_{\mathrm{LS}} - \mathbf{x}^{(k)},\ \mathsf{A}^{(Z)}\right\rangle^2}{\left\|\mathsf{A}^{(Z)}\right\|_2^2} \\ &= \sum_{i=1}^m \frac{\left\langle \mathbf{x}_{\mathrm{LS}} - \mathbf{x}^{(k)},\ \mathsf{A}^{(i)}\right\rangle^2}{\|\mathsf{A}\|_{\mathrm{F}}^2} = \frac{\left\|\mathsf{A}(\mathbf{x}_{\mathrm{LS}} - \mathbf{x}^{(k)})\right\|_2^2}{\|\mathsf{A}\|_{\mathrm{F}}^2}.\end{aligned}$$

By hypothesis, $\mathbf{x}^{(k)}$ is in the row space of A for any k when $\mathbf{x}^{(0)}$ is; in addition, the same is true for $\mathbf{x}_{\mathrm{LS}}$ by the definition of pseudoinverse [GL96]. Therefore, $\|\mathsf{A}(\mathbf{x}_{\mathrm{LS}} - \mathbf{x}^{(k)})\|_2 \geq \sigma_{\min}\|\mathbf{x}_{\mathrm{LS}} - \mathbf{x}^{(k)}\|_2$. ∎

Proof of Theorem 3.7. As in [Nee10], for any $i \in [m]$ define the affine hyperplanes:

$$\begin{aligned}\mathcal{H}_i &:= \left\{\mathbf{x} : \left\langle \mathsf{A}^{(i)},\ \mathbf{x}\right\rangle = y_i\right\}, \\ \mathcal{H}_i^{w_i} &:= \left\{\mathbf{x} : \left\langle \mathsf{A}^{(i)},\ \mathbf{x}\right\rangle = y_i + w_i\right\}.\end{aligned}$$

Assume for now that at the kth iteration of the randomized Kaczmarz algorithm applied on $(\mathsf{A}, \mathbf{b})$, the ith row is selected. Note that $\hat{\mathbf{x}}^{(k)}$ is the projection of $\hat{\mathbf{x}}^{(k-1)}$ on $\mathcal{H}_i^{w_i}$ by the definition of the randomized Kaczmarz algorithm on input $(\mathsf{A}, \mathbf{b})$. Let us denote the projection of $\hat{\mathbf{x}}^{(k-1)}$ on $\mathcal{H}_i$ by $\mathbf{x}^{(k)}$. The two affine hyperplanes $\mathcal{H}_i, \mathcal{H}_i^{w_i}$ are *parallel* with common normal $\mathsf{A}^{(i)}$, so $\mathbf{x}^{(k)}$ is the projection of $\hat{\mathbf{x}}^{(k)}$ on $\mathcal{H}_i$ and the minimum distance between $\mathcal{H}_i$ and $\mathcal{H}_i^{w_i}$ equals $|w_i|/\|\mathsf{A}^{(i)}\|_2$. In addition, $\mathbf{x}^* \in \mathcal{H}_i$ since $\langle \mathbf{x}^*,\ \mathsf{A}^{(i)}\rangle = y_i$; therefore by orthogonality we get that

$$\left\|\hat{\mathbf{x}}^{(k)} - \mathbf{x}^*\right\|_2^2 = \left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\|_2^2 + \left\|\hat{\mathbf{x}}^{(k)} - \mathbf{x}^{(k)}\right\|_2^2. \tag{6.1}$$

Since $\mathbf{x}^{(k)}$ is the projection of $\hat{\mathbf{x}}^{(k-1)}$ onto $\mathcal{H}_i$ (that is to say, $\mathbf{x}^{(k)}$ is a randomized Kaczmarz step applied on input $(\mathsf{A}, \mathbf{y})$ where the ith row is selected on the kth iteration) and $\hat{\mathbf{x}}^{(k-1)}$ is in the row space of A, Lemma 3.6 tells us that

$$\mathbb{E}_{k-1}\left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\|_2^2 \leq \left(1 - \frac{1}{\kappa_{\mathrm{F}}^2(\mathsf{A})}\right)\left\|\hat{\mathbf{x}}^{(k-1)} - \mathbf{x}^*\right\|_2^2. \tag{6.2}$$

Note that for a given selected row i we have $\|\hat{\mathbf{x}}^{(k)} - \mathbf{x}^{(k)}\|_2^2 = \frac{w_i^2}{\|\mathsf{A}^{(i)}\|_2^2}$; by the distribution of selecting the rows of A we have that

$$\mathbb{E}\left\|\hat{\mathbf{x}}^{(k)} - \mathbf{x}^{(k)}\right\|_2^2 = \sum_{i=1}^m q_i \frac{w_i^2}{\left\|\mathsf{A}^{(i)}\right\|_2^2} = \frac{\|\mathbf{w}\|_2^2}{\|\mathsf{A}\|_{\mathrm{F}}^2}. \tag{6.3}$$

Inequality (3.4) follows by taking expectation on both sides of (6.1) and bounding its resulting right-hand side using (6.2) and (6.3). Applying (3.4) inductively, it follows that

$$\mathbb{E}\left\|\hat{\mathbf{x}}^{(k)} - \mathbf{x}^*\right\|_2^2 \leq \left(1 - \frac{1}{\kappa_{\mathrm{F}}^2(\mathsf{A})}\right)^k \left\|\mathbf{x}^{(0)} - \mathbf{x}^*\right\|_2^2 + \frac{\|\mathbf{w}\|_2^2}{\|\mathsf{A}\|_{\mathrm{F}}^2}\sum_{i=0}^k \left(1 - \frac{1}{\kappa_{\mathrm{F}}^2(\mathsf{A})}\right)^i,$$

where we used that $\mathbf{x}^{(0)}$ is in the row space of A. The latter sum is bounded above by $\sum_{i=0}^{\infty}(1-\frac{1}{\kappa_{\mathrm{F}}^2(\mathsf{A})})^i = \|\mathsf{A}\|_{\mathrm{F}}^2/\sigma_{\min}^2 = \kappa_{\mathrm{F}}^2(\mathsf{A})$. ∎

Acknowledgments. We would like to thank the anonymous referees for their invaluable comments on an earlier draft of the present manuscript. The first author would like to thank Haim Avron for his technical support on several issues regarding Blendenpik and Philip A. Knight for sharing his unpublished manuscript [Kni96].

REFERENCES

[ABD+90] E. Anderson, Z. Bai, J. Dongarra, A. Greenbaum, A. McKenney, J. Du Croz, S. Hammerling, J. Demmel, C. Bischof, and D. Sorensen, *LAPACK: A portable linear algebra library for high-performance computers*, in Proceedings of the 1990 ACM/IEEE Conference on Supercomputing, Supercomputing '90, IEEE Computer Society Press, Los Alamitos, CA, 1990, pp. 2–11.

[AMT10] H. Avron, P. Maymounkov, and S. Toledo, *Blendenpik: Supercharging LAPACK's least-squares solver*, SIAM J. Sci. Comput., 32 (2010), pp. 1217–1236.

[Ans84] R. Ansorge, *Connections between the Cimmino-method and the Kaczmarz-method for the solution of singular and regular systems of equations*, *Computing*, 33 (1984), pp. 367–375.

[BBC+87] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, Software, Environments, Tools, SIAM, Philadelphia, 1987.

[Bj96] A. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996.

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

[Cen81] Y. Censor, *Row-action methods for huge and sparse systems and their applications*, SIAM Rev., 23 (1981), pp. 444–466.

[CP12] X. Chen and A. M. Powell, *Almost sure convergence of the Kaczmarz algorithm with random measurements*, J. Fourier Anal. Appl., 18 (2012), pp. 1195–1214.

[CRT11] E. S. Coakley, V. Rokhlin, and M. Tygert, *A fast randomized algorithm for orthogonal projection*, SIAM J. Sci. Comput., 33 (2011), pp. 849–868.

[CW09] K. L. Clarkson and D. P. Woodruff, *Numerical linear algebra in the streaming model*, in Proceedings of the Symposium on Theory of Computing (STOC '09), ACM, New York, 2009, pp. 205–214.

[CW12] K. L. Clarkson and D. P. Woodruff, *Low rank approximation and regression in input sparsity time*, in Proceedings of the 45th Annual ACM Symposium on the Theory of Computing (STOC'13), Palo Alto, CA, ACM, New York, 2013, pp. 81–90.

[CZ97] Y. Censor and S. A. Zenios, *Parallel Optimization: Theory, Algorithms, and Applications*, Numer. Math. Sci. Comput. Ser., Oxford University Press, New York, 1997.

[Dem88] J. W. Demmel, *The probability that a numerical analysis problem is difficult*, Math. Comp., 50 (1988), pp. 449–480.

[DH11] T. A. Davis and Y. Hu, *The university of Florida sparse matrix collection*, ACM Trans. Math. Software, 38 (2011).

[DMM06] P. Drineas, M. W. Mahoney, and S. Muthukrishnan, *Sampling algorithms for ℓ_2-regression and applications*, in Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, Philadelphia, 2006, pp. 1127–1136.

[DMMS11] P. Drineas, M. W. Mahoney, S. Muthukrishnan, and T. Sarlòs, *Faster least squares approximation*, Numer. Math., 117 (Feb. 2011), pp. 219–249.

[FCM+92] H. G. Feichtinger, C. Cenker, M. Mayer, H. Steier, and T. Strohmer, *New variants of the POCS method using affine subspaces of finite codimension with applications to irregular sampling*, in Visual Communications and Image Processing, Proc. SPIE 1818, 1992, pp. 299–310.

[FZ12] N. M. Freris and A. Zouzias, *Fast distributed smoothing of relative measurments*, IEEE Conference on Decision and Control (CDC), Maui, HI, 2012.

[Gal03] A. Galántai, *Projectors and Projection Methods*, Adv. Math. 6, Kluwer, Boston, 2004.

[GBH70] R. Gordon, R. Bender, and G. T. Herman, *Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography*, J.Theoret. Biol., 29 (1970), pp. 471–481.

[GL96] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.

[GV61] G. H. Golub and R. S. Varga, *Chebyshev semi-iterative methods, successive overrelaxation iterative methods, and second order Richardson iterative methods*, Numer. Math., 3 (1961), pp. 157–168.

[Her80] G. T. Herman, *Image Reconstruction from Projections: The Fundamentals of Computerized Tomography*, Comp. Sci. Appl. Math., Academic Press, San Francisco, 1980.

[Hig89] N. J. Higham, *A Collection of Test Matrices in MATLAB*, Technical report, Cornell University, Ithaca, NY, 1989.

[Hig96] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, 1996.

[HM93] G. T. Herman and L. B. Meyer, *Algebraic reconstruction techniques can be made computationally efficient*, IEEE Trans. Med. Imaging, 12 (1993), pp. 600–609.

[HN90] M. Hanke and W. Niethammer, *On the acceleration of Kaczmarz's method for inconsistent linear systems*, Linear Algebra Appl., 130 (1990), pp. 83–98.

[Kac37] S. Kaczmarz, *Angenäherte auflösung von systemen linearer gleichungen*, Bull. Internat. Acad. Polonaise Sci. Lett., 35 (1937), pp. 355–357.

[Kni93] P. A. Knight, *Error Analysis of Stationary Iteration and Associated Problems*. Ph.D. in Mathematics, Manchester University, Manchester, UK, 1993.

[Kni96] P. A. Knight, *A Rounding Error Analysis of Row-Action Methods*, manuscript.

[LL10] D. Leventhal and A. S. Lewis, *Randomized methods for linear constraints: Convergence rates and conditioning*, Math. Oper. Res., 35 (2010), pp. 641–654.

[McC75] S. F. McCormick, *An iterative procedure for the solution of constrained nonlinear equations with application to optimization problems*, Numer. Math., 23 (1975), pp. 371–385.

[MSM11] X. Meng, M. A. Saunders, and M. W. Mahoney, *LSRN: A Parallel Iterative Solver for Strongly Over- and Under-Determined Systems*, preprint, http://arxiv.org/abs/1109.5981, 2011.

[MZ11] A. Magen and A. Zouzias, *Low rank matrix-valued Chernoff bounds and approximate matrix multiplication*, in Proceedings of the 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, Philadelphia, 2011, pp. 1422–1436.

[Nat01] F. Natterer, *The Mathematics of Computerized Tomography*, Classics in Appl. Math. 32, SIAM, Philadelphia, 2001.

[NDT09] N. H. Nguyen, T. T. Do, and T. D. Tran, *A fast and efficient algorithm for low-rank approximation of a matrix*, in Proceedings of the Symposium on Theory of Computing (STOC '09), ACM, New York, 2009, pp. 215–224.

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

[NT12] D. Needell and J. A. Tropp, *Paved with good intentions: Analysis of a randomized block Kaczmarz method*, Linear Algebra Appl., to appear.

[Pop95] C. Popa, *Least-squares solution of overdetermined inconsistent linear systems using Kaczmarz's relaxation*, Int. J. Comput. Math., 55 (1995), pp. 79–89.

[Pop98] C. Popa, *Extensions of block-projections methods with relaxation parameters to inconsistent and rank-deficient least-squares problems*, BIT, 38 (19698), pp. 151–176.

[PS82] C. C. Paige and M. A. Saunders, *LSQR: An algorithm for sparse linear equations and sparse least squares*, ACM Trans. Math. Software, 8 (1982), pp. 43–71.

[RT08] V. Rokhlin and M. Tygert, *A fast randomized algorithm for overdetermined linear least-squares regression*, Proc. Natl. Acad. Sci. USA, 105 (2008), pp. 13212–13218.

[Saa03] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd edition, SIAM, Philadelphia, 2003.

[Sar06] T. Sarlòs, *Improved approximation algorithms for large matrices via random projections*, in Proceedings of the Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society, Los Alamitos, CA, 2006, pp. 143–152.

[Smi02] W. D. Smith, *How to Sample from a Probability Distribution*, Technical report, NEC, Princeton, NJ, 2002.

[SV09] T. Strohmer and R. Vershynin, *A Randomized Kaczmarz algorithm with exponential convergence*, J. Fourier Anal. Appl., 15 (2009), pp. 262–278.

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

[Tom55] C. Tompkins, *Projection methods in calculation*, in Proceedings of the 2nd Symposium in Linear Programming, National Bureau of Standards, Washington, DC, 1955, pp. 425–448.

[Vos91] M. D. Vose, *A linear algorithm for generating random numbers with a given distribution*, IEEE Trans. Software Engrg., 17 (1991), pp. 972–975.

[Wal77] A. J. Walker, *An efficient method for generating discrete random variables with general distributions*, ACM Trans. Math. Software, 3 (1977), pp. 253–256.

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