A fast randomized algorithm for overdetermined linear least-squares regression

Vladimir Rokhlin[†] and Mark Tygert[†]

Program in Applied Mathematics, Yale University, A. K. Watson Hall, 51 Prospect Street, New Haven, CT 06511

We introduce a randomized algorithm for overdetermined linear least-squares regression. Given an arbitrary full-rank $m \times n$ matrix A with $m \geq n$, any $m \times 1$ vector b, and any positive real number ε, the procedure computes an $n \times 1$ vector x such that x minimizes the Euclidean norm $\|Ax - b\|$ to relative precision ε. The algorithm typically requires $\mathcal{O}((\log(n) + \log(1/\varepsilon))mn + n^3)$ floating-point operations. This cost is less than the $\mathcal{O}(mn^2)$ required by the classical schemes based on QR-decompositions or bidiagonalization. We present several numerical examples illustrating the performance of the algorithm.

L east-squares fitting has permeated the sciences and engineering after its introduction over two centuries ago (see, for example, ref. 1 for a brief historical review). Linear least-squares regression is fundamental in the analysis of data, such as that generated from biology, econometrics, engineering, physics, and many other technical disciplines.

Perhaps the most commonly encountered formulation of linear least-squares regression involves a full-rank $m \times n$ matrix A and an $m \times 1$ column vector b, with $m \geq n$; the task is to find an $n \times 1$ column vector x such that the Euclidean norm $\|Ax - b\|$ is minimized. Classical algorithms using QR-decompositions or bidiagonalization require

$$C_{\text{classical}} = \mathcal{O}(mn^2) \qquad [1]$$

floating-point operations in order to compute x (see, for example, ref. 1 or Chapter 5 in ref. 2). The present article introduces a randomized algorithm that, given any positive real number ε, computes a vector x minimizing $\|Ax - b\|$ to relative precision ε, that is, the algorithm produces a vector x such that

$$\|Ax - b\| - \min_{y \in \mathbb{C}^n} \|Ay - b\| \leq \varepsilon \min_{y \in \mathbb{C}^n} \|Ay - b\|. \qquad [2]$$

This algorithm typically requires

$$C_{\text{rand}} = \mathcal{O}((\log(n) + \log(1/\varepsilon))mn + n^3) \qquad [3]$$

operations. When n is sufficiently large and m is much greater than n (that is, the regression is highly overdetermined), then the cost in Eq. 3 is less than the cost in Eq. 1. Furthermore, in the numerical experiments of *Numerical Results*, the algorithm of the present article runs substantially faster than the standard methods based on QR-decompositions.

The method of the present article is an extension of the methods introduced in refs. 3–5. Their algorithms and ours have similar costs; however, for the computation of x minimizing $\|Ax - b\|$ to relative precision ε, the earlier algorithms involve costs proportional to $1/\varepsilon$, whereas the algorithm of the present paper involves a cost proportional to $\log(1/\varepsilon)$ (see Eq. 3 above).

The present article describes algorithms optimized for the case when the entries of A and b are complex valued. Needless to say, real-valued versions of our schemes are similar. This article has the following structure: The first section sets the notation. The second section discusses a randomized linear transformation which can be applied rapidly to arbitrary vectors. The third section provides the relevant mathematical apparatus. The fourth section describes

the algorithm of the present article. The fifth section illustrates the performance of the algorithm via several numerical examples. The sixth section draws conclusions and proposes directions for future work.

Notation

In this section, we set notational conventions employed throughout the present article.

We denote an identity matrix by Eq. 1. We consider the entries of all vectors and matrices in this article to be complex valued. For any vector x, we define $\|x\|$ to be the Euclidean (l^2) norm of x. For any matrix A, we define A^* to be the adjoint of A, and we define the norm $\|A\|$ of A to be the spectral (l^2-operator) norm of A, that is, $\|A\|$ is the greatest singular value of A. We define the condition number of A to be the l^2 condition number of A, that is, the greatest singular value of A divided by the least singular value of A. If A has at least as many rows as columns, then the condition number of A is given by the expression

$$\kappa_A = \sqrt{\|A^*A\| \|(A^*A)^{-1}\|}. \qquad [4]$$

For any positive integers m and n with $m \geq n$, and any $m \times n$ matrix A, we will be using the singular value decomposition of A in the form

$$A_{m \times n} = U_{m \times n} \Sigma_{n \times n} V^*_{n \times n}, \qquad [5]$$

where U is an $m \times n$ matrix whose columns are orthonormal, V is an $n \times n$ matrix whose columns are orthonormal, and Σ is a diagonal $n \times n$ matrix whose entries are all nonnegative. We abbreviate "singular value decomposition" to "SVD" and "independent, identically distributed" to "i.i.d."

For any positive integer m, we define the discrete Fourier transform $F^{(m)}$ to be the complex $m \times m$ matrix with the entries

$$(F^{(m)})_{j,k} = \frac{1}{\sqrt{m}} e^{-2\pi i(j-1)(k-1)/m} \qquad [6]$$

for $j, k = 1, 2, \ldots, m-1, m$, where $i = \sqrt{-1}$ and $e = \exp(1)$; if the size m is clear from the context, then we omit the superscript in $F^{(m)}$, denoting the discrete Fourier transform by simply F.

Preliminaries

In this section, we discuss a subsampled randomized Fourier transform. Refs. 3–6 introduced a similar transform for similar purposes.

For any positive integers l and m with $l \leq m$, we define the $l \times m$ SRFT to be the $l \times m$ random matrix

$$T_{l \times m} = G_{l \times m} H_{m \times m}, \qquad [7]$$

where G and H are defined as follows.

In Eq. **7**, G is the $l \times m$ random matrix given by the formula

$$G_{l \times m} = S_{l \times m} F_{m \times m} D_{m \times m}, \quad \quad \textbf{[8]}$$

where S is the $l \times m$ matrix whose entries are all zeros, aside from a single 1 in column s_j of row j for $j = 1, 2, \ldots, l-1, l$, where $s_1, s_2, \ldots, s_{l-1}, s_l$ are i.i.d. integer random variables, each distributed uniformly over $\{1, 2, \ldots, m-1, m\}$; moreover, F is the $m \times m$ discrete Fourier transform, and D is the diagonal $m \times m$ matrix whose diagonal entries $d_1, d_2, \ldots, d_{m-1}, d_m$ are i.i.d. complex random variables, each distributed uniformly over the unit circle. (In our numerical implementations, we drew $s_1, s_2, \ldots, s_{l-1}, s_l$ from $\{1, 2, \ldots, m-1, m\}$ without replacement, instead of using i.i.d. draws.)

In Eq. **7**, H is the $m \times m$ random matrix given by the formula

$$H_{m \times m} = \Theta_{m \times m} \Pi_{m \times m} Z_{m \times m} \tilde{\Theta}_{m \times m} \tilde{\Pi}_{m \times m} \tilde{Z}_{m \times m}, \quad \quad \textbf{[9]}$$

where Π and $\tilde{\Pi}$ are $m \times m$ permutation matrices chosen independently and uniformly at random, and Z and $\tilde{Z}$ are diagonal $m \times m$ matrices whose diagonal entries $\zeta_1, \zeta_2, \ldots, \zeta_{m-1}, \zeta_m$ and $\tilde{\zeta}_1, \tilde{\zeta}_2, \ldots, \tilde{\zeta}_{m-1}, \tilde{\zeta}_m$ are i.i.d. complex random variables, each distributed uniformly over the unit circle; furthermore, Θ and $\tilde{\Theta}$ are the $m \times m$ matrices defined via the formulae

$$
\Theta_{m \times m} = \begin{pmatrix} \cos(\theta_1) & \sin(\theta_1) & 0 & 0 & 0 \\ -\sin(\theta_1) & \cos(\theta_1) & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot
$$

$$
\cdot \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \cos(\theta_2) & \sin(\theta_2) & 0 & 0 \\ 0 & -\sin(\theta_2) & \cos(\theta_2) & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \ddots \end{pmatrix} \cdots
$$

$$
\cdots \begin{pmatrix} \ddots & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \cos(\theta_{m-2}) & \sin(\theta_{m-2}) & 0 \\ 0 & 0 & -\sin(\theta_{m-2}) & \cos(\theta_{m-2}) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot
$$

$$
\cdot \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \cos(\theta_{m-1}) & \sin(\theta_{m-1}) \\ 0 & 0 & 0 & -\sin(\theta_{m-1}) & \cos(\theta_{m-1}) \end{pmatrix} \quad \textbf{[10]}
$$

and (the same as Eq. **10**, but with tildes)

$$
\tilde{\Theta}_{m \times m} = \begin{pmatrix} \cos(\tilde{\theta}_1) & \sin(\tilde{\theta}_1) & 0 & 0 & 0 \\ -\sin(\tilde{\theta}_1) & \cos(\tilde{\theta}_1) & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot
$$

$$
\cdot \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \cos(\tilde{\theta}_2) & \sin(\tilde{\theta}_2) & 0 & 0 \\ 0 & -\sin(\tilde{\theta}_2) & \cos(\tilde{\theta}_2) & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \ddots \end{pmatrix} \cdots
$$

$$
\cdots \begin{pmatrix} \ddots & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \cos(\tilde{\theta}_{m-2}) & \sin(\tilde{\theta}_{m-2}) & 0 \\ 0 & 0 & -\sin(\tilde{\theta}_{m-2}) & \cos(\tilde{\theta}_{m-2}) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot
$$

$$
\cdot \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \cos(\tilde{\theta}_{m-1}) & \sin(\tilde{\theta}_{m-1}) \\ 0 & 0 & 0 & -\sin(\tilde{\theta}_{m-1}) & \cos(\tilde{\theta}_{m-1}) \end{pmatrix}, \quad \textbf{[11]}
$$

where $\theta_1, \theta_2, \ldots, \theta_{m-2}, \theta_{m-1}, \tilde{\theta}_1, \tilde{\theta}_2, \ldots, \tilde{\theta}_{m-2}, \tilde{\theta}_{m-1}$ are i.i.d. real random variables drawn uniformly from $[0, 2\pi]$. We observe that Θ, $\tilde{\Theta}$, Π, $\tilde{\Pi}$, Z, and $\tilde{Z}$ are all unitary, and so H is also unitary.

We call the transform T an "SRFT" for lack of a better term.

Remark 1. *Our earlier articles, refs. 7 and 8, omitted the matrix H in the definition of the SRFT (Eq. **7**). Numerical experiments indicate that including H improves the performance of the algorithm of the present article on sparse matrices.*

The following lemma is similar to the subspace Johnson–Lindenstrauss lemma (Corollary 11) of ref. 4, and is proven (in a slightly different form) as Lemma 4.4 of ref. 8. The lemma provides a highly probable upper bound on the condition number of the product of the $l \times m$ matrix G defined in Eq. **8** and an independent $m \times n$ matrix U whose columns are orthonormal, assuming that l is less than m and is sufficiently greater than n^2.

Lemma 1. *Suppose that α and β are real numbers >1, and l, m, and n are positive integers, such that*

$$m > l \geq \left(\frac{\alpha^2 + 1}{\alpha^2 - 1} \right)^2 \beta n^2. \quad \quad \textbf{[12]}$$

*Suppose further that G is the $l \times m$ random matrix defined in Eq. **8**. Suppose in addition that U is an $m \times n$ random matrix whose columns are orthonormal, and that U is independent of G.*

Then, the condition number of GU is at most α with probability at least $1 - \frac{1}{\beta}$.

The following corollary of *Lemma 1* follows immediately from the fact that the random matrix H defined in Eq. **9** is unitary and independent of the random matrix G defined in Eq. **8**. The corollary provides a highly probable upper bound on the condition number of the $l \times m$ SRFT (defined in Eq. **7**) applied to an $m \times n$ matrix U whose columns are orthonormal, assuming that l is less than m and is sufficiently greater than n^2.

Corollary 1. *Suppose that α and β are real numbers greater than 1, and l, m, and n are positive integers, such that Eq. **12** holds. Suppose further that T is the $l \times m$ SRFT defined in Eq. **7**. Suppose in addition that U is an $m \times n$ matrix whose columns are orthonormal.*

Then, the condition number of TU is at most α with probability at least $1 - \frac{1}{\beta}$.

The following lemma states that, if A is an $m \times n$ matrix, b is an $m \times 1$ vector, and T is the $l \times m$ SRFT defined in Eq. **7**, then, with high probability, an $n \times 1$ vector z minimizing $\|TAz - Tb\|$ also minimizes $\|Az - b\|$ to within a reasonably small factor. Whereas solving $Az \approx b$ in the least-squares sense involves m simultaneous linear equations, solving $TAz \approx Tb$ involves just l simultaneous equations. This lemma is modeled after similar results in refs. 3–5, and is proven (in a slightly different form) as Lemma 4.8 of ref. 8.

Lemma 2. *Suppose that α and β are real numbers >1, and l, m, and n are positive integers, such that*

$$m > l \geq \left(\frac{\alpha^2 + 1}{\alpha^2 - 1}\right)^2 \beta(n+1)^2. \qquad [13]$$

*Suppose further that T is the l × m SRFT defined in Eq. **7**. Suppose in addition that A is an m × n matrix, and b is an m × 1 vector. Suppose finally that z is an n × 1 vector minimizing the quantity*

$$\|TAz - Tb\|. \qquad [14]$$

Then,

$$\|Az - b\| \leq \alpha \min_{y \in \mathbb{C}^n} \|Ay - b\| \qquad [15]$$

with probability at least $1 - \frac{1}{\beta}$.

Mathematical Apparatus

In this section, we prove a theorem that (in conjunction with *Corollary 1*) guarantees that the algorithm of the present article is fast.

In the proof of *Theorem 1* below, we will need the following technical lemma.

Lemma 3. *Suppose that l, m, and n are positive integers such that $m \geq l \geq n$. Suppose further that A is an m × n matrix, and that the SVD of A is*

$$A_{m \times n} = U_{m \times n} \Sigma_{n \times n} V_{n \times n}^*, \qquad [16]$$

where U is an m × n matrix whose columns are orthonormal, V is an n × n matrix whose columns are orthonormal, and Σ is a diagonal n × n matrix whose entries are all nonnegative. Suppose in addition that T is an l × m matrix, and that the SVD of the l × n matrix TU is

$$T_{l \times m} U_{m \times n} = \tilde{U}_{l \times n} \tilde{\Sigma}_{n \times n} \tilde{V}_{n \times n}^*. \qquad [17]$$

Then, there exist an n × n matrix P, and an l × n matrix Q whose columns are orthonormal, such that

$$T_{l \times m} A_{m \times n} = Q_{l \times n} P_{n \times n}. \qquad [18]$$

*Furthermore, if P is any n × n matrix, and Q is any l × n matrix whose columns are orthonormal, such that P and Q satisfy Eq. **18**, then*

$$P_{n \times n} = (Q_{l \times n})^* \tilde{U}_{l \times n} \tilde{\Sigma}_{n \times n} \tilde{V}_{n \times n}^* \Sigma_{n \times n} V_{n \times n}^*; \qquad [19]$$

if, in addition, the matrices A and TU both have full rank (rank n), then there exists a unitary n × n matrix W such that

$$\tilde{U}_{l \times n} = Q_{l \times n} W_{n \times n}. \qquad [20]$$

Proof: An example of matrices P and Q satisfying Eq. **18** such that the columns of Q are orthonormal is $P = \tilde{\Sigma} \tilde{V}^* \Sigma V^*$ and $Q = \tilde{U}$.

We now assume that P is any n × n matrix, and Q is any l × n matrix whose columns are orthonormal, such that P and Q satisfy Eq. **18**. Combining Eqs. **18**, **16**, and **17** yields

$$Q_{l \times n} P_{n \times n} = \tilde{U}_{l \times n} \tilde{\Sigma}_{n \times n} \tilde{V}_{n \times n}^* \Sigma_{n \times n} V_{n \times n}^*. \qquad [21]$$

Combining Eq. **21** and the fact that the columns of Q are orthonormal (so that $Q^*Q = \mathbf{1}$) yields Eq. **19**.

For the remainder of the proof, we assume that the matrices A and TU both have full rank. To establish Eq. **20**, we demonstrate that the column spans of Q and $\tilde{U}$ are the same. It then follows from the fact that the columns of Q are an orthonormal basis for this column span, as are the columns of $\tilde{U}$, that there exists a unitary n × n matrix W satisfying Eq. **20**. We now complete the proof by showing that

$$\text{column span of } Q_{l \times n} = \text{column span of } \tilde{U}_{l \times n}. \qquad [22]$$

Obviously, it follows from Eq. **21** that

column span of $Q_{l \times n} P_{n \times n}$

$$= \text{column span of } \tilde{U}_{l \times n} \tilde{\Sigma}_{n \times n} \tilde{V}_{n \times n}^* \Sigma_{n \times n} V_{n \times n}^*; \qquad [23]$$

we will simplify both sides of Eq. **23**, in order to obtain Eq. **22**.

It follows from the assumption that A and TU both have full rank that the matrices Σ and $\tilde{\Sigma}$ in the SVDs in Eqs. **16** and **17** are nonsingular, and so (as the unitary matrices V and $\tilde{V}$ are also nonsingular)

column span of $\tilde{U}_{l \times n} \tilde{\Sigma}_{n \times n} \tilde{V}_{n \times n}^* \Sigma_{n \times n} V_{n \times n}^*$

$$= \text{column span of } \tilde{U}_{l \times n}. \qquad [24]$$

Combining Eqs. **23**, **24**, and the fact that the column span of $\tilde{U}$ is n-dimensional (after all, the n columns of $\tilde{U}$ are orthonormal) yields that the column span of QP is n-dimensional. Combining this fact, the fact that the column span of QP is a subspace of the column span of Q, and the fact that the column span of Q is n-dimensional (after all, the n columns of Q are orthonormal) yields

$$\text{column span of } Q_{l \times n} P_{n \times n} = \text{column span of } Q_{l \times n}. \qquad [25]$$

Combining Eqs. **23**, **24**, and **25** yields Eq. **22**, completing the proof.

The following theorem states that, given an m × n matrix A, the condition number of a certain preconditioned version of A corresponding to an l × m matrix T is equal to the condition number of TU, where U is an m × n matrix of orthonormal left singular vectors of A.

Theorem 1. *Suppose that l, m, and n are positive integers such that $m \geq l \geq n$. Suppose further that A is a full-rank m × n matrix, and that the SVD of A is*

$$A_{m \times n} = U_{m \times n} \Sigma_{n \times n} V_{n \times n}^*. \qquad [26]$$

Suppose in addition that T is an l × m matrix such that the l × n matrix TU has full rank.

Then, there exist an n × n matrix P, and an l × n matrix Q whose columns are orthonormal, such that

$$T_{l \times m} A_{m \times n} = Q_{l \times n} P_{n \times n}. \qquad [27]$$

*Furthermore, if P is any n × n matrix, and Q is any l × n matrix whose columns are orthonormal, such that P and Q satisfy Eq. **27**, then the condition numbers of AP^{-1} and TU are equal.*

Proof: *Lemma 3* guarantees the existence of matrices P and Q satisfying Eq. **27** such that the columns of Q are orthonormal.

For the remainder of the proof, we assume that P is an n × n matrix, and Q is an l × n matrix whose columns are orthonormal, such that P and Q satisfy Eq. **27**. Combining Eq. **19**, Eq. **20**, and the fact that the columns of Q are orthonormal (so that $Q^*Q = \mathbf{1}$) yields

$$P_{n \times n} = W_{n \times n} \tilde{\Sigma}_{n \times n} \tilde{V}_{n \times n}^* \Sigma_{n \times n} V_{n \times n}^*, \qquad [28]$$

where W is the matrix from Eq. **20**, and $\tilde{\Sigma}$ and $\tilde{V}$ are the matrices from the SVD in Eq. **17**. Combining Eq. **26**, Eq. **28**, and the fact that V, $\tilde{V}$, and W are unitary yields

$$A_{m \times n} P_{n \times n}^{-1} = U_{m \times n} \tilde{V}_{n \times n} \tilde{\Sigma}_{n \times n}^{-1} W_{n \times n}^*. \qquad [29]$$

Combining Eq. **29**, the fact that $\tilde{V}$ and W are unitary, the fact that the columns of U are orthonormal (so that $U^*U = \mathbf{1}$), and the fact that $\tilde{\Sigma}$ is diagonal yields

$$\left\|(A_{m \times n} P_{n \times n}^{-1})^* (A_{m \times n} P_{n \times n}^{-1})\right\| = \left\|\tilde{\Sigma}_{n \times n}^{-1}\right\|^2 \qquad [30]$$

and

$$\|((A_{m\times n}P_{n\times n}^{-1})^{*}(A_{m\times n}P_{n\times n}^{-1}))^{-1}\| = \|\tilde{\Sigma}_{n\times n}\|^{2}. \qquad [31]$$

Combining Eqs. **4**, **30**, and **31**, and the SVD in Eq. **17** yields that the condition numbers of AP^{-1} and TU are equal, completing the proof.

The Algorithm

In this section, we describe the algorithm of the present article, giving details about its implementation and computational costs.

Description of the Algorithm. Suppose that m and n are positive integers with $m \geq n$, A is a full-rank $m \times n$ matrix, and b is an $m \times 1$ column vector. In this subsection, we describe a procedure for the computation of an $n \times 1$ column vector x such that x minimizes $\|Ax - b\|$ to arbitrarily high precision.

Rather than directly calculating the vector x minimizing $\|Ax - b\|$, we will first calculate the vector y minimizing $\|Cy - b\|$, where $C = AP^{-1}$ and $y = Px$, with an appropriate choice of an $n \times n$ matrix P; the matrix P is known as a preconditioning matrix. With an appropriate choice of P, the condition number of C is reasonably small, and so an iterative solver such as the conjugate gradient method will require only a few iterations in order to obtain y minimizing $\|Cy - b\|$ to high precision. Once we have calculated y, we obtain x via the formula $x = P^{-1}y$.

To construct the preconditioning matrix P, we compute $E = TA$, where T is the $l \times m$ SRFT defined in Eq. **7**, with $m \geq l \geq n$. We then form a pivoted QR-decomposition of E, computing an $l \times n$ matrix Q whose columns are orthonormal, an upper-triangular $n \times n$ matrix R, and an $n \times n$ permutation matrix Π, such that $E = QR\Pi$. We use the product $P = R\Pi$ as the preconditioning matrix. Fortuitously, because this matrix P is the product of an upper-triangular matrix and a permutation matrix, we can apply P^{-1} or $(P^{-1})^{*}$ to any arbitrary vector rapidly, without calculating the entries of P^{-1} explicitly.

The condition number of $C = AP^{-1}$ is reasonably small with very high probability whenever l is sufficiently greater than n, due to *Theorem 1* and *Corollary 1*; moreover, numerical experiments reported in *Numerical Results* suggest that the condition number of C is practically always less than 3 or so when $l = 4n$. Therefore, when l is sufficiently greater than n, the conjugate gradient method requires only a few iterations in order to compute y minimizing $\|Cy - b\|$ to high precision; furthermore, the conjugate gradient method requires only applications of A, A^{*}, P^{-1}, and $(P^{-1})^{*}$ to vectors, and all of these matrices are readily available for application to vectors. Once we have calculated y, we obtain x minimizing $\|Ax - b\|$ via the formula $x = P^{-1}y$.

There is a natural choice for the starting vector of the conjugate gradient iterations. Combining the fact that $E = TA$ with Eqs. **14** and **15** yields that, with high probability, the $n \times 1$ vector z minimizing $\|Ez - Tb\|$ also minimizes $\|Az - b\|$ to within a factor of 3, provided that l is sufficiently greater than n (in practice, $l = 4n$ is sufficient). Thus, z is a good choice for the starting vector of the iterations. Moreover, combining the facts that $E = QP$ and that the columns of Q are orthonormal yields that $z = P^{-1}Q^{*}Tb$, providing a convenient means of computing z.

In summary, if ε is any specified positive real number, we can compute an $n \times 1$ column vector x such that x minimizes $\|Ax - b\|$ to relative precision ε, via the following five steps:

1. Compute $E = TA$, where T is the $l \times m$ SRFT defined in Eq. **7**, with $m \geq l \geq n$. (See, for example, Subsection 3.3 of ref. 8 for details on applying the SRFT rapidly.)
2. Form a pivoted QR-decomposition of E from Step 1, computing an $l \times n$ matrix Q whose columns are orthonormal, an upper-triangular $n \times n$ matrix R, and an $n \times n$ permutation matrix Π, such that $E = QR\Pi$. (See, for example, Chapter 5 in ref. 2 for details on computing such a pivoted QR-decomposition.)

3. Compute the $n \times 1$ column vector $z = P^{-1}(Q^{*}(Tb))$, where T is the $l \times m$ SRFT defined in Eq. **7**, Q is the $l \times n$ matrix from Step 2 whose columns are orthonormal, and $P = R\Pi$; R and Π are the upper-triangular and permutation matrices from Step 2. (See, for example, Subsection 3.3 of ref. 8 for details on applying the SRFT rapidly.)
4. Compute an $n \times 1$ column vector y which minimizes $\|AP^{-1}y - b\|$ to relative precision ε, via the preconditioned conjugate gradient iterations, where $P = R\Pi$ is the preconditioning matrix; R and Π are the upper-triangular and permutation matrices from Step 2. Use z from Step 3 as the starting vector. (See, for example, Algorithm 7.4.3 in ref. 1 for details on the preconditioned conjugate gradient iterations for linear least-squares problems.)
5. Compute $x = P^{-1}y$, where y is the vector from Step 4, and again $P = R\Pi$; R and Π are the upper-triangular and permutation matrices from Step 2.

Cost. In this subsection, we estimate the number of floating-point operations required by each step of the algorithm of the preceding subsection.

We denote by κ the condition number of the preconditioned matrix AP^{-1}. The five steps of the algorithm incur the following costs:

1. Applying T to every column of A costs $\mathcal{O}(mn\log(l))$.
2. Computing the pivoted QR-decomposition of E costs $\mathcal{O}(n^{2}l)$.
3. Applying T to b costs $\mathcal{O}(m\log(l))$. Applying Q^{*} to Tb costs $\mathcal{O}(nl)$. Applying $P^{-1} = \Pi^{-1}R^{-1}$ to $Q^{*}Tb$ costs $\mathcal{O}(n^{2})$.
4. When $l \geq 4n^{2}$, Eq. **15** guarantees with high probability that the vector z has a residual $\|Az - b\|$ that is no greater than 3 times the minimum possible. When started with such a vector, the preconditioned conjugate gradient algorithm requires $\mathcal{O}(\kappa\log(1/\varepsilon))$ iterations in order to improve the relative precision of the residual to ε (see, for example, formula 7.4.7 in ref. 1). Applying A and A^{*} a total of $\mathcal{O}(\kappa\log(1/\varepsilon))$ times costs $\mathcal{O}(mn\kappa\log(1/\varepsilon))$. Applying P^{-1} and $(P^{-1})^{*}$ a total of $\mathcal{O}(\kappa\log(1/\varepsilon))$ times costs $\mathcal{O}(n^{2}\kappa\log(1/\varepsilon))$. These costs dominate the costs of the remaining computations in the preconditioned conjugate gradient iterations.
5. Applying $P^{-1} = \Pi^{-1}R^{-1}$ to y costs $\mathcal{O}(n^{2})$.

Summing up the costs in the five steps above, we see that the cost of the entire algorithm is

$$C_{\text{theoretical}} = \mathcal{O}((\log(l) + \kappa\log(1/\varepsilon))mn + n^{2}l). \qquad [32]$$

The condition number κ of the preconditioned matrix AP^{-1} can be made arbitrarily close to 1, by choosing l sufficiently large. According to *Theorem 1* and *Corollary 1*, choosing $l \geq 4n^{2}$ guarantees that κ is at most 3, with high probability.

Remark 2. *Currently, our estimates require that l be at least $4n^{2}$ in order to ensure with high probability that κ is at most 3 and that the residual $\|Az - b\|$ is no greater than three times the minimum possible. However, our numerical experiments indicate that it is not necessary for l to be as large as $4n^{2}$ (though it is sufficient). Indeed, in all of our tests, choosing $l = 4n$ produced a condition number κ less than 3 and a residual $\|Az - b\|$ no greater than three times the minimum possible residual. With $l = 4n$ and $\kappa \leq 3$, the cost in Eq. **32** becomes*

$$C_{\text{typical}} = \mathcal{O}((\log(n) + \log(1/\varepsilon))mn + n^{3}). \qquad [33]$$

Numerical Results

In this section, we describe the results of several numerical tests of the algorithm of the present article.

Table 1. Timings

m	n	l	t_0	t	t_0/t
32768	64	256	.14E1	.13E1	1.1
32768	128	512	.55E1	.27E1	2.0
32768	256	1024	.22E2	.59E1	3.7
32768	512	2048	.89E2	.15E2	5.7

We use the algorithm to compute an $n \times 1$ vector x such that x minimizes $\|Ax - b\|$ to high precision, where b is an $m \times 1$ unit vector, and A is the $m \times n$ matrix defined via the formula

$$A_{m \times n} = U_{m \times n} \Sigma_{n \times n} V^*_{n \times n}; \qquad [34]$$

in all experiments described below, U is obtained by applying the Gram–Schmidt process to the columns of an $m \times n$ matrix whose entries are i.i.d. centered complex Gaussian random variables, V is obtained by applying the Gram–Schmidt process to the columns of an $n \times n$ matrix whose entries are i.i.d. centered complex Gaussian random variables, and Σ is a diagonal $n \times n$ matrix, with the diagonal entries

$$\Sigma_{k,k} = 10^{-6(k-1)/(n-1)} \qquad [35]$$

for $k = 1, 2, \ldots, n - 1, n$. Clearly, the condition number κ_A of A is

$$\kappa_A = \Sigma_{1,1}/\Sigma_{n,n} = 10^6. \qquad [36]$$

The $m \times 1$ unit vector b is defined via the formula

$$b = 10^{-3}w + Ay, \qquad [37]$$

where w is a random $m \times 1$ unit vector orthogonal to the column span of A, and Ay is a vector from the column span of A such that $\|b\| = 1$.

We implemented the algorithm in Fortran 77 in double-precision arithmetic, and used the Lahey/Fujitsu Express v6.2 compiler. We used one core of a 1.86 GHz Intel Centrino Core Duo microprocessor with 1 GB of RAM. For the direct computations, we used the classical algorithm for pivoted *QR*-decompositions based on plane (Householder) reflections (see, for example, Chapter 5 in ref. 2).

Table 1 displays timing results with $m = 32768$ for various values of n; Table 2 displays the corresponding errors. Table 3 displays timing results with $n = 256$ for various values of m; Table 4 displays the corresponding errors.

The headings of the tables are as follows:

- m is the number of rows in the matrix A, as well as the length of the vector b, in $\|Ax - b\|$.
- n is the number of columns in the matrix A, as well as the length of the vector x, in $\|Ax - b\|$.
- l is the number of rows in the matrix T used in Steps 1 and 3 of the procedure in *Description of the Algorithm*.
- t_0 is the time in seconds required by the direct, classical algorithm.
- t is the time in seconds required by the algorithm of the present article.

Table 2. Condition numbers after preconditioning and accuracies

m	n	l	κ	i	ε_{rel}
32768	64	256	2.7	14	.120E−15
32768	128	512	2.9	14	.132E−15
32768	256	1024	2.9	14	.429E−15
32768	512	2048	2.9	13	.115E—14

Table 3. Timings

m	n	l	t_0	t	t_0/t
2048	256	1024	.12E1	.71E0	1.6
4096	256	1024	.25E1	.94E0	2.6
8192	256	1024	.51E1	.14E1	3.5
16384	256	1024	.10E2	.26E1	4.1
32768	256	1024	.22E2	.50E1	4.4
65536	256	1024	.49E2	.11E2	4.4

- t_0/t is the factor by which the algorithm of the present article is faster than the classical algorithm.
- κ is the condition number of AP^{-1}, the preconditioned version of the matrix A.
- i is the number of iterations required by the preconditioned conjugate gradient method to yield the requested precision ε_{rel} of 0.5E–14 or better in Table 2, and 0.5E–10 or better in Table 4.
- ε_{rel} is defined via the formula

$$\varepsilon_{rel} = \frac{\delta - \delta_{min}}{\kappa_A \cdot \delta_{min}}, \qquad [38]$$

where κ_A is the condition number of A given in Eq. **36**, $\delta = \|Ax - b\|$ (x is the solution vector produced by the randomized algorithm), and

$$\delta_{min} = \min_{y \in \mathbb{C}^n} \|Ay - b\| = 10^{-3}. \qquad [39]$$

Remark 3. *Standard perturbation theory shows that ε_{rel} is the appropriately normalized measure of the precision produced by the algorithm; see, for example, formula 1.4.27 in ref. 1.*

The values for ε_{rel} and i reported in the tables are the worst (maximum) values encountered during 10 independent randomized trials of the algorithm, as applied to the same matrix A and vector b. The values for t reported in the tables are the average values over 10 independent randomized trials. None of the quantities reported in the tables varied significantly over repeated randomized trials.

The following observations can be made from the examples reported here, and from our more extensive experiments:

1. When $m = 32768, n = 512$, and the condition number of A is 10^6, the randomized algorithm runs over five times faster than the classical algorithm based on plane (Householder) reflections, even at full double precision.
2. As observed in *Remark 2*, our choice $l = 4n$ seems to ensure that the condition number κ of the preconditioned matrix is at most 3. More generally, κ seems to be smaller than a function of the ratio l/n.
3. The algorithm of the present article produces high precision at reasonably low cost.

Table 4. Condition numbers after preconditioning and accuracies

m	n	l	κ	i	ε_{rel}
2048	256	1024	2.2	4	.326E−10
4096	256	1024	2.6	5	.364E−10
8192	256	1024	2.7	6	.160E−10
16384	256	1024	2.8	7	.599E—11
32768	256	1024	2.9	8	.502E--11
65536	256	1024	2.9	8	.177E—11

Conclusions and Generalizations

This article provides a fast algorithm for overdetermined linear least-squares regression. If the matrices A and A^* from the regression involving $\|Ax - b\|$ can be applied sufficiently rapidly to arbitrary vectors, then the algorithm of the present article can be accelerated further. Moreover, the methods developed here for overdetermined regression extend to underdetermined regression.

The theoretical bounds in *Lemma 1*, *Corollary 1*, and *Lemma 2* should be considered preliminary. Our numerical experiments indicate that the algorithm of the present article performs better than our estimates guarantee. Furthermore, there is nothing magical about the subsampled randomized Fourier transform defined in Eq. **7**. In our experience, several other similar transforms appear to work at least as well, and we are investigating these alternatives (see, for example, ref. 9).

1. Björck Å (1996) *Numerical Methods for Least Squares Problems* (Soc Indust Apply Math, Philadelphia).
2. Golub GH, Van Loan CF (1996) *Matrix Computations* (Johns Hopkins University Press, Baltimore), 3rd Ed.
3. Sarlós T (2006) Improved approximation algorithms for large matrices via random projections. *Proceedings FOCS 2006* (IEEE Press, New York), pp 143–152.
4. Sarlós T (2006) Improved approximation algorithms for large matrices via random projections, long form (Eötvös Loránd University Informatics Laboratory), Tech Rep. Available at http://www.ilab.sztaki.hu/~stamas/publications/rp-long.pdf.
5. Drineas P, Mahoney MW, Muthukrishnan S, Sarlós T (2007) Faster least squares approximation (arXiv), Tech Rep 0710.1435. Available at http://arxiv.org/.
6. Ailon N, Chazelle B (2007) Approximate nearest neighbors and the fast Johnson–Lindenstrauss transform. *SIAM J Comput*, in press.
7. Liberty E, Woolfe F, Martinsson P-G, Rokhlin V, Tygert M (2007) Randomized algorithms for the low-rank approximation of matrices. *Proc Natl Acad Sci USA* 104:20167–20172.
8. Woolfe F, Liberty E, Rokhlin V, Tygert M (2007) A fast randomized algorithm for the approximation of matrices. *Appl Comput Harmonic Anal*, in press.
9. Ailon N, Liberty E (2007) Fast dimension reduction using Rademacher series on dual BCH codes. (Yale Univ Department of Computer Science), Tech Rep 1385.

APPLIED MATHEMATICS