A fast algorithm for computing minimal-norm solutions to underdetermined systems of linear equations

Mark Tygert

July 24, 2009

Abstract

We introduce a randomized algorithm for computing the minimal-norm solution to an underdetermined system of linear equations. Given an arbitrary full-rank matrix $A_{m\times n}$ with $m < n$, any vector $b_{m\times 1}$, and any positive real number ε less than 1, the procedure computes a vector $x_{n\times 1}$ approximating to relative precision ε or better the vector $p_{n\times 1}$ of minimal Euclidean norm satisfying $A_{m\times n}\, p_{n\times 1} = b_{m\times 1}$. The algorithm typically requires $\mathcal{O}(mn\, \log(\sqrt{n}/\varepsilon) + m^3)$ floating-point operations, generally less than the $\mathcal{O}(m^2\, n)$ required by the classical schemes based on QR-decompositions or bidiagonalization. We present several numerical examples illustrating the performance of the algorithm.

1 Introduction

Underdetermined systems of linear equations have arisen frequently in modern statistics and data analysis, and have been attracting much attention recently in various application domains; see, for example, [3], [4], and [6]. The solutions to underdetermined systems are not unique; the present article focuses on the solutions whose Euclidean norms are minimal.

Given a full-rank matrix $A_{m\times n}$ and a vector $b_{m\times 1}$, with $m < n$, we would like to compute an accurate approximation to the vector $p_{n\times 1}$ of minimal Euclidean norm satisfying

$$A_{m\times n}\, p_{n\times 1} = b_{m\times 1}. \tag{1}$$

Classical algorithms using QR-decompositions or bidiagonalization require

$$C_{\text{classical}} = \mathcal{O}(m^2\, n) \tag{2}$$

floating-point operations in order to compute $p_{n\times 1}$ (see, for example, Chapter 5 in [8]).

The present paper introduces a randomized algorithm that, given any positive real number ε less than 1, computes a vector $x_{n\times 1}$ approximating $p_{n\times 1}$ to relative precision ε or better with respect to the Euclidean norm, that is, the algorithm produces a vector $x_{n\times 1}$ such that

$$\|x_{n\times 1} - p_{n\times 1}\| \leq \varepsilon\, \|p_{n\times 1}\|, \tag{3}$$

where $\| \cdot \|$ denotes the Euclidean norm. This algorithm typically requires

$$C_{\text{rand}} = \mathcal{O}(mn \log(\sqrt{n}/\varepsilon) + m^3) \tag{4}$$

floating-point operations. When m is sufficiently large and n is much greater than m (that is, the system of linear equations is highly underdetermined), then the cost in (4) is less than the cost in (2). Moreover, in the numerical experiments of Section 7, the algorithm of the present article runs substantially faster than the standard methods based on QR-decompositions.

The present paper describes an algorithm optimized for the case when the entries of $A_{m\times n}$ and $b_{m\times 1}$ are complex valued. Needless to say, real-valued versions of our scheme are similar. The present article has the following structure: Section 2 sets the notation. Section 3 discusses a randomized linear transformation which can be applied rapidly to arbitrary vectors. Section 4 describes the algorithm of the present paper. Section 5 proves that the procedure succeeds with high probability. Section 6 estimates the computational costs of the algorithm. Section 7 illustrates the performance of the scheme via several numerical examples. Section 8 contains several concluding comments.

2 Notation

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

We abbreviate "independent and identically distributed" to "i.i.d." We consider the entries of all vectors and matrices in this paper 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. 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.

For any positive integer n, we define the discrete Fourier transform $F_{n\times n}$ to be the matrix with the entries

$$F_{j,k} = \frac{1}{\sqrt{n}}\, e^{-2\pi i (j-1)(k-1)/n} \tag{5}$$

for $j, k = 1, 2, \ldots, n-1, n$, where $i = \sqrt{-1}$ and $e = \exp(1)$.

3 Preliminaries

In this section, we discuss a subsampled randomized Fourier transform. [1], [7], [10], and [11] introduced a similar transform for similar purposes (these articles motivated us to write the present paper).

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

$$T_{l\times n} = G_{l\times n}\, H_{n\times n}, \tag{6}$$

where $G_{l\times n}$ and $H_{n\times n}$ are defined as follows.

In (6), $G_{l\times n}$ is the random matrix given by the formula

$$G_{l\times n} = S_{l\times n}\, F_{n\times n}\, D_{n\times n}, \tag{7}$$

where $S_{l\times n}$ is the 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 $\{1, 2, \ldots, n-1, n\}$; moreover, $F_{n\times n}$ is the discrete Fourier transform defined in (5), and $D_{n\times n}$ is the diagonal matrix whose diagonal entries $d_1, d_2, \ldots, d_{n-1}, d_n$ 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 $\{1, 2, \ldots, n-1, n\}$ without replacement, instead of using i.i.d. draws.) We observe that both $F_{n\times n}$ and $D_{n\times n}$ are unitary.

In (6), $H_{n\times n}$ is the random matrix given by the formula

$$H_{n\times n} = \Theta_{n\times n}\, \Pi_{n\times n}\, Z_{n\times n}\, \tilde{\Theta}_{n\times n}\, \tilde{\Pi}_{n\times n}\, \tilde{Z}_{n\times n}, \tag{8}$$

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

$$\begin{aligned}
\Theta_{n\times n} = &\begin{pmatrix} \cos(\theta_1) & \sin(\theta_1) & 0 & 0 & 0 \\ -\sin(\theta_1) & \cos(\theta_1) & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \\
&\cdot \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \cos(\theta_2) & \sin(\theta_2) & 0 & 0 \\ 0 & -\sin(\theta_2) & \cos(\theta_2) & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \ddots \end{pmatrix} \cdots \\
&\cdots \begin{pmatrix} \ddots & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \cos(\theta_{n-2}) & \sin(\theta_{n-2}) & 0 \\ 0 & 0 & -\sin(\theta_{n-2}) & \cos(\theta_{n-2}) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \\
&\cdot \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \cos(\theta_{n-1}) & \sin(\theta_{n-1}) \\ 0 & 0 & 0 & -\sin(\theta_{n-1}) & \cos(\theta_{n-1}) \end{pmatrix}
\end{aligned} \tag{9}$$

and (the same as (9), but with tildes)

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

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

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

$$\cdot \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \cos(\tilde{\theta}_{n-1}) & \sin(\tilde{\theta}_{n-1}) \\ 0 & 0 & 0 & -\sin(\tilde{\theta}_{n-1}) & \cos(\tilde{\theta}_{n-1}) \end{pmatrix}, \quad (10)$$

where θ_1, θ_2, $\ldots$, θ_{n-2}, θ_{n-1}, $\tilde{\theta}_1$, $\tilde{\theta}_2$, $\ldots$, $\tilde{\theta}_{n-2}$, $\tilde{\theta}_{n-1}$ are i.i.d. real random variables drawn uniformly from $[0, 2\pi]$. We observe that $\Theta_{n\times n}$, $\tilde{\Theta}_{n\times n}$, $\Pi_{n\times n}$, $\tilde{\Pi}_{n\times n}$, $Z_{n\times n}$, and $\tilde{Z}_{n\times n}$ are all unitary, and so $H_{n\times n}$ is also unitary.

We call the transform $T_{l\times n}$ an "SRFT" for lack of a better term.

The following technical lemma is a slight reformulation of Lemma 4.4 of [12].

Lemma 3.1. *Suppose that α and β are real numbers greater than 1, and l, m, and n are positive integers, such that*

$$n > l \geq \frac{\alpha^2 \beta}{(\alpha - 1)^2} m^2. \quad (11)$$

Suppose further that $T_{l\times n}$ is the SRFT defined in (6), and that $Q_{n\times m}$ is a matrix whose columns are orthonormal.

Then, the least singular value σ_m of $T_{l\times n} Q_{n\times m}$ satisfies

$$\sigma_m \geq \sqrt{\frac{l}{\alpha n}} \quad (12)$$

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

4 Description of the algorithm

Suppose that ε is a positive real number less than 1, and l, m, and n are positive integers with $m < l < n$. Suppose further that $A_{m\times n}$ is a full-rank matrix, $b_{m\times 1}$ is a vector, and $p_{n\times 1}$ is the vector of minimal Euclidean norm satisfying $A_{m\times n}\, p_{n\times 1} = b_{m\times 1}$. In order to construct a vector $x_{n\times 1}$ such that $\|x_{n\times 1} - p_{n\times 1}\| \leq \varepsilon \, \|p_{n\times 1}\|$ with high probability (increasingly high probability as a parameter $\alpha > 1$ increases), we compute the vector $c_{n\times 1}$ of minimal Euclidean norm that is a linear combination of random vectors and satisfies $A_{m\times n}\, c_{n\times 1} = b_{m\times 1}$, then use the algorithm of [9] to compute the orthogonal projection of $c_{n\times 1}$ onto the column span of $(A_{m\times n})^*$. More precisely, we perform the following five steps:

1. Construct the matrix
$$S_{l\times m} = T_{l\times n}\, (A_{m\times n})^*, \tag{13}$$
applying the SRFT $T_{l\times n}$ defined in (6) to every column of $(A_{m\times n})^*$ (see, for example, Subsection 3.3 of [12] for details on applying the SRFT rapidly).

2. Construct the vector $z_{l\times 1}$ of minimal Euclidean norm solving the system of linear equations
$$(S_{l\times m})^*\, z_{l\times 1} = b_{m\times 1}, \tag{14}$$
where $S_{l\times m}$ is the matrix defined in (13) (see, for example, Algorithm 5.7.2 in [8] for details on constructing $z_{l\times 1}$).

3. Construct the vector
$$c_{n\times 1} = (T_{l\times n})^*\, z_{l\times 1}, \tag{15}$$
where $z_{l\times 1}$ is the vector of minimal Euclidean norm solving (14), and $T_{l\times n}$ is the same realization of the SRFT as in (13) (see, for example, Subsection 3.3 of [12] for details on applying the adjoint of the SRFT rapidly).

4. Use the algorithm of [9] for the construction of a vector $y_{m\times 1}$ minimizing
$$\|(A_{m\times n})^*\, y_{m\times 1} - c_{n\times 1}\|^2 \tag{16}$$
to relative precision $\varepsilon^2 l/(\alpha n)$ or better, where $c_{n\times 1}$ is the vector defined in (15). The parameter l for the algorithm of [9] should be the same as for the present algorithm.

5. Construct the desired vector
$$x_{n\times 1} = (A_{m\times n})^*\, y_{m\times 1}, \tag{17}$$
where $y_{m\times 1}$ is the vector from Step 4.

Remark 4.1. In Step 2 above, we assume that $S_{l\times m}$ defined in (13) is a full-rank matrix. Lemma 3.1 in Section 3 above guarantees this with high probability when $l > m^2$, by taking α in the lemma arbitrarily large; numerical experiments indicate that $l > m$ suffices.

Remark 4.2. It is possible to improve the approximation $x_{n\times 1}$ via preconditioned conjugate gradient iterations similar to those proposed in [9]. However, the approximation produced by the above algorithm is already highly accurate (see, for example, Section 7 or Theorem 5.4 in Section 5 below), and further iterative improvement may double the running time of the algorithm.

5 Proof of accuracy

In this section, we prove Theorem 5.4, guaranteeing that the algorithm of Section 4 produces high accuracy with high probability.

The following lemma states that the orthogonal projection onto the column span of $(A_{m\times n})^*$ of the vector $c_{n\times 1}$ defined in (15) is the vector of minimal Euclidean norm that the algorithm aims to approximate.

Lemma 5.1. *Suppose that l, m, and n are positive integers with $m < l < n$. Suppose further that $A_{m\times n}$ is a full-rank matrix, $b_{m\times 1}$ is a vector, $S_{l\times m}$ is the matrix defined in (13) and is a full-rank matrix, and $c_{n\times 1}$ is the vector defined in (15).*

Then, the orthogonal projection $p_{n\times 1}$ of $c_{n\times 1}$ onto the column span of $(A_{m\times n})^*$ is the vector of minimal Euclidean norm satisfying

$$A_{m\times n}\, p_{n\times 1} = b_{m\times 1}. \tag{18}$$

Proof. Combining (13), (14), and (15) yields that

$$A_{m\times n}\, c_{n\times 1} = b_{m\times 1}. \tag{19}$$

Combining (19) and the fact that $p_{n\times 1}$ is the orthogonal projection of $c_{n\times 1}$ onto the column span of $(A_{m\times n})^*$ completes the proof. □

The following lemma states that, with high probability, the Euclidean norm of the vector $c_{n\times 1}$ defined in (15) is not too much greater than the Euclidean norm of its orthogonal projection $p_{n\times 1}$ onto the column span of $(A_{m\times n})^*$.

Lemma 5.2. *Suppose that α and β are real numbers greater than 1, and l, m, and n are positive integers, such that (11) holds. Suppose further that $A_{m\times n}$ is a full-rank matrix, $b_{m\times 1}$ is a vector, $c_{n\times 1}$ is the vector defined in (15), and $p_{n\times 1}$ is the orthogonal projection of $c_{n\times 1}$ onto the column span of $(A_{m\times n})^*$.*

Then,

$$\|c_{n\times 1}\| \leq \sqrt{\frac{\alpha n}{l}}\, \|p_{n\times 1}\| \tag{20}$$

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

Proof. Using the fact that $A_{m\times n}$ is a full-rank matrix, we construct a QR-decomposition

$$(A_{m\times n})^* = Q_{n\times m}\, R_{m\times m} \tag{21}$$

such that the columns of $Q_{n\times m}$ are an orthonormal basis for the column span of $(A_{m\times n})^*$. We first show that the SRFT $T_{l\times n}$ used in (13) and (15) provides

$$\|z_{l\times 1}\| \leq \sqrt{\frac{\alpha n}{l}}\, \|(Q_{n\times m})^*\, (T_{l\times n})^*\, z_{l\times 1}\| \tag{22}$$

with probability at least $1 - \frac{1}{\beta}$, where $z_{l\times 1}$ is the vector of minimal Euclidean norm solving (14). We then express the left- and right-hand sides of (22) in terms of $c_{n\times 1}$ and $p_{n\times 1}$, rather than $z_{l\times 1}$, in order to obtain (20).

It follows from the fact that $z_{l\times 1}$ is the vector of minimal Euclidean norm solving (14) that $z_{l\times 1}$ belongs to the column span of $S_{l\times m}$ from (14), that is, there exists a vector $w_{m\times 1}$ such that

$$z_{l\times 1} = S_{l\times m}\, w_{m\times 1}. \tag{23}$$

Combining (23), (13), and (21) yields that

$$\|z_{l\times 1}\|^2 = (z_{l\times 1})^*\, z_{l\times 1} = (w_{m\times 1})^*\, (R_{m\times m})^*\, (Q_{n\times m})^*\, (T_{l\times n})^*\, z_{l\times 1}. \tag{24}$$

The Cauchy-Schwarz inequality yields that

$$(w_{m\times 1})^*\, (R_{m\times m})^*\, (Q_{n\times m})^*\, (T_{l\times n})^*\, z_{l\times 1} \leq \|R_{m\times m}\, w_{m\times 1}\|\, \|(Q_{n\times m})^*\, (T_{l\times n})^*\, z_{l\times 1}\|. \tag{25}$$

It follows from (12) that

$$\|R_{m\times m}\, w_{m\times 1}\| \leq \sqrt{\frac{\alpha n}{l}}\, \|T_{l\times n}\, Q_{n\times m}\, R_{m\times m}\, w_{m\times 1}\| \tag{26}$$

with probability at least $1 - \frac{1}{\beta}$. Combining (21), (13), and (23) yields that

$$T_{l\times n}\, Q_{n\times m}\, R_{m\times m}\, w_{m\times 1} = z_{l\times 1}. \tag{27}$$

Combining (24), (25), (26), and (27) yields (22).

We now express the left- and right-hand sides of (22) in terms of $c_{n\times 1}$ and $p_{n\times 1}$, rather than $z_{l\times 1}$.

First, we consider the left-hand side of (22). Combining (15) and the fact that the columns of $(T_{l\times n})^*$ are orthonormal yields that

$$\|z_{l\times 1}\| = \|c_{n\times 1}\|. \tag{28}$$

Next, we consider the right-hand side of (22). It follows from the fact that the columns of $Q_{n\times m}$ are an orthonormal basis for the column span of $(A_{m\times n})^*$ that the orthogonal projection $p_{n\times 1}$ of $c_{n\times 1}$ onto the column span of $(A_{m\times n})^*$ is

$$p_{n\times 1} = Q_{n\times m}\, (Q_{n\times m})^*\, c_{n\times 1}. \tag{29}$$

Combining (29) and the fact that the columns of $Q_{n\times m}$ are orthonormal yields that

$$\|p_{n\times 1}\| = \|(Q_{n\times m})^*\, c_{n\times 1}\|. \tag{30}$$

Combining (30) and (15) yields that

$$\|p_{n\times 1}\| = \|(Q_{n\times m})^*\, (T_{l\times n})^*\, z_{l\times 1}\|. \tag{31}$$

Finally, combining (22), (28), and (31) yields (20). □

The following lemma states that the vector $x_{n\times 1}$ produced by the algorithm is an accurate approximation to the orthogonal projection onto the column span of $(A_{m\times n})^*$ of the vector $c_{n\times 1}$ defined in (15), provided that the projection $p_{n\times 1}$ satisfies (20).

Lemma 5.3. *Suppose that ε and α are positive real numbers with $\varepsilon < 1 < \alpha$, and l, m, and n are positive integers with $m < l < n$. Suppose further that $A_{m\times n}$ is a full-rank matrix, $b_{m\times 1}$ is a vector, $S_{l\times m}$ is the matrix defined in (13) and is a full-rank matrix, $c_{n\times 1}$ is the vector defined in (15), $p_{n\times 1}$ is the orthogonal projection of $c_{n\times 1}$ onto the column span of $(A_{m\times n})^*$, $y_{m\times 1}$ is a vector minimizing (16) to relative precision $\varepsilon^2 l/(\alpha n)$ or better, and $x_{n\times 1}$ is the vector defined in (17). Suppose in addition that (20) holds.*

Then,

$$\|x_{n\times 1} - p_{n\times 1}\| \leq \varepsilon \, \|p_{n\times 1}\|. \tag{32}$$

Proof. It follows from (17) that $x_{n\times 1}$ belongs to the column span of $(A_{m\times n})^*$. Combining this fact and the fact that $c_{n\times 1} - p_{n\times 1}$ is the orthogonal projection of $c_{n\times 1}$ onto the orthogonal complement of the column span of $(A_{m\times n})^*$ yields that $c_{n\times 1} - p_{n\times 1}$ is the orthogonal projection of $c_{n\times 1} - x_{n\times 1}$ onto the orthogonal complement of the column span of $(A_{m\times n})^*$. Similarly, $p_{n\times 1} - x_{n\times 1}$ is the orthogonal projection of $c_{n\times 1} - x_{n\times 1}$ onto the column span of $(A_{m\times n})^*$. We thus obtain the Pythagorean identity

$$\|c_{n\times 1} - p_{n\times 1}\|^2 + \|p_{n\times 1} - x_{n\times 1}\|^2 = \|c_{n\times 1} - x_{n\times 1}\|^2. \tag{33}$$

It follows from the fact that $p_{n\times 1}$ is the orthogonal projection of $c_{n\times 1}$ onto the column span of $(A_{m\times n})^*$ that the minimal value of (16) is $\|p_{n\times 1} - c_{n\times 1}\|^2$. Combining this fact, (17), and the fact that $y_{m\times 1}$ minimizes (16) to relative precision $\varepsilon^2 l/(\alpha n)$ or better yields that

$$\|x_{n\times 1} - c_{n\times 1}\|^2 - \|p_{n\times 1} - c_{n\times 1}\|^2 \leq \frac{\varepsilon^2 l}{\alpha n} \, \|p_{n\times 1} - c_{n\times 1}\|^2. \tag{34}$$

Combining (33) and (34) yields that

$$\|x_{n\times 1} - p_{n\times 1}\|^2 \leq \frac{\varepsilon^2 l}{\alpha n} \, \|c_{n\times 1} - p_{n\times 1}\|^2. \tag{35}$$

It follows from the fact that $c_{n\times 1} - p_{n\times 1}$ is the orthogonal projection of $c_{n\times 1}$ onto the orthogonal complement of the column span of $(A_{m\times n})^*$ that

$$\|c_{n\times 1} - p_{n\times 1}\| \leq \|c_{n\times 1}\|. \tag{36}$$

Combining (35), (36), and (20) yields (32). □

Combining Lemmas 3.1, 5.1, 5.2, and 5.3 yields the following theorem, guaranteeing that the algorithm produces high accuracy with high probability.

Theorem 5.4. *Suppose that ε, α, and β are positive real numbers with $\varepsilon < 1 < \alpha$ and $\beta > 2$, and l, m, and n are positive integers, such that (11) holds. Suppose further that $A_{m\times n}$ is a full-rank matrix, $b_{m\times 1}$ is a vector, and $p_{n\times 1}$ is the vector of minimal Euclidean norm satisfying*

$$A_{m\times n} \, p_{n\times 1} = b_{m\times 1}. \tag{37}$$

Then, the vector $x_{n\times 1}$ defined in (17) satisfies

$$\|x_{n\times 1} - p_{n\times 1}\| \leq \varepsilon \, \|p_{n\times 1}\| \tag{38}$$

with probability at least $1 - \frac{2}{\beta}$.

Remark 5.5. The probability of failure in Theorem 5.4 is $2/\beta$, rather than just the probability $1/\beta$ of failure in Lemma 5.2, in order to account for the possibility that the algorithm of [9] fails to produce a vector $y_{m\times 1}$ minimizing (16) to relative precision $\varepsilon^2 l/(\alpha n)$ or better, after using at most the number of floating-point operations specified in Section 6 below.

Remark 5.6. Empirically, requiring (11) appears to be excessive. In practice, choosing any $l \geq 4m$ and $\alpha = 4$ makes failure of the algorithm too improbable to detect (*cf.* Remark 2 in [9]).

6 Computational costs

In this section, we tabulate the numbers of floating-point operations required by the five steps in the algorithm of Section 4:

1. Applying $T_{l\times n}$ to every column of $(A_{m\times n})^*$ costs $\mathcal{O}(mn\,\log(l))$.
2. Constructing $z_{l\times 1}$ costs $\mathcal{O}(lm^2)$.
3. Applying $(T_{l\times n})^*$ to $z_{l\times 1}$ costs $\mathcal{O}(n\,\log(n))$.
4. Constructing $y_{l\times 1}$ via the algorithm of [9] costs $\mathcal{O}(mn\,\log(\alpha n/\varepsilon^2) + lm^2)$.
5. Applying $(A_{m\times n})^*$ to $y_{m\times 1}$ costs $\mathcal{O}(mn)$.

Summing up the costs in the five steps above, and using the facts that $m < l < n$ and $\varepsilon < 1 < \alpha$, we see that the cost of the entire algorithm is

$$C_{\text{theoretical}} = \mathcal{O}(mn\,\log(\sqrt{\alpha n}/\varepsilon) + lm^2) \tag{39}$$

floating-point operations. Theorem 5.4 in Section 5 above guarantees that the algorithm produces high accuracy with high probability when l and α satisfy (11). In practice, choosing $l = 4m$ and $\alpha = 4$ makes failure of the algorithm too improbable to detect (see also Remark 2 in [9]), and the cost in (39) then becomes

$$C_{\text{typical}} = \mathcal{O}(mn\,\log(\sqrt{n}/\varepsilon) + m^3) \tag{40}$$

floating-point operations.

7 Numerical Results

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

For various positive integers m and n with $m < n$, we use the algorithm to compute a vector $x_{n\times 1}$ approximating the vector $p_{n\times 1}$ of minimal Euclidean norm satisfying $A_{m\times n}\,p_{n\times 1} = b_{m\times 1}$, where $b_{m\times 1}$ is a vector, and $A_{m\times n}$ is the matrix defined via the formula

$$A_{m\times n} = U_{m\times m}\,\Sigma_{m\times m}\,(V_{n\times m})^*; \tag{41}$$

in the experiments described below, $U_{m\times m}$ is obtained by applying the Gram-Schmidt process to the columns of an $m\times m$ matrix whose entries are i.i.d. centered complex Gaussian random variables, $V_{n\times m}$ is obtained by applying the Gram-Schmidt process to the columns of an $n\times m$ matrix whose entries are i.i.d. centered complex Gaussian random variables, and $\Sigma_{m\times m}$ is a diagonal matrix, with the diagonal entries

$$\Sigma_{j,j} = 10^{-6(j-1)/(m-1)} \tag{42}$$

for $j = 1, 2, \ldots, m-1, m$. Clearly, the condition number κ_A of $A_{m\times n}$ is

$$\kappa_A = \Sigma_{1,1}/\Sigma_{m,m} = 10^6. \tag{43}$$

The vector $b_{m\times 1}$ is defined via the formula

$$b_{m\times 1} = A_{m\times n}\, p_{n\times 1}, \tag{44}$$

where $p_{n\times 1}$ is the vector defined via the formula

$$p_{n\times 1} = \frac{1}{\sqrt{m}} \sum_{j=1}^{m} \varepsilon_j\, v^{(j)}_{n\times 1}, \tag{45}$$

with $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_{m-1}, \varepsilon_m$ being pseudorandom positive and negative ones, and $v^{(1)}_{n\times 1}$, $v^{(2)}_{n\times 1}$, $\ldots$, $v^{(m-1)}_{n\times 1}$, $v^{(m)}_{n\times 1}$ being the columns of $V_{n\times m}$.

Remark 7.1. By construction, $p_{n\times 1}$ is the vector of minimal Euclidean norm such that $A_{m\times n}\, p_{n\times 1} = b_{m\times 1}$; the Euclidean norm of $p_{n\times 1}$ is minimal since $p_{n\times 1}$ belongs to the column span of $(A_{m\times n})^*$. The aim of the algorithm is to construct an approximation $x_{n\times 1}$ to $p_{n\times 1}$.

For the direct computations, we used the classical algorithm for pivoted QR-decompositions based on plane (Householder) reflections (see, for example, Chapter 5 in [8]). We implemented the algorithms in Fortran 77 in double-precision arithmetic, and used the Lahey/Fujitsu Express v6.2 compiler, with the optimization flag `--o2` enabled. We used one core of a 1.86 GHz Intel Centrino Core Duo microprocessor with 2 MB of L2 cache and 1 GB of RAM. For the algorithm of [9] used in Step 4 of the algorithm of Section 4, we requested that $y_{m\times 1}$ minimize (16) to relative precision $(10^{-14}\cdot\kappa_A)^2\cdot m/n$ or better, where κ_A is the condition number of $A_{m\times n}$ given in (43). We used a double-precision version of P. N. Swarztrauber's FFTPACK library for fast Fourier transforms.

Table 1 displays timing results with $n = 16384$ for various values of m; Table 2 displays the corresponding errors. Table 3 displays timing results with $m = 256$ for various values of n; Table 4 displays the corresponding errors.

The headings of the tables have the following meanings:

- m is the number of rows in the matrix $A_{m\times n}$, as well as the length of the vector $b_{m\times 1}$, in $A_{m\times n}\, p_{n\times 1} = b_{m\times 1}$.
- n is the number of columns in the matrix $A_{m\times n}$, as well as the length of the vector $p_{n\times 1}$, in $A_{m\times n}\, p_{n\times 1} = b_{m\times 1}$.

Table 1

m	n	l	t_0	t_{r}	t_0/t_{r}
128	16384	512	.27E1	.24E1	1.2
256	16384	1024	.11E2	.56E1	2.0
512	16384	2048	.60E2	.20E2	3.0

Table 2

m	n	l	ε_0	ε_{r}
128	16384	512	.14E–14	.16E–14
256	16384	1024	.11E–14	.17E–14
512	16384	2048	.80E–15	.29E–14

Table 3

m	n	l	t_0	t_{r}	t_0/t_{r}
256	4096	1024	.26E1	.20E1	1.3
256	8192	1024	.51E1	.32E1	1.6
256	16384	1024	.11E2	.56E1	2.0
256	32768	1024	.29E2	.16E2	2.5

Table 4

m	n	l	ε_0	ε_{r}
256	4096	1024	.27E–15	.31E–14
256	8192	1024	.45E–15	.27E–14
256	16384	1024	.11E–14	.17E–14
256	32768	1024	.22E–14	.16E–14

- l is the number of rows in the matrix $T_{l\times n}$ used in Steps 1 and 3 of the algorithm of Section 4, as well as the analogous parameter used in the algorithm of [9] needed in Step 4.
- t_0 is the time in seconds required by the direct, classical algorithm.
- t_{r} is the time in seconds required by the algorithm of the present paper.
- t_0/t_{r} is the factor by which the algorithm of the present paper is faster than the classical algorithm.
- ε_0 is defined via the formula

$$\varepsilon_0 = \frac{\|x^{(0)}_{n\times 1} - p_{n\times 1}\|}{\kappa_A \, \|p_{n\times 1}\|}, \tag{46}$$

  where κ_A is the condition number of $A_{m\times n}$ given in (43), and $x^{(0)}_{n\times 1}$ is the vector produced by the direct, classical algorithm approximating the vector $p_{n\times 1}$ of minimal Euclidean norm such that $A_{m\times n}\, p_{n\times 1} = b_{m\times 1}$.
- ε_{r} is defined via the formula

$$\varepsilon_{\mathrm{r}} = \frac{\|x_{n\times 1} - p_{n\times 1}\|}{\kappa_A \, \|p_{n\times 1}\|}, \tag{47}$$

  where κ_A is the condition number of $A_{m\times n}$ given in (43), and $x_{n\times 1}$ is the vector produced by the algorithm of Section 4 approximating the vector $p_{n\times 1}$ of minimal Euclidean norm such that $A_{m\times n}\, p_{n\times 1} = b_{m\times 1}$.

Remark 7.2. Standard perturbation theory shows that ε_0 and ε_{r} are the appropriately normalized measures of the relative precision produced by the algorithms; see, for example, Section 5.5.3 in [5].

The values for ε_{r} reported in the tables are the worst (maximal) values encountered during 10 independent randomized trials of the algorithm, as applied to the same matrix $A_{m\times n}$ and vector $b_{m\times 1}$. The values for t_{r} 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 $m = 512$, $n = 16384$, and the condition number of $A_{m\times n}$ is 10^6, the randomized algorithm runs 3 times faster than the classical algorithm based on plane (Householder) reflections, even at full double precision.

2. Our choice $l = 4m$ appears to make failure of the algorithm too improbable to detect.

3. The algorithm of the present article reliably produces high precision at reasonably low cost.

8 Conclusion

This article provides a fast algorithm for computing the minimal-norm solution to an underdetermined system of linear equations. If the matrices $A_{m\times n}$ and $(A_{m\times n})^*$ associated with the system of linear equations can be applied sufficiently rapidly to arbitrary vectors, then the algorithm of the present paper can be accelerated further.

The theoretical bounds in Lemma 3.1, Lemma 5.2, and Theorem 5.4 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 (6). In our experience, several other similar transforms seem to work at least as well, and we are investigating these alternatives (see, for example, [2]).

Acknowledgments

We would like to thank Vladimir Rokhlin and Arthur Szlam for helpful discussions.

References

[1] N. Ailon and B. Chazelle, *The fast Johnson-Lindenstrauss transform and approximate nearest neighbors*, SIAM J. Comput., 39 (2009), pp. 302–322.

[2] N. Ailon and E. Liberty, *Fast dimension reduction using Rademacher series on dual BCH codes*, Discrete Comput. Geom., (2008). To appear.

[3] A. M. Bruckstein, D. L. Donoho, and M. Elad, *From sparse solutions of systems of equations to sparse modeling of signals and images*, SIAM Rev., 51 (2009), pp. 34–81.

[4] E. Candès and T. Tao, *Near-optimal signal recovery from random projections: Universal encoding strategies*, IEEE Trans. Inform. Theory, 52 (2006), pp. 5406–5425.

[5] G. Dahlquist and Å. Björck, *Numerical Methods*, Dover Publications, Mineola, New York, 1974.

[6] D. Donoho, *Compressed sensing*, IEEE Trans. Inform. Theory, 52 (2006), pp. 1289–1306.

[7] P. Drineas, M. W. Mahoney, S. Muthukrishnan, and T. Sarlós, *Faster least squares approximation*, Tech. Rep. 0710.1435, arXiv, October 2007. Available at http://arxiv.org/.

[8] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, Maryland, third ed., 1996.

[9] V. Rokhlin and M. Tygert, *A fast randomized algorithm for overdetermined linear least-squares regression*, Proc. Natl. Acad. Sci. USA, 105 (2008), pp. 13212–13217.

[10] T. Sarlós, *Improved approximation algorithms for large matrices via random projections*, in Proceedings of FOCS 2006, the 47th Annual IEEE Symposium on Foundations of Computer Science, October 2006, pp. 143–152.

[11] ——, *Improved approximation algorithms for large matrices via random projections, revised, extended long form.* Manuscript in preparation for publication, currently available at http://www.ilab.sztaki.hu/∼stamas/publications/rp-long.pdf, 2006.

[12] F. Woolfe, E. Liberty, V. Rokhlin, and M. Tygert, *A fast randomized algorithm for the approximation of matrices*, Appl. Comput. Harmon. Anal., 25 (2008), pp. 335–366.