

Faster Least Squares Approximation

Petros Drineas^{*} Michael W. Mahoney[†] S. Muthukrishnan[‡] Tamás Sarlós[§]

Abstract

Least squares approximation is a technique to find an approximate solution to a system of linear equations that has no exact solution. In a typical setting, one lets n be the number of constraints and d be the number of variables, with $n \gg d$. Then, methods dating back to Gauss and Legendre find a solution in $O(nd^2)$ time. We present two randomized algorithms that provide very accurate relative-error approximations to the solution of a least squares approximation problem more rapidly than existing exact algorithms. Both of our algorithms preprocess the data with a randomized Hadamard transform. One then uniformly randomly samples constraints and solves the smaller problem on those constraints, and the other performs a sparse random projection and solves the smaller problem on those projected coordinates. In both cases, the solution to the smaller problem provides a relative-error approximation to the exact solution; and if n is sufficiently larger than d , the approximate solution can be computed in $o(nd^2)$ time.

1 Introduction

In many applications in mathematics and statistical data analysis, it is of interest to find an approximate solution to a system of linear equations that has no exact solution. For example, let a matrix $A \in \mathbb{R}^{n \times d}$ and a vector $b \in \mathbb{R}^n$ be given. If $n \gg d$, there will not in general exist a vector $x \in \mathbb{R}^d$ such that $Ax = b$, and yet it is often of interest to find a vector x such that $Ax \approx b$ in some precise sense. The method of least squares accomplishes this by minimizing the sum of squares of the elements of the residual vector, i.e., by solving the optimization problem

$$\mathcal{Z} = \min_{x \in \mathbb{R}^d} \|Ax - b\|_2. \quad (1)$$

It is well-known that the minimum-length vector among those satisfying (1) may be computed as

$$x_{opt} = A^+b, \quad (2)$$

where A^+ denotes the Moore-Penrose generalized inverse of the matrix A [4, 11]. This solution has a very natural statistical interpretation as providing an optimal estimator among all linear unbiased estimators, and it has a very natural geometric interpretation as providing an orthogonal projection of the vector b onto the span of the columns of the matrix A .

Techniques dating back to Gauss and Legendre, who originally formulated the method of least squares [19], find a solution to (1) in $O(nd^2)$ time. In particular, recall that to minimize the quantity in Equation (1), we can set the derivative of $\|Ax - b\|_2^2 = (Ax - b)^T(Ax - b)$ with

^{*}Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, drinep@cs.rpi.edu.

[†]Department of Mathematics, Stanford University, Stanford, CA, mmahoney@cs.stanford.edu.

[‡]Google, Inc., New York, NY, muthu@google.com.

[§]Yahoo! Research, Sunnyvale, CA, stamas@yahoo-inc.com.

respect to x equal to zero, from which it follows that the minimizing vector x_{opt} is a solution of the so-called normal equations

$$A^T A x_{opt} = A^T b. \quad (3)$$

Geometrically, this means that the residual vector $b^\perp = b - Ax_{opt}$ is required to be orthogonal to the column space of A , i.e., $b^{\perp T} A = 0$. If the matrix $A^T A$ has full rank and is well-conditioned, the normal equations can be solved directly by using the Cholesky decomposition $A^T A = R^T R$, giving $R^T R x_{opt} = A^T b$, where R is an upper triangular matrix. A second solution method, which is more numerically stable if A is ill-conditioned or rank-deficient, is to compute a QR decomposition. In this case, if $A = QR$, where Q is an orthogonal matrix and R is an upper triangular matrix, then one can solve $R x_{opt} = Q^T b$. A third alternative, which is particularly useful if A is very ill-conditioned, is to use the Singular Value Decomposition (SVD). If $A = U_A \Sigma_A V_A^T$ denotes the SVD of A , then

$$x_{opt} = V_A \Sigma_A^{-1} U_A^T b = A^+ b.$$

Although this SVD method is the most computationally intensive (by constant factors), these and other exact methods of solution all require $\Omega(nd^2)$ computation time [11]. For descriptions of some of the many widely-used variants of these basic algorithms to solve the least squares approximation problem, see, e.g., Golub and Van Loan [11].

1.1 Our results

In this paper, we describe two randomized algorithms that will provide very accurate relative-error approximations to the exact solution \mathcal{Z} and the optimal vector x_{opt} faster than existing exact algorithms for a large class of overconstrained least-squares problems. In particular, we will prove the following theorem.

Theorem 1 *Suppose $A \in \mathbb{R}^{n \times d}$, $b \in \mathbb{R}^n$, and let $\epsilon \in (0, 1]$. Then, there exists a randomized algorithm such that, with probability at least $3/4$, the following three claims hold: first, the computed solution \tilde{x}_{opt} satisfies*

$$|A\tilde{x}_{opt} - b|_2 \leq (1 + \epsilon)\mathcal{Z}; \quad (4)$$

second, if $\kappa(A)$ is the condition number of A and if we assume that $\gamma \in [0, 1]$ is the fraction of the “weight” of b that lies inside the column space of A , then \tilde{x}_{opt} satisfies

$$|x_{opt} - \tilde{x}_{opt}|_2 \leq \sqrt{\epsilon} \left(\kappa(A) \sqrt{\gamma^{-2} - 1} \right) |x_{opt}|_2; \quad (5)$$

and third, the solution \tilde{x}_{opt} can be computed in $o(nd^2)$ time if n is sufficiently larger than d and less than 2^d .

We will discuss the exact assumptions on n below, and we will provide a precise statement of the running time for our two algorithms (including the ϵ -dependence) in Theorem 2 (Section 3) and Theorem 3 (Section 4), respectively. It is worth noting that the claims of Theorem 1 can be made to hold with probability $1 - \delta$, for any $\delta > 0$, by running the algorithm $O(\log(1/\delta))$ times.

Here, we provide a brief overview of our main algorithms. Let $\mathcal{H} = HD$ denote an $n \times n$ randomized Hadamard transform that is the product of H and D , where the $n \times n$ matrix H denotes a normalized deterministic Hadamard transform and the $n \times n$ random diagonal matrix D has diagonal elements drawn uniformly from $\{-1, +1\}$. This transform is described more precisely in Sections 2 and 3.3, and it was used recently as one step in the development of a “fast” version of the Johnson-Lindenstrauss lemma [1, 13]. Our first algorithm is a random sampling algorithm. After multiplying the input A and b by \mathcal{H} , this algorithm uniformly randomly samples

$r = O(d \log(n) \log(d \log n) / \epsilon)$ constraints from the preprocessed problem. Then, this algorithm solves the least squares problem on just those sampled constraints to obtain a number \tilde{Z} and a d -vector \tilde{x}_{opt} such that the three claims of Theorem 1 are satisfied. In particular, we will first apply an $n \times n$ randomized Hadamard transform \mathcal{H} to the matrix A and vector b . Since we will actually sample only $O(r)$ of the constraints from the Hadamard-preprocessed problem, we will spend only $O(nd \log r)$ time at this step [2]. Then, of course, solving the $r \times d$ sampled least squares problem will require only $O(rd^2)$ time. Assuming that ϵ is a constant independent of n and d , if $\log n \leq d$ and $n / \log n = \omega(d \log d)$, then the running time of this algorithm is $o(nd^2)$.

In a similar manner, our second algorithm also initially transforms the input A and b by \mathcal{H} . This algorithm then multiplies the result by a $k \times n$, where $k = O(d/\epsilon)$, “sparse projection” matrix \mathcal{T} . (This matrix \mathcal{T} is described in detail in Section 4.2. Its construction depends on a sparsity parameter, and it is identical to the “sparse projection” matrix in Matoušek’s version of the Ailon-Chazelle result [1, 13].) Finally, our second algorithm solves the least squares problem on just those k coordinates to obtain a number \tilde{Z} and a d -vector \tilde{x}_{opt} such that the three claims of Theorem 1 are satisfied. In particular, assuming that ϵ is a constant independent of n and d , if $\log n \leq d$ and $n / \log^2 n = \omega(d^2)$, then the running time of this algorithm is $o(nd^2)$.

Although the running time of our second algorithm is worse than the running time of our first algorithm (at least according to our analysis), we include both algorithms in this paper for the sake of completeness and since improved bounds for the fast Johnson-Lindenstrauss transform might lead to improved bounds on the running time of our second algorithm. Clearly, an interesting open problem is to relax the above constraints on n for either of the proposed algorithms.

1.2 Preconditioning and related work

Both of our algorithms may be viewed as preconditioning the input matrix A and the target vector b with a carefully-constructed data-independent random matrix \mathcal{X} . For our random sampling algorithm, we let $\mathcal{X} = \mathcal{S}\mathcal{H}$, where \mathcal{S} is a matrix that represents the sampling operation, while for our random projection algorithm, we let $\mathcal{X} = \mathcal{T}\mathcal{H}$. Thus, we replace the least squares approximation problem (1) with the least squares approximation problem

$$\tilde{Z} = \min_{x \in \mathbb{R}^d} |\mathcal{X}(Ax - b)|_2, \quad (6)$$

and we explicitly compute the solution to (6) using a traditional deterministic algorithm [11], e.g., by using the SVD to compute the generalized inverse

$$\tilde{x}_{opt} = (\mathcal{X}A)^+ \mathcal{X}b. \quad (7)$$

Alternatively, one could use standard iterative methods such as the the Conjugate Gradient Normal Residual method (CGNR, see [11] for details), which can produce an ϵ -approximation to the optimal solution of (6) in $O(\kappa(\mathcal{X}A)rd \log(1/\epsilon))$ time, where $\kappa(\mathcal{X}A)$ is the condition number of $\mathcal{X}A$, and r is the number of rows of $\mathcal{X}A$ (see Section 1.1).

We should note several lines of related work. First, techniques such as the “method of averages” [7] preprocess the input into the form (6) and can be used to obtain exact or approximate solutions to the least squares problem (1) in $o(nd^2)$ time under strong statistical assumptions on A and b . To the best of our knowledge, however, the two algorithms we present and analyze are the first algorithms to provide nontrivial approximation guarantees for overconstrained least squares approximation problems in $o(nd^2)$ time,¹ while making no assumptions on the input

¹We should mention that Ibarra, Moran, and Hui provide a reduction of the least squares approximation problem to the matrix multiplication problem; in particular, they show that $MM(d)O(n/d)$ time, where $MM(d)$ is the time

data. Second, in parallel with our work and motivated by our preliminary results as reported in [8] and [17], Rokhlin and Tygert leveraged similar techniques to analyze theoretically and evaluate empirically random-projection based algorithms for overconstrained least-squares problems in [15]. Third, very recently, Clarkson and Woodruff proved space lower bounds on related problems [6], and Nguyen, Do, and Tran achieved a small improvement in the sampling complexity for related problems [14].

1.3 Outline

After a brief review of relevant background in Section 2, we present our main sampling-based algorithm for approximating least squares approximation in $o(nd^2)$ time in Section 3. Then, in Section 4 we present a second projection-based algorithm that also can be used to approximate least squares approximation in $o(nd^2)$ time. Preliminary versions of parts of this paper have appeared as conference proceedings in the 17th ACM-SIAM Symposium on Discrete Algorithms [8] and in the 47th IEEE Symposium on Foundations of Computer Science [17]. In particular, the core of our analysis in this paper was introduced in [8], where an expensive-to-compute probability distribution was used to construct a relative-error approximation sampling algorithm for the least squares approximation problem. Then, after the development of the Fast Johnson-Lindenstrauss transform [1],[17] proved that similar ideas could be used to improve the running time of randomized algorithms for the least squares approximation problem. In this paper, we have combined these ideas, treated the two algorithms in a manner to highlight their similarities and differences, and considerably simplified the analysis.

2 Notation and background

In this section, we first review several relevant definitions and facts from linear algebra; for more details, see [18, 11, 5, 4]. Let $[n]$ denote the set $\{1, 2, \dots, n\}$. For any matrix $A \in \mathbb{R}^{n \times d}$, let $A_{(i)}, i \in [n]$ denote the i -th row of A as a row vector, and let $A^{(j)}, j \in [d]$ denote the j -th column of A as a column vector. Let the rank of A be $\rho \leq \min\{n, d\}$. The Singular Value Decomposition (SVD) of A is denoted by $A = U_A \Sigma_A V_A^T$, where $U_A \in \mathbb{R}^{n \times \rho}$ is the matrix of left singular vectors, $\Sigma_A \in \mathbb{R}^{\rho \times \rho}$ is the diagonal matrix of singular values, and $V_A \in \mathbb{R}^{d \times \rho}$ is the matrix of right singular vectors. Let $\sigma_i(A), i \in [\rho]$, denote the i -th singular value of A , and $\sigma_{\max}(A)$ and $\sigma_{\min}(A)$ denote the maximum and minimum singular value of A . The condition number of A is $\kappa(A) = \sigma_{\max}(A)/\sigma_{\min}(A)$. The Moore-Penrose generalized inverse, or pseudoinverse, of A may be expressed in terms of the SVD as $A^+ = V_A \Sigma_A^{-1} U_A^T$ [4]. Finally, for any orthogonal matrix $U \in \mathbb{R}^{n \times \ell}$, let $U^\perp \in \mathbb{R}^{n \times (n-\ell)}$ denote an orthogonal matrix whose columns are an orthonormal basis spanning the subspace of \mathbb{R}^n that is orthogonal to the column space of U . In terms of U_A^\perp , the solution of the least squares approximation problem (1) is

$$\mathcal{Z} = \min_{x \in \mathbb{R}^d} \|Ax - b\|_2 = \left\| U_A^\perp U_A^\perp{}^T b \right\|_2.$$

Finally, let $\|A\|_F^2 = \sum_{i=1}^n \sum_{j=1}^d A_{ij}^2$ denote the square of the Frobenius norm of A , and let $\|A\|_2 = \sup_{x \in \mathbb{R}^d, x \neq 0} \|Ax\|_2 / \|x\|_2$ denote the spectral norm of A .

For simplicity, throughout this paper we will assume that n is a power of two and that the rank of the $n \times d$ matrix A equals d ; padding A and b with all-zero rows suffices to remove the first

needed to multiply two $d \times d$ matrices, is sufficient to solve this problem [12]. As will be clear from our analysis, our algorithms are fundamentally based on matrix multiplication as well. Note, however, that all of the running times we report in this paper assume the use of standard matrix multiplication algorithms, since $o(d^3)$ matrix multiplication algorithms are almost never used in practice.

assumption, whereas a simple modification of our analysis would remove the second assumption. In addition, in an effort to simplify the presentation, in this paper we make no effort to optimize constants. Finally, recall that the (non-normalized) $n \times n$ matrix of the Hadamard transform H_n may be defined recursively as follows:

$$H_n = \begin{bmatrix} H_{n/2} & H_{n/2} \\ H_{n/2} & -H_{n/2} \end{bmatrix}, \quad \text{with} \quad H_2 = \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix}.$$

The $n \times n$ normalized matrix of the Hadamard transform is equal to $\frac{1}{\sqrt{n}}H_n$; hereafter, we will denote this normalized matrix by H . Computing the product Hx for any vector $x \in \mathbb{R}^n$ takes $O(n \log n)$ time, but if (as will be the case) we only need to access, say, r elements in the transformed vector, then those r elements can be computed in $O(n \log r)$ time [2].

3 A sampling-based randomized algorithm

In this section, we present a randomized sampling algorithm to provide a very accurate relative-error approximation to the solution of the least squares approximation problem (1) in $o(nd^2)$ time. We also state and prove an associated quality-of-approximation theorem.

3.1 The main algorithm and main theorem

Algorithm 1 takes as input an $n \times d$ matrix A , an n -vector b , and an error parameter $\epsilon \in (0, 1]$. This algorithm starts by preprocessing the matrix A and right hand side vector b with a randomized Hadamard transform. It then constructs a smaller problem by uniformly randomly sampling a small number of constraints from the preprocessed problem. Our main quality-of-approximation theorem—see Theorem 2 below—will state that with constant probability (over the random decisions made by the algorithm) the vector \tilde{x}_{opt} returned by this algorithm will satisfy relative-error bounds of the form (4) and (5) and will be computed in $o(nd^2)$ time.

In more detail, after preprocessing with the randomized Hadamard transform, as described in Section 2, Algorithm 1 will uniformly randomly sample $r = O(d \log(n) \log(d \log n)/\epsilon)$ constraints from the preprocessed least squares problem, it will rescale each sampled constraint by $\sqrt{n/r}$, and it will solve the least squares problem induced on just those sampled and rescaled constraints. (Note that the algorithm explicitly computes only those rows of $\mathcal{H}[Ab]$ that need to be accessed.) More formally, we will let \mathcal{S} denote a “sampling matrix” specifying which of the n constraints are to be sampled and how they are to be rescaled. In particular, fix $r = O(d \log(n) \log(d \log n)/\epsilon)$ and let \mathcal{S}' be a diagonal matrix in which the i^{th} diagonal entry equals $\sqrt{n/r}$ with probability r/n and equals 0 otherwise, and let \mathcal{S} be derived from \mathcal{S}' by deleting every row in which the diagonal element equals 0. If \mathcal{S} has more than $10r$ rows, then the algorithm aborts; we will provide a bound on this failure event in our analysis. Then, we can consider the problem

$$\tilde{\mathcal{Z}} = \min_{x \in \mathbb{R}^d} |\mathcal{S}\mathcal{H}Ax - \mathcal{S}\mathcal{H}b|_2,$$

which is just a least squares approximation problem involving the r constraints sampled from a Hadamard-preprocessed version of (1). The minimum-length vector $\tilde{x}_{opt} \in \mathbb{R}^d$ among those that achieve the minimum value $\tilde{\mathcal{Z}}$ in this problem is

$$\tilde{x}_{opt} = (\mathcal{S}\mathcal{H}A)^+ \mathcal{S}\mathcal{H}b.$$

Since Algorithm 1 samples a number of rows $r \ll n$ from the original problem, it will compute \tilde{x}_{opt} exactly. Our main quality-of-approximation results for Algorithm 1 are presented in the following theorem; the remainder of this section will be devoted to its proof.

Input: $A \in \mathbb{R}^{n \times d}$, $b \in \mathbb{R}^n$, and an error parameter $\epsilon \in (0, 1]$.

Output: $\tilde{x}_{opt} \in \mathbb{R}^d$.

1. Let $r = O(d \log(n) \log(d \log n)/\epsilon)$.
2. Let $\mathcal{S}' \in \mathbb{R}^{n \times n}$ be a diagonal matrix with

$$\mathcal{S}'_{ii} = \begin{cases} \sqrt{n/r} & , \text{ with probability } r/n \\ 0 & , \text{ with probability } 1 - r/n \end{cases}$$

3. Construct the sampling matrix \mathcal{S} by deleting the all-zeros rows of \mathcal{S}' .
4. If \mathcal{S} has more than $10r$ rows, then abort.
5. Let $H \in \mathbb{R}^{n \times n}$ be the normalized Hadamard transform matrix.
6. Let $D \in \mathbb{R}^{n \times n}$ be a diagonal matrix with

$$D_{ii} = \begin{cases} +1 & , \text{ with probability } 1/2 \\ -1 & , \text{ with probability } 1/2 \end{cases}$$

7. Let $\mathcal{H} = HD$. Compute and return $\tilde{x}_{opt} = (\mathcal{S}\mathcal{H}A)^+ \mathcal{S}\mathcal{H}b$.

Algorithm 1: A fast random sampling algorithm for least squares approximation

Theorem 2 Suppose $A \in \mathbb{R}^{n \times d}$, $b \in \mathbb{R}^n$, and let $\epsilon \in (0, 1]$. Run Algorithm 1, and return \tilde{x}_{opt} as an approximation to the solution of the least squares problem (1). Then, with probability at least $3/4$, the following three claims hold: first, \tilde{x}_{opt} satisfies

$$\|A\tilde{x}_{opt} - b\|_2 \leq (1 + \epsilon)\mathcal{Z};$$

second, if we assume that $\|U_A U_A^T b\|_2 \geq \gamma \|b\|_2$ for some $\gamma \in (0, 1]$ then \tilde{x}_{opt} satisfies

$$\|x_{opt} - \tilde{x}_{opt}\|_2 \leq \sqrt{\epsilon} \left(\kappa(A) \sqrt{\gamma^{-2} - 1} \right) \|x_{opt}\|_2;$$

and third,

$$O \left(nd \log(d \log(n)/\epsilon) + d^3 \log(n) \log(d \log n)/\epsilon \right)$$

time suffices to compute the solution \tilde{x}_{opt} .

Remark: Assuming that ϵ is a constant independent of n and d , Theorem 2 implies $o(nd^2)$ running times for a wide class of over-constrained least-squares problems. In particular, if $\log n \leq d$ and $n/\log n = \omega(d \log d)$, then the running time of the above algorithm is $o(nd^2)$. In words, as long as n is sufficiently larger than d and does not exceed 2^d , then the above algorithm improves upon the running time of standard least-squares algorithms.

Remark: Consider, for example, the special case where $\log n \leq d$ and $n/\log n = \Omega(d^2 \log d)$. Then, the running time of the above algorithm becomes $O(nd \log d)$.

3.2 A structural result sufficient for relative-error approximation

In this subsection, we will state and prove a lemma that establishes sufficient conditions on any matrix \mathcal{X} such that the solution \tilde{x}_{opt} to the least squares problem (6) will satisfy relative-error bounds of the form (4) and (5). (We will then show that the matrix \mathcal{SH} constructed by Algorithm 1 satisfies these conditions.) Recall that the SVD of A is $A = U_A \Sigma_A V_A^T$. In addition, for notational simplicity, we let $b^\perp = U_A^\perp U_A^{\perp T} b$ denote the part of the right hand side vector b lying outside of the column space of A .

The two conditions that we will require of the matrix \mathcal{X} are:

$$\sigma_{min}(\mathcal{X}U_A) \geq 9/10; \text{ and} \quad (8)$$

$$\left| U_A^T \mathcal{X}^T \mathcal{X} b^\perp \right|_2^2 \leq \epsilon \mathcal{Z}^2 / 2, \quad (9)$$

for some $\epsilon \in [0, 1]$. Several things should be noted about these conditions and the following lemma. First, although Condition (9) depends on the right hand side vector b , Algorithm 1 (as well as Algorithm 2 in Section 4.1 below) will satisfy it without using any information from b . Second, although Condition (8) only states that $\sigma_i(\mathcal{X}U_A) > 9/10$, for all $i \in [d]$, for both of our randomized algorithms we will show that $|\sigma_i(\mathcal{X}U_A) - 1| \leq 1/10$, for all $i \in [d]$. In fact, by slightly increasing the parameter r determining the number of rows sampled in Algorithm 1—or similarly the projection density parameter q of Algorithm 2—it also holds that $|\sigma_i(\mathcal{X}U_A) - 1| \leq \sqrt{\epsilon}$, for all $i \in [d]$. Thus, one should think of $\mathcal{X}U_A$ as an approximate isometry. Third, Condition (9) simply states that $\mathcal{X}b^\perp = \mathcal{X}U_A^\perp U_A^{\perp T} b$ remains approximately orthogonal to $\mathcal{X}U_A$. Finally, note that the following lemma is a deterministic statement, since it makes no explicit reference to either of our randomized algorithms. Failure probabilities will enter below when we show that our randomized algorithms satisfy Conditions (8) and (9).

Lemma 1 *Consider the overconstrained least squares approximation problem (1), and let the $n \times d$ matrix U_A contain the the top d left singular vectors of A . Assume that the matrix \mathcal{X} satisfies Conditions (8) and (9) above, for some $\epsilon \in (0, 1]$. Then, the solution \tilde{x}_{opt} to the least squares approximation problem (6) satisfies:*

$$|A\tilde{x}_{opt} - b|_2 \leq (1 + \epsilon)\mathcal{Z}, \text{ and} \quad (10)$$

$$|x_{opt} - \tilde{x}_{opt}|_2 \leq \frac{1}{\sigma_{min}(A)} \sqrt{\epsilon} \mathcal{Z}. \quad (11)$$

Proof: Let us first rewrite the down-scaled regression problem induced by \mathcal{X} as

$$\min_{x'} |\mathcal{X}b - \mathcal{X}Ax'|_2^2 = \min_{y'} \left| \mathcal{X}(Ax_{opt} + b^\perp) - \mathcal{X}A(x_{opt} + y') \right|_2^2 \quad (12)$$

$$\begin{aligned} &= \min_{y'} \left| \mathcal{X}b^\perp - \mathcal{X}Ay' \right|_2^2 \\ &= \min_{z'} \left| \mathcal{X}b^\perp - \mathcal{X}U_A z' \right|_2^2, \end{aligned} \quad (13)$$

where (12) follows since $b = Ax_{opt} + b^\perp$ and (13) follows since the columns of the matrix A span the same subspace as the columns of U_A . Now, define the vector z such that $U_A z = A(x_{opt} - \tilde{x}_{opt})$, and note that z is an optimal solution of (13). The latter fact follows since

$$\left| \mathcal{X}b^\perp - \mathcal{X}A(x_{opt} - \tilde{x}_{opt}) \right|_2^2 = \left| \mathcal{X}b^\perp - \mathcal{X}(b - b^\perp) + \mathcal{X}A\tilde{x}_{opt} \right|_2^2 = \left| \mathcal{X}A\tilde{x}_{opt} - \mathcal{X}b \right|_2^2.$$

Thus, by the normal equations (3), we have that

$$(\mathcal{X}U_A)^T \mathcal{X}U_A z = (\mathcal{X}U_A)^T \mathcal{X}b^\perp. \quad (14)$$

Taking the norm of both sides and observing that under Condition (8) we have $\sigma_i((\mathcal{X}U_A)^T \mathcal{X}U_A) = \sigma_i^2(\mathcal{X}U_A) \geq 1/\sqrt{2}$, for all i , it follows that

$$|z|_2^2/2 \leq |(\mathcal{X}U_A)^T \mathcal{X}U_A z|_2^2 = |(\mathcal{X}U_A)^T \mathcal{X}b^\perp|_2^2. \quad (15)$$

Upper bounding the r.h.s. of (15) with Condition (9) we observe that

$$|z|_2^2 \leq \epsilon \mathcal{Z}^2. \quad (16)$$

To establish the first claim of the lemma, let us rewrite the norm of the residual vector as

$$\begin{aligned} |b - A\tilde{x}_{opt}|_2^2 &= |b - Ax_{opt} + Ax_{opt} - A\tilde{x}_{opt}|_2^2 \\ &= |b - Ax_{opt}|_2^2 + |Ax_{opt} - A\tilde{x}_{opt}|_2^2 \end{aligned} \quad (17)$$

$$= \mathcal{Z}^2 + |U_A z|_2^2 \quad (18)$$

$$\leq \mathcal{Z}^2 + \epsilon \mathcal{Z}^2, \quad (19)$$

where (17) follows by Pythagoras, since $b - Ax_{opt} = b^\perp$, which is orthogonal to A , and consequently to $A(x_{opt} - \tilde{x}_{opt})$; (18) follows by the definition of z and \mathcal{Z} ; and (19) follows by (16) and since the Euclidean norm is unitarily invariant. The first claim of the lemma follows since $\sqrt{1 + \epsilon} \leq 1 + \epsilon$.

To establish the second claim of the lemma, recall that $A(x_{opt} - \tilde{x}_{opt}) = U_A z$. If we take the norm of both sides of this expression, we have that

$$|x_{opt} - \tilde{x}_{opt}|_2^2 \leq \frac{|U_A z|_2^2}{\sigma_{min}^2(A)} \quad (20)$$

$$\leq \frac{\epsilon \mathcal{Z}^2}{\sigma_{min}^2(A)}, \quad (21)$$

where (20) follows since $\sigma_{min}(A)$ is the smallest singular value of A and since the rank of A is d ;² and (21) follows by (16). Taking the square root, the second claim of the lemma, and thus the entire lemma, follow. \diamond

If we make no assumption on b , then (11) from Lemma 1 may provide a weak bound in terms of $|x_{opt}|_2$. If, on the other hand, we make the additional assumption that a constant fraction of the “weight” of b lies in the subspace spanned by the columns of A , then (11) can be strengthened. (Such an assumption is reasonable, since most least-squares problems are practically interesting if at least some part of b lies in the subspace spanned by the columns of A .)

Lemma 2 *Using the notation of Lemma 2, if in addition we assume that $|U_A U_A^T b|_2 \geq \gamma |b|_2$, for some fixed $\gamma \in (0, 1]$, then it follows that*

$$|x_{opt} - \tilde{x}_{opt}|_2 \leq \sqrt{\epsilon} \left(\kappa(A) \sqrt{\gamma^{-2} - 1} \right) |x_{opt}|_2. \quad (22)$$

²As $x_{opt} - \tilde{x}_{opt}$ lies in the row space of A , inequality (20) still holds if $\text{rank}(A) < d$.

Proof: Since $|U_A U_A^T b|_2 \geq \gamma |b|_2$, it follows that

$$\begin{aligned} \mathcal{Z}^2 &= |b|_2^2 - |U_A U_A^T b|_2^2 \\ &\leq (\gamma^{-2} - 1) |U_A U_A^T b|_2^2 \\ &\leq \sigma_{\max}^2(A) (\gamma^{-2} - 1) |x_{opt}|_2^2. \end{aligned}$$

This last inequality follows from

$$|x_{opt}|_2 = |V_A \Sigma_A^{-1} U_A^T b|_2 \geq \sigma_{\min}(\Sigma_A^{-1}) |U_A^T b|_2 = |U_A U_A^T b|_2 / \sigma_{\max}(A).$$

By combining this with (11) of Lemma 1, the lemma follows. \diamond

Thus, in order to prove Theorem 2, it suffices to show that the matrix \mathcal{SH} constructed by Algorithm 1 satisfies Conditions (8) and (9) with a probability of at least 3/4 and that we can perform the computations in the allotted time. Prior work has focused on constructing expensive nonuniform sampling probabilities [8, 9]; here we will see that preprocessing with the randomized Hadamard preprocessing gets around this difficulty.

3.3 The effect of the randomized Hadamard preprocessing

The randomized Hadamard transform $\mathcal{H} = HD$ was introduced recently as one step in the development of a “fast” version of the Johnson-Lindenstrauss lemma [1, 13]. Note that since \mathcal{H} is an orthogonal transformation, the solution x'_{opt} to the transformed problem

$$\mathcal{Z} = \min_{x \in \mathbb{R}^d} |\mathcal{H}(Ax - b)|_2,$$

is identical to the solution x_{opt} of the original problem (1). In this subsection, we state a lemma that quantifies the manner in which \mathcal{H} approximately “uniformizes” information in the left singular subspace of the matrix A when A is premultiplied by \mathcal{H} . In substance, this lemma is due to Ailon and Chazelle [1]. We state it for a general $n \times d$ orthogonal matrix U , although we will be interested in the case when $n > d$ and where U consists of the top d left singular vectors of the matrix A .

Lemma 3 *Let U be an $n \times d$ orthogonal matrix, let $N = \max\{n, d\}$, and let $\mathcal{H} = HD$ be the $n \times n$ randomized Hadamard transform defined above. Then, with probability at least 0.95,*

$$\left| (\mathcal{H}U)_{ij} \right| \leq C_0 \sqrt{\frac{\log N}{n}} \quad \text{for all } i \in [n], j \in [d], \text{ and thus} \quad (23)$$

$$\left| (\mathcal{H}U)_{(i)} \right|_2^2 \leq C_0^2 \frac{d \log N}{n} \quad \text{for all } i \in [n], \quad (24)$$

for some constant C_0 .

Proof: Ailon and Chazelle [1] prove that $\mathcal{H} = HD$ “spreads out” input vectors in the sense that $\max_{i \in [d]} |\mathcal{H}U^{(i)}|_\infty = s$ holds with probability at least $1 - 1/20$, where $s = O(n^{-1/2} \sqrt{\log(nd)})$. Since they assume that $d > n$, they prove that (23) holds if $s = O(n^{-1/2} \sqrt{\log(d)})$. By the same reasoning, under the assumption we have made, (23) holds if $s = O(n^{-1/2} \sqrt{\log N})$. To establish (24), fix $i \in [n]$. Summing the square of (23), for $1 \leq j \leq d$, establishes (24) and thus the lemma. \diamond

Remark: If U_A is a matrix containing the left singular vectors of A , then $U_{\mathcal{H}A} = \mathcal{H}U_A$ is the matrix containing the left singular vectors of $\mathcal{H}A$. By applying \mathcal{H} to A , we will ensure that $U_{\mathcal{H}A}$ (which we will *not* need to compute, but which we will use in our analysis) is “well-spread,” in the sense that $\max_{ij} |(U_{\mathcal{H}A})_{ij}|^2$ is close to, i.e., only an $O(\log n)$ factor away from, $1/n$, and thus that $\max_i |(U_{\mathcal{H}A})_{(i)}|_2^2$ is close to d/n . This approximate uniformity will manifest itself algorithmically. For our main random sampling algorithm (Algorithm 1 of Section 3.1), it will follow from (24) that uniform sampling probabilities will be approximately optimal, up to an $O(\log n)$ factor that will be absorbed into the sampling complexity. Similarly, for the sparse projection algorithm (see Algorithm 2 of Section 4.1 below), it will follow from (23) that the number of nonzero elements per row of the “sparse projection” matrix (to be defined in Section 4.2) will depend only on $\log n$.

3.4 Approximating matrix multiplication

In this subsection, we state a lemma on approximating the product of two matrices by random sampling that will be used in two different ways in the proof of Theorem 2. Given an $m \times n$ matrix A and an $n \times p$ matrix B , one can approximate the product AB by randomly sampling a small number r of columns of A and the corresponding rows of B according to carefully-constructed data-dependent nonuniform probability distributions. The following lemma is a basic quality-of-approximation result for approximating the product of two matrices and is described in more detail by Drineas, Mahoney, and Muthukrishnan [9]. The spectral norm bound holds if $B = A^T$; it was originally proven in the special case $\beta = 1$ by Rudelson and Vershynin [16].

Lemma 4 *Suppose $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{n \times p}$, and $r \leq n$. In addition, let $\{p_i\}_{i=1}^n$ be any probability distribution over $[n]$ such that*

$$p_i \geq \beta \frac{|A^{(i)}|_2^2}{\|A\|_F^2}, \quad (25)$$

for some $\beta \in (0, 1]$. Construct C and R by randomly sampling (and, if chosen, rescaling by $1/\sqrt{rp_i}$) each column of A and the corresponding row of B . Then,

$$\mathbf{E} [\|AB - CR\|_F] \leq \frac{1}{\sqrt{\beta r}} \|A\|_F \|B\|_F. \quad (26)$$

If, in addition, $B = A^T$, then

$$\mathbf{E} [\|AA^T - CC^T\|_2] \leq C_1 \sqrt{\frac{\log r}{\beta r}} \|A\|_F \|A\|_2, \quad (27)$$

for some constant C_1 .

Remark: In prior work [8, 9], nonuniform sampling probabilities were computed to satisfy (25) with $\beta = 1$. Here, we view β as a nonuniformity parameter and choose it such that (25) is satisfied with uniform sampling probabilities, i.e., $p_i = 1/n$, for all $i \in [n]$. In particular, (by Lemma 3; see also (29) in Section 3.5) after preprocessing with the randomized Hadamard transform \mathcal{H} , uniform sampling probabilities satisfy the conditions of Lemma 4 if we choose $\beta = 1/(C_0^2 \log n)$, for some constant C_0 . Thus, Algorithm 1 will be able to sample uniformly with only a small $O(\log(n) \log(d \log(n))/\log(d))$ increase in sample complexity.

3.5 Completing the proof of Theorem 2

In this subsection, we complete the proof of Theorem 2. Since Algorithm 1 is randomized, there will be a failure probability over the randomized choices made by the algorithm. First of all, we notice that the algorithm might abort with probability at most $1/10$. This follows since the expected number of non-zero diagonal elements in \mathcal{S}' is at most r , and thus an application of Markov's inequality yields an exit probability of at most $1/10$. In addition, we have already seen that the claims of Lemma 3 may fail with probability $1/20$. Finally, in the following, we will apply Markov's inequality two times—see (30) and (31) below—to remove an expectation. By appropriate choice of constants we can ensure that each event fails with probability no more than $1/20$. Thus, via the union bound, with a probability at least $3/4$, none of the bad events occur.

We start by establishing the following lemma, which states that all of the singular values of $\mathcal{S}\mathcal{H}U_A$ are close to 1. The proof of this lemma depends on the matrix perturbation theory bound (27) from Lemma 4. Keep in mind that Algorithm 1 selects $r = O(d \log(n) \log(d \log n)/\epsilon)$ rows. It follows immediately from this lemma that the smallest singular value of $\mathcal{S}\mathcal{H}U_A$ is at least $9/10$, and thus that Condition (8) is satisfied by Algorithm 1.

Lemma 5 *Subject to a failure probability: $|1 - \sigma_i^2(\mathcal{S}\mathcal{H}U_A)| < 1/10$, for all $i \in [d]$.*

Proof: Note that for all $i \in [d]$

$$\begin{aligned} |1 - \sigma_i^2(\mathcal{S}\mathcal{H}U_A)| &= |\sigma_i(U_A^T U_A) - \sigma_i(U_A^T \mathcal{H}^T \mathcal{S}^T \mathcal{S}\mathcal{H}U_A)| \\ &\leq \|U_A^T U_A - U_A^T \mathcal{H}^T \mathcal{S}^T \mathcal{S}\mathcal{H}U_A\|_2 \\ &= \|U_A^T \mathcal{H}^T \mathcal{H}U_A - U_A^T \mathcal{H}^T \mathcal{S}^T \mathcal{S}\mathcal{H}U_A\|_2, \end{aligned} \quad (28)$$

where (28) follows since $\mathcal{H}^T \mathcal{H} = I_n$. We can view $U_A^T \mathcal{H}^T \mathcal{S}^T \mathcal{S}\mathcal{H}U_A$ as approximating the product of two matrices $U_A^T \mathcal{H}^T = (\mathcal{H}U_A)^T$ and $\mathcal{H}U_A$ by random sampling the columns from $(\mathcal{H}U_A)^T$ and the rows from $\mathcal{H}U_A$. Although the sampling probabilities in Algorithm 1 are uniform and do not depend on the norms of the columns of $(\mathcal{H}U_A)^T$, it follows from Lemma 3 that

$$\frac{1}{n} \geq \beta^* \frac{|(\mathcal{H}U_A)_{(i)}|_2^2}{d} \quad \text{for all } i \in [n], \quad (29)$$

if we choose $\beta^* = 1/(C_0^2 \log n)$, for some constant C_0 . Note that from the unitarity of \mathcal{H} it follows that $\|\mathcal{H}U\|_F^2 = \|U\|_F^2 = d$. Thus, the conditions of the approximate matrix multiplication Lemma 4 are satisfied (with $\beta = \beta^*$), and it follows from (27) of Lemma 4 that

$$\mathbf{E} [\|U_A^T \mathcal{H}^T \mathcal{H}U_A - U_A^T \mathcal{H}^T \mathcal{S}^T \mathcal{S}\mathcal{H}U_A\|_2] \leq C_1 \sqrt{\frac{\log r}{\beta^* r}} \|\mathcal{H}U_A\|_F \|\mathcal{H}U_A\|_2 = C_1 \sqrt{\frac{d \log r}{\beta^* r}}, \quad (30)$$

for some constant C_1 . The lemma follows by applying Markov's inequality to (30), combining this with (28), and using the assumed choice of r . \diamond

We next prove the following lemma, from which it will follow that Condition (9) is satisfied by Algorithm 1. The proof of this lemma depends on the matrix perturbation bound (26) from Lemma 4. Recall that $b^\perp = U_A^\perp U_A^{\perp T} b$.

Lemma 6 *Subject to a failure probability: $\left|(\mathcal{S}\mathcal{H}U_A)^T \mathcal{S}\mathcal{H}b^\perp\right|_2^2 \leq \epsilon O(1)\mathcal{Z}^2$.*

Proof: First note that since $|U_A^T \mathcal{H}^T \mathcal{H} b^\perp|_2^2 = |U_A^T b^\perp|_2^2 = 0$ it follows that

$$\left| (\mathcal{S} \mathcal{H} U_A)^T \mathcal{S} \mathcal{H} b^\perp \right|_2^2 = \left| U_A^T \mathcal{H}^T \mathcal{S}^T \mathcal{S} \mathcal{H} b^\perp - U_A^T \mathcal{H}^T \mathcal{H} b^\perp \right|_2^2.$$

Thus, we can view $(\mathcal{S} \mathcal{H} U_A)^T \mathcal{S} \mathcal{H} b^\perp$ as approximating the product of two matrices $(\mathcal{H} U_A)^T$ and $\mathcal{H} b^\perp$ by random sampling columns from $(\mathcal{H} U_A)^T$ and rows/elements from $\mathcal{H} b^\perp$. As before, although the sampling probabilities are uniform and do not depend on, e.g, the norms of the columns of $(\mathcal{H} U_A)^T$ or the rows of $\mathcal{H} b^\perp$, it follows from Lemma 3 that (29) is satisfied if we choose $\beta^* = 1/(C_0^2 \log n)$, for some constant C_0 . Thus, the conditions of the approximate matrix multiplication Lemma 4 are satisfied (with $\beta = \beta^*$), and it follows from (26) of Lemma 4 that

$$\mathbf{E} \left[\left| (\mathcal{S} \mathcal{H} U_A)^T \mathcal{S} \mathcal{H} b^\perp \right|_2 \right] \leq \frac{1}{\sqrt{\beta^* r}} \|\mathcal{H} U_A\|_F \left\| \mathcal{H} b^\perp \right\|_F = \frac{1}{\sqrt{\beta^* r}} \sqrt{d} \mathcal{Z} = C_0 \sqrt{\frac{d \log n}{r}} \mathcal{Z}. \quad (31)$$

The lemma follows by applying Markov's inequality, using the assumed choice of r , and squaring both sides. ◊

Thus, by appropriate choice of constants, both Condition (8) and Condition (9) are satisfied with $\mathcal{X} = \mathcal{S} \mathcal{H}$ with probability at least 3/4. In order to complete the proof of Theorem 2, we discuss the running time of the algorithm. In Step 7 we need to compute the matrix $\mathcal{S} H D A$ and the vector $\mathcal{S} H D b$. We only describe the former; the latter is essentially the same. Clearly, $D A$ can be computed in $O(nd)$ time. In order to compute $\mathcal{S} H D A$ we will take advantage of the special structure on the matrices \mathcal{S} and H , and we will only compute those elements of $H D A$ that we need to access via the sampling procedure \mathcal{S} . In particular, we will follow the lines of Theorem 2.3 of Ailon and Liberty [2], in which it is shown that a straightforward modification of the Walsh-Hadamard algorithm can compute $\mathcal{S} H D A$ in $O(nd \log r')$ time, where $r' \leq 10r$ is the number of rows in \mathcal{S} . Then, in Step 7 we need $O(r' d^2)$ time to compute the solution. Thus, the entire algorithm runs in $O(nd \log r + r d^2)$ time, where $r = O(d \log(n) \log(d \log n)/\epsilon)$. This completes the proof of Theorem 2.

4 A projection-based randomized algorithm

In this section, we present a second randomized algorithm to provide a very accurate relative-error approximation to the solution of the least squares approximation problem (1) in $o(nd^2)$ time. We also state and prove an associated quality-of-approximation theorem.

4.1 The main algorithm and main theorem

Algorithm 2 takes as input an $n \times d$ matrix A , an n -vector b , and an error parameter $\epsilon \in (0, 1/2)$. This algorithm also starts by preprocessing the matrix A and right hand side vector b with a randomized Hadamard transform. It then constructs a smaller problem by performing a “sparse projection” on the preprocessed problem. Our main quality-of-approximation theorem—see Theorem 3 below—will state that with constant probability (over the random decisions made by the algorithm) the vector \tilde{x}_{opt} returned by this algorithm will satisfy relative-error bounds of the form (4) and (5) and will be computed in $o(nd^2)$ time.

In more detail, Algorithm 2 begins by preprocessing the matrix A and right hand side vector b with a randomized Hadamard transform \mathcal{H} , as described in Section 2. (As before, this algorithm explicitly computes only those rows of $\mathcal{H}[Ab]$ that need to be accessed to perform the sparse projection.) After this initial preprocessing, Algorithm 2 will perform a random “sparse

Input: $A \in \mathbb{R}^{n \times d}$, $b \in \mathbb{R}^n$, and an error parameter $\epsilon \in (0, 1/2)$.

Output: $\tilde{x}_{opt} \in \mathbb{R}^d$.

1. Let $k = O(d/\epsilon)$ and $q = O(d \log^2(n)/n + d^2 \log(n)/n)$.
2. Let $\mathcal{T}' \in \mathbb{R}^{k \times n}$ be a random matrix with

$$\mathcal{T}'_{ij} = \begin{cases} +\sqrt{\frac{1}{kq}} & , \text{ with probability } q/2 \\ -\sqrt{\frac{1}{kq}} & , \text{ with probability } q/2 \\ 0 & , \text{ with probability } 1 - q, \end{cases}$$

for all i, j independently.

3. If the number of non-zero entries in \mathcal{T} is more than $10kqn$, then abort.
4. Let $H \in \mathbb{R}^{n \times n}$ be the normalized Hadamard transform matrix.
5. Let $D \in \mathbb{R}^{n \times n}$ be a diagonal matrix with

$$D_{ii} = \begin{cases} +1 & , \text{ with probability } 1/2 \\ -1 & , \text{ with probability } 1/2 \end{cases}$$

6. Let $\mathcal{H} = HD$. Compute and return $\tilde{x}_{opt} = (\mathcal{T}\mathcal{H}A)^+ \mathcal{T}\mathcal{H}b$.

Algorithm 2: A fast random projection algorithm for least squares approximation

projection” by multiplying the result of the first step by the matrix \mathcal{T} (described in more detail in Section 4.2). Then, we can consider the problem

$$\tilde{\mathcal{Z}} = \min_{x \in \mathbb{R}^d} |\mathcal{T}\mathcal{H}Ax - \mathcal{T}\mathcal{H}b|_2,$$

which is just a least squares approximation problem involving the k coordinates projected from a Hadamard-preprocessed version of (1). The minimum-length vector $\tilde{x}_{opt} \in \mathbb{R}^d$ among those that achieve the minimum value $\tilde{\mathcal{Z}}$ in this problem is

$$\tilde{x}_{opt} = (\mathcal{T}\mathcal{H}A)^+ \mathcal{T}\mathcal{H}b.$$

As before, since $k \ll n$, we will compute \tilde{x}_{opt} exactly. Our main quality-of-approximation results for Algorithm 2 are presented in the following theorem; the remainder of this section will be devoted to its proof.

Theorem 3 *Suppose $A \in \mathbb{R}^{n \times d}$, $b \in \mathbb{R}^n$, and let $\epsilon \in (0, 1/2)$. Run Algorithm 2, and return \tilde{x}_{opt} as an approximation to the solution of the least squares problem (1). Then, with probability at least $3/4$, the following three claims hold: first, \tilde{x}_{opt} satisfies*

$$|A\tilde{x}_{opt} - b|_2 \leq (1 + \epsilon)\mathcal{Z};$$

second, if we assume that $|U_A U_A^T b|_2 \geq \gamma |b|_2$ for some $\gamma \in (0, 1]$ then \tilde{x}_{opt} satisfies

$$|x_{opt} - \tilde{x}_{opt}|_2 \leq \sqrt{\epsilon} \left(\kappa(A) \sqrt{\gamma^{-2} - 1} \right) |x_{opt}|_2;$$

and third,

$$O(nd \log(d \log(n)/\epsilon) + d^4 \log^2(n)/\epsilon)$$

time suffices to compute the solution \tilde{x}_{opt} .

Remark: Assuming that ϵ is a constant independent of n and d , Theorem 2 implies $o(nd^2)$ running times for a wide class of over-constrained least-squares problems. In particular, if $\log n \leq d$ and $n/\log^2 n = \omega(d^2)$, then the running time of the above algorithm is $o(nd^2)$. In words, as long as n is sufficiently larger than d and does not exceed 2^d , then the above algorithm improves upon the running time of standard least-squares algorithms.

Remark: Consider, for example, the special case where $\log n \leq d$ and $n/\log^2 n = \Omega(d^3)$. Then, the running time of the above algorithm becomes $O(nd \log d)$.

4.2 Sparse projection matrices

In this subsection, we state a lemma about the action of a sparse random matrix operating on a vector. Recall that given any set of n points in Euclidean space, the Johnson-Lindenstrauss lemma states that those points can be mapped via a linear function to $k = O(\epsilon^{-2} \log n)$ dimensions such that the distances between all pairs of points are preserved to within a multiplicative factor of $1 \pm \epsilon$; see [13] and references therein for details. Recently, Ailon and Chazelle proved that this result can be obtained by composing a sparse random matrix with the randomized Hadamard transform \mathcal{H} of Section 3.3 [1].

More formally, given $n \in \mathbb{Z}^+$ (which will be the number of columns in our random matrix), let $\epsilon \in (0, 1/2)$ be an error parameter, $\delta \in (0, 1)$ be a failure probability, and $\alpha \in [1/\sqrt{n}, 1]$ be a “uniformity” parameter. In addition, let q be a “sparsity” parameter defining the expected number of nonzero elements per row, and let k be number of rows in our matrix. Then, define the $k \times n$ random matrix $\mathcal{T} = \mathcal{T}(k, q)$ as in Algorithm 2. Matoušek proved the following lemma, as the key step in his version of the Ailon-Chazelle result [1, 13].

Lemma 7 *Let $\mathcal{T} = \mathcal{T}(k, q)$ be the sparse random matrix defined above, where $q = C_q \alpha^2 \log(\frac{n}{\epsilon \delta})$ for some sufficiently large constant C_q (but still such that $q \leq 1$), and $k = C_k \epsilon^{-2} \log(\frac{4}{\delta})$ for some sufficiently large constant C_k (but such that k is integral). Then for every vector $x \in \mathbb{R}^n$ such that $|x|_\infty / |x|_2 \leq \alpha$, we have that with probability at least $1 - \delta$*

$$(1 - \epsilon) |x|_2 \leq |\mathcal{T}x|_2 \leq (1 + \epsilon) |x|_2.$$

Remark: In order to achieve sufficient concentration for all vectors $x \in \mathbb{R}^n$, the linear mapping defining the Johnson-Lindenstrauss transform is typically “dense,” in the sense that $\Omega(n)$ of the elements in each of the k rows of the matrix defining the mapping are nonzero. In this case, implementing the mapping on d vectors (in, e.g., a matrix A) via a matrix multiplication requires $O(ndk)$ time, which is no faster than the $O(nd^2)$ time required to compute an exact solution to (1) if $k = \Omega(d)$. The Ailon-Chazelle result [1, 13] states that the mapping can be “sparse,” in the sense that only $o(n)$ of the elements in each of the k rows need to be nonzero, provided that the vector x is “well-spread,” in the sense that $|x|_\infty / |x|_2$ is close to $1/\sqrt{n}$, which by Lemma 3 can be achieved by preprocessing with \mathcal{H} . Thus, in this case, our second algorithm for approximating least squares approximation (Algorithm 2 of Section 4.1) will be able to perform the matrix multiplication to implement the sparse projection in $o(nd^2)$ time.

4.3 Proof of Theorem 3

In this subsection, we provide a proof of Theorem 3. Recall that by the results of Section 3.2, in order to prove Theorem 3, we must show that the matrix \mathcal{TH} constructed by Algorithm 2 satisfies Conditions (8) and (9) with a probability of at least $3/4$. Since Algorithm 2 is randomized, there will be a failure probability over the randomized choices made by the algorithm. First of all, the algorithm might abort with probability at most $1/10$. To see this, notice that the expected number of non-zero entries in the matrix \mathcal{T} , denoted by $|\mathcal{T}|$, is

$$\mathbf{E}[|\mathcal{T}|] = knq = O(d^2 \log^2(n)/\epsilon + d^3 \log(n)/\epsilon).$$

Thus Markov's inequality gives an exit probability of at most $1/10$. In addition, we have already seen that the claims of Lemma 3 may fail with probability $1/20$. Finally, in the following, we will have two events that may fail with some probability—see Lemmas 9 and 11 below. By appropriate choice of constants we can ensure that each event fails with probability no more than $1/20$. Thus, via the union bound, with a probability at least $3/4$, none of the bad events occur.

In order to prove that all of the singular values of $\mathcal{TH}U_A$ are close to 1, we will use the following lemma, which provides a means to bound the spectral norm of a matrix. We will apply it with the parameters ϵ' set as $\epsilon' = 1/120$ and ϵ_0 set as $\epsilon_0 = 1/2$. The lemma is due to Arora, Hazan, and Kale [3], although it is a modification of a lemma due to Feige and Ofek [10].

Lemma 8 *Let M be a $d \times d$ symmetric matrix, and define the grid*

$$\Omega = \left\{ x : x \in \frac{\epsilon_0}{\sqrt{d}} \mathbb{Z}^d, |x|_2 \leq 1 \right\}. \quad (32)$$

Then, the cardinality of Ω is at most $\exp(cd)$, where $c = (2 + 1/\epsilon_0)$. In addition, if for every $x, y \in \Omega$, we have that $|x^T M y| \leq \epsilon'$, then for every unit vector x , we have that $|x^T M x| \leq \frac{\epsilon'}{(1-\epsilon_0)^2}$.

We next establish the following lemma, which states that all of the singular values of $\mathcal{TH}U_A$ are close to 1. The proof of this lemma depends on the bound provided by Lemma 8. It follows immediately from this lemma that the smallest singular value of $\mathcal{TH}U_A$ is at least $9/10$, and thus that Condition (8) is satisfied by Algorithm 2.

Lemma 9 *Subject to a failure probability: $|1 - \sigma_i^2(\mathcal{TH}U_A)| < 1/10$, for all $i \in [d]$.*

Proof: Define the $d \times d$ symmetric matrix M as $M = U_A^T \mathcal{H}^T \mathcal{T}^T \mathcal{TH}U_A - I_d$, recall that $I_d = U_A^T \mathcal{H}^T \mathcal{H}U_A$, and note that

$$|1 - \sigma_i^2(\mathcal{TH}U_A)| \leq \|M\|_2, \quad (33)$$

for all $i \in [d]$. Fix $\epsilon_0 = 1/2$, and define the grid Ω as (32). Note that there are no more than e^{8d} pairs $(x, y) \in \Omega \times \Omega$, since $|\Omega| \leq e^{4d}$ by Lemma 8. Since $\|M\|_2 = \sup_{|x|_2=1} |x^T M x|$, in order to show that $\|M\|_2 \leq 1/10$, it suffices by Lemma 8 to show that $|x^T M y| \leq 1/40$, for all $x, y \in \Omega$. To do so, first, consider a single x, y pair. Let

$$\begin{aligned} \Delta_1 &= |\mathcal{TH}U_A(x+y)|_2^2 - |\mathcal{H}U_A(x+y)|_2^2 \\ \Delta_2 &= |\mathcal{TH}U_A x|_2^2 - |\mathcal{H}U_A x|_2^2 \\ \Delta_3 &= |\mathcal{TH}U_A y|_2^2 - |\mathcal{H}U_A y|_2^2, \end{aligned}$$

and note that

$$\Delta_1 = (x+y)^T U_A^T \mathcal{H}^T \mathcal{T}^T \mathcal{TH}U_A (x+y) - (x+y)^T I_d (x+y).$$

By multiplying out the right hand side of this and rearranging terms, it follows that

$$x^T U_A^T \mathcal{H}^T \mathcal{T}^T \mathcal{T} \mathcal{H} U_A y - x^T I_d y = \frac{1}{2} (\Delta_1 + \Delta_2 + \Delta_3).$$

In order to use Lemma 7 to bound the quantities Δ_1, Δ_2 , and Δ_3 , we need a bound on the ‘uniformity’ parameter $\alpha = |\mathcal{H}U_A x|_\infty / |\mathcal{H}U_A x|_2$. To do so, note that

$$\alpha^* \equiv \frac{|\mathcal{H}U_A x|_\infty}{|\mathcal{H}U_A x|_2} = \frac{\max_i |(\mathcal{H}U_A)_{(i)} x|}{|\mathcal{H}U_A x|_2} \leq \max_i |(\mathcal{H}U_A)_{(i)}|_2 \leq C_0 \sqrt{\frac{d \log n}{n}},$$

for some constant C_0 , where the first inequality follows since $|\mathcal{H}U_A x|_2 = |x|_2$, and where the second inequality follows by (24) of Lemma 3. This holds for both our chosen points x and y (and in fact for all $x \in \Omega$). Next, note that if $|\mathcal{H}U_A x|_\infty \leq \alpha'/2$ and $|\mathcal{H}U_A y|_\infty \leq \alpha'/2$, for any α' , then by the triangle inequality $|\mathcal{H}U_A(x+y)|_\infty \leq \alpha'$, for the same α' . Thus, it follows from Lemma 7 that each of the following three statements holds with probability at least $1 - \delta$:

$$\begin{aligned} |\Delta_1| &\leq \epsilon' |U_A(x+y)|_2^2 \leq 4\epsilon' \\ |\Delta_2| &\leq \epsilon' |U_A x|_2^2 \leq \epsilon' \\ |\Delta_3| &\leq \epsilon' |U_A y|_2^2 \leq \epsilon'. \end{aligned}$$

Thus, for this single x, y pair, with probability no less than $1 - 3\delta$,

$$|x^T M y| = |x^T U_A^T \mathcal{H}^T \mathcal{T}^T \mathcal{T} \mathcal{H} U_A y - x^T y| \leq \frac{1}{2} 6\epsilon' = 3\epsilon'. \quad (34)$$

Next, recall that there are no more than e^{8d} pairs of vectors $(x, y) \in \Omega \times \Omega$, and we need (34) to hold for all of them. If we choose δ as $\delta = 1/(60e^{8d})$, then it follows by a union bound that (34) holds for all $(x, y) \in \Omega \times \Omega$, with probability at least $1 - 1/20$. Additionally, let us set ϵ' as $\epsilon' = 1/120$. From Lemma 7, this choice of δ and ϵ' implies that we should choose

$$q = O\left(\alpha^{*2} \log(ne^d/\epsilon')\right) = O\left(\alpha^{*2} \log n + \alpha^{*2} d\right) = O\left(C_0^2 d \log^2(n)/n + C_0^2 d^2 \log(n)/n\right)$$

and that $k = O(d/\epsilon)$. (Note that $k = O(d)$ actually suffices for this lemma, but we will need $k = O(d/\epsilon)$ below.) The lemma follows since these are the values for k and q chosen by Algorithm 2. \diamond

In order to prove that Condition (9) is satisfied, we start with the following lemma, which states that, given vectors x and y , we can use the random ‘sparse projection’ matrix \mathcal{T} to approximate $|x^T y|$ by $|x^T \mathcal{T}^T \mathcal{T} y|$, provided that $|x|_\infty$ (or $|y|_\infty$, but not necessarily both) is bounded above by the parameter α used to construct \mathcal{T} . The proof of this technical lemma is elementary, but tedious, and so is deferred to Section 4.4.

Lemma 10 *Let x, y be vectors in \mathbb{R}^n such that $|x|_\infty \leq \alpha$. Let $\mathcal{T} = \mathcal{T}(k, q)$ be the $k \times n$ sparse random matrix, as constructed in Section 4.2, with sparsity parameter q . If $q \geq \alpha^2$, then*

$$\mathbf{E} \left[|x^T \mathcal{T}^T \mathcal{T} y - x^T y|^2 \right] \leq \frac{3}{k} |x|_2^2 |y|_2^2 + \frac{3}{k} |y|_2^2, \text{ if } |x|_\infty \leq \alpha.$$

Proof: The proof is deferred to Section 4.4. \diamond

We next prove the following lemma, from which it will follow that Condition (9) is satisfied by Algorithm 2. The proof of this lemma depends on the bound provided by Lemma 10. Recall that $b^\perp = U_A^\perp U_A^{\perp T} b$.

Lemma 11 *Subject to a failure probability:* $\left| (\mathcal{T}\mathcal{H}U_A)^T \mathcal{T}\mathcal{H}b^\perp \right|_2^2 = \epsilon O(1) \mathcal{Z}^2$.

Proof: We first note that since $U_A^T b^\perp = 0$, it follows that $U_A^{(i)T} b^\perp = U_A^{(i)T} \mathcal{H}^T \mathcal{H} b^\perp = 0$, for all $i \in [d]$. Thus, we have that

$$\left| U_A^T \mathcal{H}^T \mathcal{T}^T \mathcal{T} \mathcal{H} b^\perp \right|_2^2 = \sum_{i=1}^d \left(\left((\mathcal{H}U_A)^{(i)} \right)^T \mathcal{T}^T \mathcal{T} \mathcal{H} b^\perp - U_A^{(i)T} \mathcal{H}^T \mathcal{H} b^\perp \right)^2. \quad (35)$$

We will first bound the expectation of the left hand side of (35) by using Lemma 10 to bound each term on the right hand side of (35). By (23) of Lemma 3,

$$\left| (\mathcal{H}U_A)^{(i)} \right|_\infty \leq C_0 \sqrt{\log(n)/n} \equiv \alpha^*,$$

for some constant C_0 , for all $i \in [d]$. By the choice of the ‘‘sparsity’’ parameter q in Algorithm 2, $q > \alpha^{*2}$, and thus the conditions of Lemma 10 are satisfied. Thus, it follows from Lemma 10 that

$$\begin{aligned} \mathbf{E} \left[\left| U_A^T \mathcal{H}^T \mathcal{T}^T \mathcal{T} \mathcal{H} b^\perp \right|_2^2 \right] &= \sum_{i=1}^d \mathbf{E} \left[\left(\left((\mathcal{H}U_A)^{(i)} \right)^T \mathcal{T}^T \mathcal{T} \mathcal{H} b^\perp - U_A^{(i)T} \mathcal{H}^T \mathcal{H} b^\perp \right)^2 \right] \\ &\leq \sum_{i=1}^d \frac{3}{k} \left| (\mathcal{H}U_A)^{(i)} \right|_2^2 \left| \mathcal{H} b^\perp \right|_2^2 + \frac{3}{k} \left| \mathcal{H} b^\perp \right|_2^2 \\ &= \frac{6d}{k} \left| \mathcal{H} b^\perp \right|_2^2, \end{aligned}$$

where the last line follows since $\left| (\mathcal{H}U_A)^{(i)} \right|_2 = 1$, for all $i \in [d]$. Thus, by Markov’s inequality, with probability at least 0.95, $\left| U_A^T \mathcal{H}^T \mathcal{T}^T \mathcal{T} \mathcal{H} b^\perp \right|_2^2 \leq \frac{120d}{k} \mathcal{Z}^2$. The lemma follows since $k = O(d/\epsilon)$. \diamond

Thus, by appropriate choice of constants, both Condition (8) and Condition (9) are satisfied with $\mathcal{X} = \mathcal{T}\mathcal{H}$ with probability at least 3/4. In order to complete the proof of Theorem 3, we discuss the running time of the algorithm. In Step 6 we need to compute the matrix $\mathcal{T}HDA$ and the vector $\mathcal{T}HDb$. As before, we only describe the former; the latter is essentially the same. First, note that in order to perform the multiplication by \mathcal{T} of the $n \times d$ matrix $\mathcal{H}A$, it clearly suffices to spend $O(d|\mathcal{T}|)$ time, where $|\mathcal{T}|$ denotes the number of nonzero elements in the $k \times n$ matrix \mathcal{T} . Then, using the results from Theorem 2.3 of Ailon and Liberty [2], only $O(nd \log |\mathcal{T}|)$ time is needed in order to perform the preprocessing by \mathcal{H} for those $|\mathcal{T}|$ coordinates. After preprocessing with the randomized Hadamard transform \mathcal{H} and the multiplication by the random matrix \mathcal{T} , Algorithm 2 has $k = O(d/\epsilon)$ constraints and solves a least squares problem on those constraints; this requires $O(kd^2)$ time. Thus, the entire algorithm runs in $O(nd \log |\mathcal{T}| + d|\mathcal{T}| + kd^2)$ time. Since $|\mathcal{T}| \leq 10kqn$ in Algorithm 2, this is

$$O \left(nd \log \left(\frac{d \log n}{\epsilon} \right) + nd \log (d + \log n) + \frac{d^3 \log^2 n}{\epsilon} + \frac{d^4 \log n}{\epsilon} + \frac{d^3}{\epsilon} \right)$$

time. This completes the proof of Theorem 3.

4.4 Proof of Lemma 10

In this subsection, we will provide a proof of Lemma 10.

Let $\mathcal{T} = \mathcal{T}(k, q)$ be the $k \times n$ sparse random matrix, as constructed in Section 4.2, with sparsity parameter q . In addition, given $x, y \in \mathbb{R}^n$, let $\Delta = x^T \mathcal{T}^T \mathcal{T} y - x^T y$. We will provide a bound for $\mathbf{E}[\Delta^2] = \mathbf{E}[(x^T \mathcal{T}^T \mathcal{T} y - x^T y)^2]$. Let $t_{(i)}$ be the i^{th} row of \mathcal{T} , for $i \in [k]$, in which case

$$\Delta = \sum_{i=1}^k \left(x^T t_{(i)}^T t_{(i)} y - \frac{1}{k} x^T y \right).$$

Rather than computing $\mathbf{E}[\Delta^2]$ directly, we will instead use that $\mathbf{E}[\Delta^2] = (\mathbf{E}[\Delta])^2 + \mathbf{Var}[\Delta]$.

We first claim that $\mathbf{E}[\Delta] = 0$. By the linearity of expectation,

$$\mathbf{E}[\Delta] = \sum_{i=1}^k \left[\mathbf{E} \left[x^T t_{(i)}^T t_{(i)} y \right] - \frac{1}{k} x^T y \right]. \quad (36)$$

In order to evaluate (36), let's first analyze $t_{(i)} = t$ for some i , say $i = 1$. Recall: that $\mathbf{E}[t_i] = 0$; and that $\mathbf{E}[t_i t_j] = 0$ for $i \neq j$; and also that $\mathbf{E}[t_i^2] = 1/k$. Thus,

$$\mathbf{E}[x^T t^T t y] = \mathbf{E} \left[\sum_{i=1}^n \sum_{j=1}^n x_i t_i t_j y_j \right] = \sum_{i=1}^n \sum_{j=1}^n x_i \mathbf{E}[t_i t_j] y_j = \sum_{i=1}^n x_i \mathbf{E}[t_i^2] y_i = \frac{1}{k} x^T y.$$

By combining this with (36), it follows that $\mathbf{E}[\Delta] = 0$, and thus that $\mathbf{E}[\Delta^2] = \mathbf{Var}[\Delta]$.

In order to provide a bound for $\mathbf{Var}[\Delta]$, note that

$$\mathbf{Var}[\Delta] = \sum_{i=1}^k \mathbf{Var} \left[x^T t_{(i)}^T t_{(i)} y - \frac{1}{k} x^T y \right] \quad (37)$$

$$= \sum_{i=1}^k \mathbf{Var} \left[x^T t_{(i)}^T t_{(i)} y \right], \quad (38)$$

where (37) follows since the k random variables $x^T t_{(i)}^T t_{(i)} y - \frac{1}{k} x^T y$ are independent since the elements of \mathcal{T} are independent, and where (38) follows since $\frac{1}{k} x^T y$ is constant. In order to evaluate (38), let's again first analyze $t_{(i)} = t$ for some i , say $i = 1$. Then,

$$\begin{aligned} \mathbf{Var}[x^T t^T t y] &= \mathbf{E}[(x^T t^T t y)^2] - (\mathbf{E}[x^T t^T t y])^2 \\ &= \mathbf{E}[(x^T t^T t y)^2] - \frac{1}{k^2} (x^T y)^2. \end{aligned} \quad (39)$$

We will bound the $\mathbf{E}[(x^T t^T t y)^2]$ term directly:

$$\begin{aligned} \mathbf{E} \left[\left(\sum_{i=1}^n \sum_{j=1}^n x_i t_i t_j y_j \right)^2 \right] &= \mathbf{E} \left[\sum_{i_1=1}^n \sum_{i_2=1}^n \sum_{j_1=1}^n \sum_{j_2=1}^n x_{i_1} x_{i_2} t_{i_1} t_{i_2} t_{j_1} t_{j_2} y_{j_1} y_{j_2} \right] \\ &= \sum_{i_1=1}^n \sum_{i_2=1}^n \sum_{j_1=1}^n \sum_{j_2=1}^n x_{i_1} x_{i_2} \mathbf{E}[t_{i_1} t_{i_2} t_{j_1} t_{j_2}] y_{j_1} y_{j_2}. \end{aligned} \quad (40)$$

Notice that if any of the four indices i_1, i_2, j_1, j_2 appears only once, then the expectation $\mathbf{E}[t_{i_1} t_{i_2} t_{j_1} t_{j_2}]$ corresponding to those indices equals zero. This expectation is non-zero if the 4 indices are paired

in couples or if all four are equal. That is, non-zero expectation happens if

$$\begin{aligned}
\text{(A)} & : i_1 = i_2 \neq j_1 = j_2 && (n^2 - n \text{ terms}) \\
\text{(B)} & : i_1 = j_1 \neq i_2 = j_2 && (n^2 - n \text{ terms}) \\
\text{(C)} & : i_1 = j_2 \neq i_2 = j_1 && (n^2 - n \text{ terms}) \\
\text{(D)} & : i_1 = i_2 = j_1 = j_2 && (n \text{ terms}).
\end{aligned}$$

For Case (A), let $i_1 = i_2 = \ell$ and let $j_1 = j_2 = p$, in which case the corresponding terms in equation (40) become:

$$\begin{aligned}
\sum_{\ell=1}^n \sum_{p=1:p \neq \ell}^n x_\ell^2 \mathbf{E} [t_\ell^2 t_p^2] y_p^2 &= \sum_{\ell=1}^n \sum_{p=1:p \neq \ell}^n x_\ell^2 \mathbf{E} [t_\ell^2] \mathbf{E} [t_p^2] y_p^2 \\
&= \frac{1}{k^2} \sum_{\ell=1}^n \sum_{p=1:p \neq \ell}^n x_\ell^2 y_p^2 \\
&= \frac{1}{k^2} \sum_{\ell=1}^n \sum_{p=1:p \neq \ell}^n x_\ell^2 y_p^2 + \frac{1}{k^2} \sum_{p=1}^n x_p^2 y_p^2 - \frac{1}{k^2} \sum_{p=1}^n x_p^2 y_p^2 \\
&= \frac{1}{k^2} |x|_2^2 |y|_2^2 - \frac{1}{k^2} \sum_{p=1}^n x_p^2 y_p^2.
\end{aligned}$$

Similarly, Cases (B) and (C) give:

$$\begin{aligned}
\sum_{\ell=1}^n \sum_{p=1:p \neq \ell}^n x_\ell x_p \mathbf{E} [t_\ell^2 t_p^2] y_\ell y_p &= \frac{1}{k^2} (x^T y)^2 - \frac{1}{k^2} \sum_{p=1}^n x_p^2 y_p^2 \\
&\quad (\text{where } i_1 = j_1 = \ell \text{ and } i_2 = j_2 = p), \text{ and} \\
\sum_{\ell=1}^n \sum_{p=1:p \neq \ell}^n x_\ell x_p \mathbf{E} [t_\ell^2 t_p^2] y_\ell y_p &= \frac{1}{k^2} (x^T y)^2 - \frac{1}{k^2} \sum_{p=1}^n x_p^2 y_p^2 \\
&\quad (\text{where } i_1 = j_2 = \ell \text{ and } i_2 = j_1 = p).
\end{aligned}$$

Finally, for Case (D), let $i_1 = i_2 = j_1 = j_2 = \ell$, in which case:

$$\sum_{\ell=1}^n x_\ell^2 \mathbf{E} [t_\ell^4] y_\ell^2 = \frac{1}{k^2 q} \sum_{\ell=1}^n x_\ell^2 y_\ell^2,$$

where we have used that $\mathbf{E} [t_\ell^4] = 1/(k^2 q)$. By combining these four terms for each of the k terms in the sum, it follows from (38) and (39) that

$$\begin{aligned}
\mathbf{E} [\Delta^2] &= k \left(\frac{1}{k^2} |x|_2^2 |y|_2^2 + \frac{2}{k^2} (x^T y)^2 - \frac{3}{k^2} \sum_{p=1}^n x_p^2 y_p^2 + \frac{1}{k^2 q} \sum_{p=1}^n x_p^2 y_p^2 - \frac{1}{k^2} (x^T y)^2 \right) \\
&\leq \frac{2}{k} |x|_2^2 |y|_2^2 + \frac{1}{kq} \sum_{p=1}^n x_p^2 y_p^2,
\end{aligned} \tag{41}$$

where we have used that $(x^T y)^2 \leq |x|_2^2 |y|_2^2$. Under the assumption that $|x|_\infty \leq \alpha$, the second term on the right hand side of (41) is bounded above by $\frac{\alpha^2}{kq} |y|_2^2$, and so the lemma follows since we have assumed that $q \geq \alpha^2$.

References

- [1] N. Ailon and B. Chazelle. Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 557–563, 2006.
- [2] N. Ailon and E. Liberty. Fast dimension reduction using Rademacher series on dual BCH codes. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1–9, 2008.
- [3] S. Arora, E. Hazan, and S. Kale. A fast random sampling algorithm for sparsifying matrices. In *Proceedings of the 10th International Workshop on Randomization and Computation*, pages 272–279, 2006.
- [4] A. Ben-Israel and T.N.E. Greville. *Generalized Inverses: Theory and Applications*. Springer-Verlag, New York, 2003.
- [5] R. Bhatia. *Matrix Analysis*. Springer-Verlag, New York, 1997.
- [6] K. L. Clarkson and D. P. Woodruff. Numerical linear algebra in the streaming model. *Manuscript. To appear in STOC 2009*.
- [7] G. Dahlquist, B. Sjöberg, and P. Svensson. Comparison of the method of averages with the method of least squares. *Mathematics of Computation*, 22(104):833–845, 1968.
- [8] 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*, pages 1127–1136, 2006.
- [9] P. Drineas, M.W. Mahoney, and S. Muthukrishnan. Relative-error CUR matrix decompositions. *SIAM Journal on Matrix Analysis and Applications*, 30:844–881, 2008.
- [10] U. Feige and E. Ofek. Spectral techniques applied to sparse random graphs. *Random Structures and Algorithms*, 27(2):251–275, 2005.
- [11] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, 1996.
- [12] O.H. Ibarra, S. Moran, and R. Hui. A generalization of the fast LUP matrix decomposition algorithm and applications. *Journal of Algorithms*, 3:45–56, 1982.
- [13] J. Matoušek. On variants of the Johnson–Lindenstrauss lemma. *Manuscript*.
- [14] N. H. Nguyen, T. T. Do, and T. D. Tran. A fast and efficient algorithm for low-rank approximation of a matrix. *Manuscript. To appear in STOC 2009*.
- [15] V. Rokhlin and M. Tygert. A fast randomized algorithm for overdetermined linear least-squares regression. *Proc. Natl. Acad. Sci. USA*, 105(36):13212–13217, 2008.
- [16] M. Rudelson and R. Vershynin. Sampling from large matrices: an approach through geometric functional analysis. *Journal of the ACM*, 54(4):Article 21, 2007.
- [17] T. Sarlós. Improved approximation algorithms for large matrices via random projections. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 143–152, 2006.

- [18] G.W. Stewart and J.G. Sun. *Matrix Perturbation Theory*. Academic Press, New York, 1990.
- [19] S.M. Stigler. *The History of Statistics: The Measurement of Uncertainty before 1900*. Harvard University Press, Cambridge, 1986.