FINDING STRUCTURE WITH RANDOMNESS: STOCHASTIC ALGORITHMS FOR CONSTRUCTING APPROXIMATE MATRIX DECOMPOSITIONS

N. HALKO, P. G. MARTINSSON, AND J. A. TROPP

Technical Report No. 2009-05
September 2009

APPLIED & COMPUTATIONAL MATHEMATICS
CALIFORNIA INSTITUTE OF TECHNOLOGY
mail code 217-50 · pasadena, ca 91125

FINDING STRUCTURE WITH RANDOMNESS: STOCHASTIC ALGORITHMS FOR CONSTRUCTING APPROXIMATE MATRIX DECOMPOSITIONS

N. HALKO[†], P. G. MARTINSSON[†], AND J. A. TROPP[‡]

Abstract. Low-rank matrix approximations, such as the truncated singular value decomposition and the rank-revealing QR decomposition, play a central role in data analysis and scientific computing. This work surveys recent research which demonstrates that *randomization* offers a powerful tool for performing low-rank matrix approximation. These techniques exploit modern computational architectures more fully than classical methods and open the possibility of dealing with truly massive data sets. In particular, these techniques offer a route toward principal component analysis (PCA) for petascale data.

This paper presents a modular framework for constructing randomized algorithms that compute partial matrix decompositions. These methods use random sampling to identify a subspace that captures most of the action of a matrix. The input matrix is then compressed—either explicitly or implicitly—to this subspace, and the reduced matrix is manipulated deterministically to obtain the desired low-rank factorization. In many cases, this approach beats its classical competitors in terms of accuracy, speed, and robustness. These claims are supported by extensive numerical experiments and a detailed error analysis.

The specific benefits of randomized techniques depend on the computational environment. Consider the model problem of finding the k dominant components of the singular value decomposition of an $m \times n$ matrix. (i) For a dense input matrix, randomized algorithms require $O(mn \log(k))$ floating-point operations (flops) in contrast with $O(mnk)$ for classical algorithms. (ii) For a sparse input matrix, the flop count matches classical Krylov subspace methods, but the randomized approach is more robust and can be reorganized to exploit multi-processor architectures. (iii) For a matrix that is too large to fit in slow memory, the randomized techniques require only a constant number of passes over the data, as opposed to $O(k)$ passes for classical algorithms. In fact, it is sometimes possible to perform matrix approximation with a *single pass* over the data.

Key words. Dimension reduction, eigenvalue decomposition, interpolative decomposition, Johnson–Lindenstrauss lemma, matrix approximation, parallel algorithm, pass-efficient algorithm, principal component analysis, randomized algorithm, random matrix, rank-revealing QR factorization, singular value decomposition, streaming algorithm.

AMS subject classifications. [MSC2010] Primary: 65F30. Secondary: 68W20, 60B20.

Part I: Introduction

1. Overview. On a well-known list of the "Top 10 Algorithms" that have influenced the practice of science and engineering during the 20th century [42], we find an entry that is not really an algorithm: the *idea* of using matrix factorizations to accomplish basic tasks in numerical linear algebra. In the accompanying article [126], Stewart explains that

> The underlying principle of the decompositional approach to matrix computation is that it is not the business of the matrix algorithmicists to solve particular problems but to construct computational platforms from which a variety of problems can be solved.

Stewart goes on to argue that this point of view has had many fruitful consequences, including the development of robust software for performing these factorizations in a

[†]Department of Applied Mathematics, University of Colorado at Boulder, Boulder, CO 80309-0526. Supported by NSF awards #0748488 and #0610097.

[‡]Applied & Computational Mathematics, California Institute of Technology, MC 305-16, Pasadena, CA 91125-5000. Supported by ONR award #N000140810883.

highly accurate and provably correct manner.

The decompositional approach to matrix computation remains fundamental, but we believe the time has come to consider alternative techniques for computing these decompositions. Several reasons motivate this claim:

- A salient feature of modern applications, especially in data mining, is that the matrices are stupendously big. Classical algorithms are not always well adapted to solving the type of large-scale problems that now arise.
- In the information sciences, it is common that data are missing or inaccurate. Classical algorithms are designed to produce highly accurate matrix decompositions, but it seems profligate to spend extra computational resources when the imprecision of the data inherently limits the resolution of the output.
- Data transfer now plays a major role in the computational cost of numerical algorithms. Techniques that require few passes over the data may be substantially faster in practice, even if they require as many—or more—floating-point operations.
- As the structure of computer processors continues to evolve, it becomes increasingly important for numerical algorithms to adapt to a range of novel architectures, such as graphics processing units.

The purpose of this paper is to survey and extend recent work, including [17, 46, 58, 94, 105, 111, 112, 118, 135], which has demonstrated that *randomized* algorithms provide a powerful tool for constructing approximate matrix factorizations. These techniques are simple and effective, sometimes shockingly so. Compared with standard deterministic algorithms, the randomized methods are often faster and—perhaps surprisingly—more robust. Furthermore, they can produce factorizations that are accurate to any specified tolerance above machine precision, which allows the user to trade accuracy for speed. It is even possible to construct factorizations more accurate than those produced by classical methods. We present numerical evidence that these algorithms succeed for real computational problems.

The paper surveys randomized techniques for computing low-rank approximations. It is designed to help practitioners identify situations where randomized techniques may outperform established methods. Many of the basic ideas in this treatment are drawn from the literature, but we have assembled the components in our own fashion. This approach clarifies how random techniques interact with classical techniques to yield effective, modern algorithms supported by detailed theoretical guarantees.

REMARK 1.1. Our experience suggests that many practitioners of scientific computing view randomized algorithms as a desperate and final resort. Let us address this concern immediately. Classical Monte Carlo methods are highly sensitive to the random number generator and typically produce output with low and uncertain accuracy. In contrast, the algorithms discussed herein are insensitive to the quality of randomness and produce highly accurate results. The probability of failure is a user-specified parameter that can be rendered negligible (say, less than 10^{-15}) with a nominal impact on the computational resources required.

1.1. Approximation by low-rank matrices. The roster of standard matrix decompositions includes the pivoted QR factorization, the eigenvalue decomposition, and the singular value decomposition (SVD), all of which expose the (numerical) range of a matrix. Truncated versions of these factorizations are often used to express a

low-rank approximation of a given matrix:

$$\underset{m \times n}{\boldsymbol{A}} \approx \underset{m \times k}{\boldsymbol{B}} \underset{k \times n.}{\boldsymbol{C}}, \tag{1.1}$$

The inner dimension k is sometimes called the *numerical rank* of the matrix. When the numerical rank is much smaller than either dimension m or n, a factorization such as (1.1) allows the matrix to be stored inexpensively and to be multiplied rapidly with vectors or other matrices. The factorizations can also be used for data interpretation or to solve computational problems, such as least squares.

Matrices with low numerical rank appear in a wide variety of scientific applications. We list only a few:

- A basic method in statistics and data mining is to compute the directions of maximal variance in vector-valued data by performing *principal component analysis* (PCA) on data matrix. PCA is nothing other than a low-rank matrix approximation [70, §14.5].
- Another standard technique in data analysis is to perform low-dimensional embedding of data under the assumption that there are fewer degrees of freedom than the ambient dimension would suggest. In many cases, the method reduces to computing a partial SVD of a matrix derived from the data. See [70, §§14.8–14.9] or [33].
- The problem of estimating parameters from measured data via least-squares fitting often leads to very large systems of linear equations that are close to linearly dependent. Effective techniques for factoring the coefficient matrix lead to efficient techniques for solving the least-squares problem, [112].
- Many fast numerical algorithms for solving PDEs and for rapidly evaluating potential fields such as the fast multipole method [66] and $\mathcal{H}$-matrices [65], rely on low-rank approximations of continuum operators with exponentially decaying spectra.
- Models of multiscale physical phenomena often involve PDEs with rapidly oscillating coefficients. Techniques for *model reduction* or *coarse graining* in such environments are often based on the observation that the linear transform that maps the input data to the requested output data can be approximated by an operator of low rank [55].

1.2. Matrix approximation framework. The task of computing a low-rank approximation to a given matrix can be split naturally into two computational stages. The first is to construct a low-dimensional subspace that captures the action of the matrix. The second is to restrict the matrix to the subspace and then compute a standard factorization (QR, SVD, etc.) of the reduced matrix. To be slightly more formal, we subdivide the computation as follows.

Stage A. Compute an approximate basis for the range of the input matrix $\boldsymbol{A}$. In other words, we require a matrix $\boldsymbol{Q}$ for which

$$\boldsymbol{Q} \text{ has orthonormal columns and } \boldsymbol{A} \approx \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}. \tag{1.2}$$

We would like the basis matrix $\boldsymbol{Q}$ to contain as few columns as possible, but it is not critical to obtain the absolute minimum number at this stage as long as the approximation of the input matrix is accurate.

Stage B. Given a matrix $\boldsymbol{Q}$ that satisfies (1.2), we use $\boldsymbol{Q}$ to help compute a standard factorization (QR, SVD, etc.) of $\boldsymbol{A}$.

The task in Stage A can be executed very efficiently with random sampling methods, and it is the primary subject of this work. In the next subsection, we offer an overview of these ideas. The body of the paper provides details of the algorithms (§4) and a theoretical analysis of their performance (§§8–11).

Stage B can be completed with well-established deterministic methods. Section 3.3.3 contains an introduction to these techniques, and §5 shows how we apply them to produce low-rank factorizations.

At this stage in the development, it may not be clear why the output from Stage A facilitates our job in Stage B. Let us illustrate by describing how to obtain an approximate SVD of $\boldsymbol{A}$ given a matrix $\boldsymbol{Q}$ that satisfies (1.2). More precisely, we wish to compute matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ with orthonormal columns and a nonnegative, diagonal matrix $\boldsymbol{\Sigma}$ such that $\boldsymbol{A} \approx \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$. This goal is achieved after three simple steps:

1. Form $\boldsymbol{B} = \boldsymbol{Q}^*\boldsymbol{A}$, which yields the low-rank factorization $\boldsymbol{A} \approx \boldsymbol{Q}\boldsymbol{B}$.
2. Compute an SVD of the small matrix: $\boldsymbol{B} = \widehat{\boldsymbol{U}}\boldsymbol{\Sigma}\boldsymbol{V}^*$.
3. Set $\boldsymbol{U} = \boldsymbol{Q}\widehat{\boldsymbol{U}}$.

When $\boldsymbol{Q}$ has few columns, this procedure is efficient because we can construct $\boldsymbol{B}$ easily and compute its SVD rapidly. In practice, we can often avoid forming $\boldsymbol{B}$ explicitly by means of subtler techniques. In some cases, it is not even necessary to revisit the input matrix $\boldsymbol{A}$ during Stage B. This observation allows us to develop *single-pass algorithms*, which look at each entry of $\boldsymbol{A}$ only once.

Similar manipulations readily yield other standard factorizations, such as the pivoted QR factorization, the eigenvalue decomposition, etc.

1.3. Randomized algorithms. This paper describes a class of randomized algorithms for completing Stage A of the matrix approximation framework set forth in §1.2. We begin with more details about the approximation problem these algorithms target (§1.3.1). Afterward, we motivate the random sampling technique with a heuristic explanation (§1.3.2) that leads to a prototype algorithm (§1.3.3).

1.3.1. Problem formulations. The basic challenge in producing low-rank matrix approximations is a primitive question that we call the *fixed-precision approximation problem*. Suppose we are given a matrix $\boldsymbol{A}$ and a positive error tolerance ε. We seek a matrix $\boldsymbol{Q}$ with $k = k(\varepsilon)$ orthonormal columns such that

$$\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\| \leq \varepsilon, \tag{1.3}$$

where $\|\cdot\|$ denotes the ℓ_2 operator norm. The range of $\boldsymbol{Q}$ is a k-dimensional subspace that captures most of the action of $\boldsymbol{A}$, and we would like k to be as small as possible.

The singular value decomposition furnishes an optimal answer to the fixed-precision problem [99]. Let σ_j denote the jth largest singular value of $\boldsymbol{A}$. For each $j \geq 0$,

$$\min_{\operatorname{rank}(\boldsymbol{B}) \leq j} \|\boldsymbol{A} - \boldsymbol{B}\| = \sigma_{j+1}. \tag{1.4}$$

One way to construct a minimizer is to choose $\boldsymbol{B} = \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}$, where the columns of $\boldsymbol{Q}$ are k dominant left singular vectors of $\boldsymbol{A}$. Consequently, the minimal rank k where (1.3) holds equals the number of singular values of $\boldsymbol{A}$ that exceed the tolerance ε.

To simplify the development of algorithms, it is convenient to assume that the desired rank k is specified in advance. We call the resulting problem the *fixed-rank*

approximation problem. Given a matrix $\boldsymbol{A}$, a target rank k, and an oversampling parameter p, we seek to construct a matrix $\boldsymbol{Q}$ with $k + p$ orthonormal columns such that

$$\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\| \approx \min_{\operatorname{rank}(\boldsymbol{B}) \le k} \|\boldsymbol{A} - \boldsymbol{B}\|. \tag{1.5}$$

Although there exists a minimizer $\boldsymbol{Q}$ that solves the fixed rank problem for $p = 0$, the opportunity to use a small number of additional columns provides a flexibility that is crucial for the effectiveness of the computational methods we discuss.

We will demonstrate that algorithms for the fixed-rank problem can be adapted to solve the fixed-precision problem. The connection is based on the observation that we can build the basis matrix $\boldsymbol{Q}$ incrementally and, at any point in the computation, we can inexpensively estimate the residual error $\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\|$. Refer to §4.4 for the details of this reduction.

1.3.2. Intuition. To understand how randomness helps us solve the fixed-rank problem, it is helpful to consider some motivating examples.

First, suppose that we seek a basis for the range of a matrix $\boldsymbol{A}$ with *exact* rank k. Draw a random vector $\boldsymbol{\omega}$, and form the product $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{\omega}$. For now, the precise distribution of the random vector is unimportant; just think of $\boldsymbol{y}$ as a random sample from the range of $\boldsymbol{A}$. Let us repeat this sampling process k times:

$$\boldsymbol{y}^{(i)} = \boldsymbol{A}\boldsymbol{\omega}^{(i)}, \quad i = 1, 2, \dots, k.$$

Owing to the randomness, the set $\{\boldsymbol{\omega}^{(i)}\}$ of random vectors is likely to be in general linear position. In particular, the random vectors form a linearly independent set and no linear combination falls in the null space of $\boldsymbol{A}$. As a result, the set $\{\boldsymbol{y}^{(i)}\}_{i=1}^{k}$ of sample vectors is also linearly independent, so it spans the range of $\boldsymbol{A}$. Therefore, to produce an orthonormal basis for the range of $\boldsymbol{A}$, we just need to orthonormalize the sample vectors.

Now, imagine that $\boldsymbol{A} = \boldsymbol{B} + \boldsymbol{E}$ where $\boldsymbol{B}$ is a rank-k matrix and $\boldsymbol{E}$ is a small perturbation. If we generate exactly k samples from the range of $\boldsymbol{A}$, it is unlikely that the linear span of these vectors aligns closely with the range of $\boldsymbol{B}$ owing to the perturbation $\boldsymbol{E}$. As a result, we might miss directions that contribute significantly to the action of $\boldsymbol{A}$. To improve the situation, we fix a small integer p representing some degree of oversampling, and we generate $k + p$ samples

$$\boldsymbol{y}^{(i)} = \boldsymbol{A}\boldsymbol{\omega}^{(i)}, \quad i = 1, 2, \dots, k + p.$$

Remarkably, a very small amount of oversampling is sufficient for the span of the samples to capture the range of $\boldsymbol{B}$ to high accuracy with negligible failure probability. In fact, $p = 5$ or $p = 10$ often gives superb results.

1.3.3. A prototype algorithm. The intuitive approach of §1.3.2 can be applied to general matrices. Omitting computational details for now, we formalize the procedure in the figure labeled Proto-Algorithm.

This simple algorithm is by no means new. It is essentially the first step of an orthogonal iteration with a random initial subspace [61, §7.3.2]. The novelty comes from the additional observation that the initial subspace should have a slightly higher dimension than the invariant subspace we are trying to approximate. With this revision, it is often the case that *no further iteration is required* to obtain a high-quality solution to (1.5). We believe this idea can be traced to [94, 105].

PROTO-ALGORITHM: SOLVING THE FIXED-RANK PROBLEM

Given an $m \times n$ matrix $\boldsymbol{A}$, a target rank k, and an oversampling parameter p, this procedure computes an $m \times (k+p)$ matrix $\boldsymbol{Q}$ whose columns are orthonormal and whose range approximates the range of $\boldsymbol{A}$.

1 Draw a random $n \times (k+p)$ test matrix $\boldsymbol{\Omega}$.
2 Form the matrix product $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{\Omega}$.
3 Construct a matrix $\boldsymbol{Q}$ whose columns form an orthonormal basis for the range of $\boldsymbol{Y}$.

In order to invoke the proto-algorithm with confidence, we must address several practical and theoretical issues:

- What random matrix $\boldsymbol{\Omega}$ should we use? How much oversampling do we need?
- How is the basis $\boldsymbol{Q}$ determined from the sample matrix $\boldsymbol{Y}$?
- What are the computational costs?
- How can we solve the fixed-precision problem (1.3) when the numerical rank of the matrix is not known in advance?
- How can we use the basis $\boldsymbol{Q}$ to compute other matrix factorizations?
- Does the randomized method work for problems of practical interest? How does its speed/accuracy/robustness compare with standard techniques?
- What error bounds can we expect? With what probability?

The next few sections provide a summary of the answers to these questions. We describe several problem regimes where the proto-algorithm can be implemented efficiently, and we present a theorem that describes the performance of the most important instantiation. Finally, we elaborate on how these ideas can be applied to perform principal component analysis (PCA) of large data matrices. The rest of the paper contains a more exhaustive treatment—including pseudocode, numerical experiments, and a detailed theory.

1.4. A comparison between randomized and traditional techniques. To select an appropriate computational method for finding a low-rank approximation to a matrix, the practitioner must take into account the properties of the matrix. Is it dense or sparse? Does it fit in fast memory or is it stored out of core? Does the singular spectrum decay quickly or slowly? The behavior of a numerical linear algebra algorithm may depend on all these factors [13, 61, 131]. To facilitate a comparison between classical and randomized techniques, we summarize their relative performance in each of three representative environments. Section 6 contains a more in-depth treatment.

We focus on the task of computing an approximate SVD of an $m \times n$ matrix $\boldsymbol{A}$ with numerical rank k. For randomized schemes, Stage A generally dominates the cost of Stage B in our matrix approximation framework (§1.2). Within Stage A, the computational bottleneck is usually the matrix–matrix product $\boldsymbol{A}\boldsymbol{\Omega}$ in Step 2 of the proto-algorithm (§1.3.3). The power of randomized algorithms stems from the fact that we can reorganize this matrix multiplication for maximum efficiency in a variety of computational architectures.

1.4.1. A general dense matrix that fits in fast memory. A modern deterministic technique for approximate SVD is to perform a rank-revealing QR fac-

torization of the matrix [68] and then to manipulate the factors to obtain the final factorization. The cost of this approach is $\mathrm{O}(kmn)$ floating-point operations, or *flops*.

In contrast, randomized schemes can produce an approximate SVD using only $\mathrm{O}(mn \log(k) + (m+n)k^2)$ flops. The gain in asymptotic complexity is achieved by using a random matrix $\boldsymbol{\Omega}$ that has some internal structure, which allows us to evaluate the product $\boldsymbol{A\Omega}$ rapidly. For example, randomizing and subsampling the discrete Fourier transform works well. Sections 4.6 and 11 contain more information on this approach.

1.4.2. A matrix for which matrix–vector products can be evaluated rapidly. When $\boldsymbol{A}$ is sparse or structured, it can often be applied rapidly to a vector. In this case, the classical technique of choice is a Krylov subspace method, such as the Lanczos or Arnoldi algorithm. These techniques have an asymptotic cost of $\mathrm{O}(k\,T_{\mathrm{mult}} + (m+n)k^2)$ flops, where T_{mult} denotes the cost of a matrix–vector multiplication. Note, however, that these approaches are delicate to implement and their convergence can be slow when the matrix has an unfortunate singular spectrum.

In this environment, we can apply randomized methods using a Gaussian test matrix $\boldsymbol{\Omega}$ to complete the factorization at the same cost, $\mathrm{O}(k\,T_{\mathrm{mult}} + (m+n)k^2)$ flops. Random schemes have at least two distinct advantages over Krylov methods. First, the randomized methods produce highly accurate matrix approximations even in the absence of gaps/jumps in the singular spectrum (which is not the case for the classical Lanczos and Arnoldi methods). Second, the matrix–vector multiplies required to form $\boldsymbol{A\Omega}$ can be performed *in parallel*. The flexibility to restructure the computation often leads to dramatic accelerations in practice.

1.4.3. A general dense matrix stored in slow memory or streamed. When the input matrix is too large to fit in core memory, the cost of transferring the matrix from slow memory typically dominates the cost of performing the arithmetic. The standard techniques for low-rank approximation described in §1.4.1 require $\mathrm{O}(k)$ passes over the matrix, which can be prohibitively expensive.

In contrast, the proto-algorithm of §1.3.3 requires only one pass over the data to produce the approximate basis $\boldsymbol{Q}$ for Stage A of the approximation framework. This straightforward approach, unfortunately, is not accurate enough for matrices whose singular spectrum decays slowly, but we can address this problem using very few (say, 2 to 4) additional passes over the data [111]. See §1.6 or §4.5 for more discussion.

Typically, Stage B uses one additional pass over the matrix to construct the approximate SVD. With slight modifications, however, the two-stage randomized scheme can be revised so that it only makes a single pass over the data. Refer to §5.4 for information.

1.5. Performance analysis. A principal goal of this paper is to provide a detailed analysis of the performance of the proto-algorithm described in §1.3.3. This investigation produces precise error bounds, expressed in terms of the singular values of the input matrix. Furthermore, we determine how several choices of the random matrix $\boldsymbol{\Omega}$ impact the behavior of the algorithm.

Let us offer a taste of these results. The following theorem describes the average-case performance of the proto-algorithm with a Gaussian test matrix. It is a simplified version of Theorem 10.6.

THEOREM 1.1. *Suppose that $\boldsymbol{A}$ is a real $m \times n$ matrix. Select a target rank k and an oversampling parameter $p \geq 2$, where $k + p \leq \min\{m, n\}$. Execute the proto-*

algorithm with a standard Gaussian test matrix to obtain an $m \times (k+p)$ matrix $\boldsymbol{Q}$ with orthonormal columns. Then

$$\mathbb{E}\,\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\| \le \left[1 + \frac{4\sqrt{k+p}}{p-1} \cdot \sqrt{\min\{m,n\}}\right] \sigma_{k+1}, \tag{1.6}$$

where $\mathbb{E}$ denotes expectation with respect to the random test matrix and σ_{k+1} is the $(k+1)$th singular value of $\boldsymbol{A}$.

It is natural that the error bound contains one copy of σ_{k+1} owing to (1.4). At worst, the algorithm produces a basis whose error is larger by a small polynomial factor. Observe that a moderate amount of oversampling results in a substantial performance gain. Moreover, the error bound (1.6) in the randomized algorithm is slightly sharper than comparable bounds for deterministic techniques based on rank-revealing QR algorithms [68].

The reader might be worried about whether the expectation provides a useful account of the approximation error. Fear not: the actual outcome of the algorithm is *almost always* the typical outcome because of measure concentration effects. As we discuss in §10.3, the probability that the error satisfies

$$\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\| \le \left[1 + 11\sqrt{k+p} \cdot \sqrt{\min\{m,n\}}\right] \sigma_{k+1} \tag{1.7}$$

is at least $1 - 6 \cdot p^{-p}$ under very mild assumptions on p. This fact justifies the use of an oversampling term as small as $p = 5$. This simplified estimate is very similar to the major results in [94].

The theory developed in this paper provides much more detailed information about the performance of the proto-algorithm.

- When the singular values of $\boldsymbol{A}$ decay slightly, the error $\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\|$ does not depend on the dimensions of the matrix (§§10.2–10.3).
- We can reduce the size of the bracket in the error bound (1.6) by combining the proto-algorithm with a power iteration (§10.4). For an example, see §1.6 below.
- For the structured random matrices we mentioned in §1.4.1, related error bounds are in force (§11).
- We can obtain inexpensive *a posteriori* error estimates to verify the quality of the approximation (§4.3).

1.6. Example: Principal Component Analysis. We conclude this introduction with a short discussion of how these ideas allow us to perform principal component analysis (PCA) on large data matrices, which is a compelling application of randomized matrix approximation [111].

Suppose that $\widetilde{\boldsymbol{A}}$ is an $m \times n$ matrix holding statistical data, where each column of $\widetilde{\boldsymbol{A}}$ carries the data from one experiment and each row contains measurements of a single variable. We form the *centered* matrix $\boldsymbol{A}$ by subtracting the empirical mean from each variable. In other terms, $a_{ij} = \tilde{a}_{ij} - (1/n)\sum_j \tilde{a}_{ij}$ so that each row of $\boldsymbol{A}$ sums to zero. The first k principal components of the data (i.e., the directions of maximal variance) are the k dominant left singular vectors of the input matrix $\boldsymbol{A}$.

We can evidently perform PCA by computing an approximate SVD of the centered matrix $\boldsymbol{A}$, and the two-stage randomized method offers a natural approach. Unfortunately, the simplest version of this scheme is inadequate for PCA because the

ALGORITHM: RANDOMIZED PCA

Given an $m \times n$ matrix $\boldsymbol{A}$, the number k of principal components, and an exponent q, this procedure computes an approximate rank-$2k$ factorization $\boldsymbol{U\Sigma V}^*$. The columns of $\boldsymbol{U}$ estimate the first $2k$ principal components of $\boldsymbol{A}$.

Stage A:
1 Generate an $n \times 2k$ Gaussian test matrix $\boldsymbol{\Omega}$.
2 Form $\boldsymbol{Y} = (\boldsymbol{AA}^*)^q \boldsymbol{A\Omega}$ by multiplying alternately with $\boldsymbol{A}$ and $\boldsymbol{A}^*$
3 Construct a matrix $\boldsymbol{Q}$ whose columns form an orthonormal basis for the range of $\boldsymbol{Y}$.

Stage B:
1 Form $\boldsymbol{B} = \boldsymbol{Q}^*\boldsymbol{A}$.
2 Compute an SVD of the small matrix: $\boldsymbol{B} = \widehat{\boldsymbol{U}}\boldsymbol{\Sigma V}^*$.
3 Set $\boldsymbol{U} = \boldsymbol{Q}\widehat{\boldsymbol{U}}$.

singular spectrum of the data matrix often decays quite slowly. To address this difficulty, we incorporate q steps of a power iteration where $q = 1, 2$ is usually sufficient in practice. The complete scheme appears below as the Randomized PCA algorithm. For refinements, see the discussion in §§4–5.

This procedure requires only $2(q+1)$ passes over the matrix, so it is efficient even for matrices stored out-of-core. The flop count is

$$T_{\text{randPCA}} \sim qk\, T_{\text{mult}} + k^2(m+n),$$

where T_{mult} is the cost of a matrix–vector multiply with $\boldsymbol{A}$ or $\boldsymbol{A}^*$.

We have the following theorem on the performance of this method, which is a consequence of Corollary 10.10.

THEOREM 1.2. *Suppose that $\boldsymbol{A}$ is a real $m \times n$ matrix. Select an exponent q and a target number k of principal components, where $2 \le k \le 0.5 \min\{m, n\}$. Execute the randomized PCA algorithm to obtain a rank-$2k$ factorization $\boldsymbol{U\Sigma V}^*$. Then*

$$\mathbb{E}\,\|\boldsymbol{A} - \boldsymbol{U\Sigma V}^*\| \le \left[1 + 4\sqrt{\frac{2\min\{m,n\}}{k-1}}\right]^{1/(2q+1)} \sigma_{k+1}, \tag{1.8}$$

where $\mathbb{E}$ denotes expectation with respect to the random test matrix and σ_{k+1} is the $(k+1)$th singular value of $\boldsymbol{A}$.

This result is new. Observe that the bracket in (1.8) is essentially the same as the bracket in the basic error bound (1.6). We find that the power iteration drives the leading constant to one exponentially fast as the power q increases. Since the rank-k approximation of $\boldsymbol{A}$ can never achieve an error smaller than σ_{k+1}, the randomized procedure estimates $2k$ principal components that carry essentially as much variance as the first k actual principal components. Truncating the approximation to the first k terms typically produces very accurate results.

1.7. Outline of paper. The paper is organized into three parts: an introduction (§§1–3), a description of the algorithms (§§4–7), and a theoretical performance analysis (§§8–11). Each part commences with a short internal outline, and the three segments

are more or less self-contained. After a brief review of our notation in §§3.1–3.2, the reader can proceed to either the algorithms or the theory part.

2. Related work and historical context. The idea of using randomness to develop algorithms for numerical linear algebra appears to be fairly recent, although we can trace the intellectual origins back to earlier work in computer science and—especially—to probabilistic methods in geometric analysis. This section presents an aerial view of this young field. We begin with a survey of randomized methods for matrix approximation; then we attempt to trace some of the ideas backward to their sources. We apologize in advance for any inaccuracies or omissions, and we welcome any feedback that might improve this account.

2.1. Randomized matrix approximation. Matrices of low numerical rank contain little information relative to their apparent dimension owing to the linear dependency in their columns (or rows). As a result, it is reasonable to expect that these matrices can be approximated with far fewer degrees of freedom. A less obvious fact is that randomized schemes can be used to produce these approximations efficiently.

Several types of approximation techniques build on this idea. These methods all follow the same basic pattern:

1. Preprocess the matrix, usually to calculate sampling probabilities.
2. Take random samples from the matrix, where the term *sample* refers generically to a linear function of the matrix.
3. Postprocess the samples to compute a final approximation, typically with classical techniques from numerical linear algebra. This step may require another look at the matrix.

We continue with a description of the most common approximation schemes.

2.1.1. Sparsification. The simplest approach to matrix approximation is the method of *sparsification* or the related technique of *quantization*. The goal of this process is to replace the matrix by a surrogate that contains far fewer nonzero entries; quantization further restricts these nonzero entries to a small set of values. Sparsification can be used to reduce storage or to accelerate computations by reducing the cost of matrix–vector and matrix–matrix multiplies [96, Ch. 6]. The paper [36] describes applications in optimization.

Sparsification typically involves very simple elementwise calculations. Each entry in the approximation is drawn independently at random from a distribution determined from the corresponding entry of the input matrix. The expected value of the random approximation equals the original matrix, but the distribution is designed so that a typical realization is much sparser.

The first method of this form was devised by Achiloptas and McSherry [2], who built on earlier work on graph sparsification due to Karger [76, 77]. Arora–Hazan–Kale presented a different sampling method in [7]. See [60, 123] for some recent work on sparsification.

2.1.2. Column selection methods. A second approach to matrix approximation is based on the idea that a small set of columns describes most of the action of a numerically low-rank matrix. Indeed, classical existential results [117] demonstrate that every $m \times n$ matrix $\boldsymbol{A}$ contains a k-column submatrix $\boldsymbol{C}$ for which

$$\left\| \boldsymbol{A} - \boldsymbol{C}\boldsymbol{C}^{\dagger}\boldsymbol{A} \right\| \leq \sqrt{1 + k(n-k)} \cdot \left\| \boldsymbol{A} - \boldsymbol{A}_{(k)} \right\|, \tag{2.1}$$

where k is a parameter, the dagger † denotes the pseudoinverse, and $\boldsymbol{A}_{(k)}$ is a best rank-k approximation of $\boldsymbol{A}$. In general, it is NP-hard to produce a submatrix $\boldsymbol{C}$ that minimizes the left-hand side [30]. Nevertheless, there are efficient deterministic algorithms, such as the rank-revealing QR method of [68], that can nearly achieve the error bound (2.1).

There is a class of randomized algorithms that approach the fixed-rank approximation problem (1.5) using this intuition. These methods first compute a sampling probability for each column, either using the squared Euclidean norms of the columns or their *leverage scores*. (Leverage scores reflect the relative importance of the columns to the action of the matrix; they can be calculated easily from the dominant k right singular vectors of the matrix.) Columns are then selected randomly according to this distribution. Afterward, a postprocessing step is invoked to produce a more refined approximation of the matrix.

We believe that the earliest method of this form appeared in a 1998 paper of Frieze–Kannan–Vempala [57, 58]. Later, this work was refined substantially by Drineas–Kannan–Mahoney [46]. The basic algorithm samples columns from a distribution related to the squared ℓ_2 norms of the columns. This sampling step produces a small column submatrix whose range is aligned with the range of the input matrix. The final approximation is obtained from a truncated SVD of the submatrix. Given a target rank k and a parameter $\varepsilon > 0$, this approach produces a rank-k approximation $\boldsymbol{B}$ that satisfies

$$\|\boldsymbol{A} - \boldsymbol{B}\|_{\mathrm{F}} \leq \left\|\boldsymbol{A} - \boldsymbol{A}_{(k)}\right\|_{\mathrm{F}} + \varepsilon \|\boldsymbol{A}\|_{\mathrm{F}}, \tag{2.2}$$

where $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm. The algorithm of [46] is computationally efficient and requires only a constant number of passes over the data.

Rudelson and Vershynin later showed that the same column sampling method also yields spectral-norm error bounds [116]. The techniques in their paper have been very influential; their work has already found other applications in randomized regression [51], sparse approximation [132], and compressive sampling [21].

Deshpande et al. [39, 40] demonstrated that the error in the column sampling approach can be improved by iteration and adaptive sampling. They showed that it is possible to produce a rank-k matrix $\boldsymbol{B}$ that satisfies

$$\|\boldsymbol{A} - \boldsymbol{B}\|_{\mathrm{F}} \leq (1 + \varepsilon) \left\|\boldsymbol{A} - \boldsymbol{A}_{(k)}\right\|_{\mathrm{F}} \tag{2.3}$$

using a k-pass algorithm. Around the same time, Har-Peled [69] independently developed a recursive algorithm that offers the same approximation guarantees.

Drineas et al. and Boutsidis et al. have also developed randomized algorithms for the *column subset selection problem*, which requests a column submatrix $\boldsymbol{C}$ that achieves a bound of the form (2.1). Using the methods of Rudelson and Vershynin [116], they showed that sampling columns according to their leverage scores is likely to produce the required submatrix [49, 50]. Subsequent work [17, 18] showed that postprocessing the sampled columns with a rank-revealing QR algorithm can reduce the number of output columns required while improving the classical existential error bound (2.1). The argument in [17] explicitly decouples the linear algebraic part of the analysis from the random matrix theory. The theoretical analysis in the present work relies on a very similar technique.

2.1.3. Approximation by dimension reduction. A third approach to matrix approximation is based on the concept of *dimension reduction*. Since the rows of a

low-rank matrix are linearly dependent, they can be embedded into a low-dimensional space without altering their geometric properties substantially. A random linear map provides an efficient, nonadaptive way to perform this embedding. (Column sampling can also be viewed as an adaptive form of dimension reduction.)

The proto-algorithm we set forth in §1.3.3 is simply a dual description of the dimension reduction approach: collecting random samples from the column space of the matrix is equivalent to reducing the dimension of the rows. No precomputation is required to obtain the sampling distribution, but the sample itself takes some work to collect. Afterward, we orthogonalize the samples as preparation for constructing various matrix approximations.

We believe that the idea of using dimension reduction for algorithmic matrix approximation first appeared in a 1998 paper of Papadimitriou et al. [104, 105], who described an application to latent semantic indexing (LSI). They suggested projecting the input matrix onto a random subspace and compressing the original matrix to (a subspace of) the range of the projected matrix. They established error bounds that echo the result (2.2) of Frieze et al. [58]. Although the Euclidean column selection method is a more computationally efficient way to obtain this type of error bound, dimension reduction has other advantages, e.g., in terms of accuracy.

Somewhat later, Martinsson–Rokhlin–Tygert studied dimension reduction using a Gaussian transform matrix, and they demonstrated that the approach performs much better than earlier analyses had suggested [94]. Their work highlights the importance of oversampling, and their error bounds are very similar to the estimate (1.7) we presented in the introduction. They also demonstrated that dimension reduction can be used to compute an interpolative decomposition of the input matrix, which is essentially equivalent to performing column subset selection.

Sarlós argued in [118] that the computational costs of dimension reduction can be reduced substantially by using structured random maps proposed by Ailon–Chazelle [3]. Sarlós used these ideas to develop efficient randomized algorithms for least-squares problems; he also studied approximate matrix multiplication and low-rank matrix approximation. The recent paper [103] analyzes a very similar matrix approximation algorithm using Rudelson and Vershynin's methods [116].

The initial work on structured dimension reduction did not immediately yield algorithms for low-rank matrix approximation that were superior to classical techniques. Woolfe et al. showed how to obtain an improvement in asymptotic computational cost, and they applied these techniques to problems in scientific computing [135]. Related work includes [88, 90]. Very recently, Clarkson and Woodruff have streamlined these methods to obtain single-pass algorithms that are suitable for the streaming-data model [32].

Rokhlin–Szlam–Tygert have shown that combining dimension reduction with a power iteration is an effective way to improve its performance. These ideas lead to very efficient randomized methods for large-scale PCA [111]. Related ideas appear in work of Roweis [113].

2.1.4. Approximation by submatrices. The matrix approximation literature contains a subgenre that discusses methods for building an approximation from a submatrix and computed coefficient matrices. For example, we can construct an approximation using a subcollection of columns (the interpolative decomposition), a subcollection of rows and a subcollection of columns (the CUR decomposition), or a square submatrix (the matrix skeleton). This type of decomposition was developed and studied in several papers, including [29, 64, 125]. For data analysis applications,

see the recent paper [92].

A number of works develop randomized algorithms for this class of matrix approximations. Drineas et al. have developed techniques for computing CUR decompositions, which express $\boldsymbol{A} \approx \boldsymbol{CUR}$, where $\boldsymbol{C}$ and $\boldsymbol{R}$ denote small column and row submatrices of $\boldsymbol{A}$ and where $\boldsymbol{U}$ is a small linkage matrix. These methods identify columns (rows) that approximate the range (corange) of the matrix; the linkage matrix is then computed by solving a small least-squares problem. A randomized algorithm for CUR approximation with controlled absolute error appears in [47]; a relative error algorithm appears in [50]. We also mention a paper on computing a closely related factorization called the *compact matrix decomposition* [128].

It is also possible to produce interpolative decompositions and matrix skeletons using randomized methods, as discussed in [94, 111].

2.1.5. Other numerical problems. The literature contains a variety of other randomized algorithms for solving standard problems in and around numerical linear algebra. We list some of the basic references.

Tensor skeletons. Randomized column selection methods can be used to produce CUR-type decompositions of higher-order tensors [48].

Matrix multiplication. Column selection and dimension reduction techniques can be used to accelerate the multiplication of rank-deficient matrices [45, 118]. See also [10].

Overdetermined linear systems. The randomized Kaczmarz algorithm is a linearly convergent iterative method that can be used to solve overdetermined linear systems [102, 127].

Overdetermined least squares. Fast dimension-reduction maps can sometimes accelerate the solution of overdetermined least-squares problems [51, 118].

Nonnegative least squares. Fast dimension reduction maps can be used to reduce the size of nonnegative least-squares problems [16].

Preconditioned least squares. Randomized matrix approximations can be used to precondition conjugate gradient to solve least-squares problems [112].

Other regression problems. Randomized algorithms for ℓ_1 regression are described in [31]. Regression in ℓ_p for $p \in [1, \infty)$ has also been considered [34].

Facility location. The Fermat–Weber facility location problem can be viewed as matrix approximation with respect to a different discrepancy measure. Randomized algorithms for this type of problem appear in [121].

2.1.6. Compressive sampling. Although randomized matrix approximation and compressive sampling are based on some common intuitions, it is facile to consider either one as a subspecies of the other. We offer a short overview of the field of compressive sampling—especially the part connected with matrices—so we can highlight some of the differences.

The theory of compressive sampling starts with the observation that many types of vector-space data are *compressible*. That is, the data are approximated well using a short linear combination of basis functions drawn from a fixed collection [44]. For example, natural images are well approximated in a wavelet basis; numerically low-rank matrices are well approximated as a sum of rank-one matrices. The idea behind compressive sampling is that suitably chosen random samples from this type of compressible object carry a large amount of information. Furthermore, it is possible to reconstruct the compressible object from a small set of these random samples, often by solving a convex optimization problem. The initial discovery works of Candès–

Romberg–Tao [22] and Donoho [43] were written in 2004.

The earliest work in compressive sampling focused on vector-valued data; soon after, researchers began to study compressive sampling for matrices. In 2007, Recht–Fazel–Parillo demonstrated that it is possible to reconstruct a rank-deficient matrix from Gaussian measurements [110]. More recently, Candès–Recht [24] and Candès–Tao [26] considered the problem of completing a low-rank matrix from a random sample of its entries. Matrix completion can be viewed as the problem that is dual to sparsification.

The usual goals of compressive sampling are (i) to design a method for collecting informative, nonadaptive data about a compressible object and (ii) to reconstruct a compressible object given some measured data. In both cases, there is an implicit assumption that we have limited—if any—access to the underlying data.

In the problem of matrix approximation, we typically have a complete representation of the matrix at our disposal. The point is to compute a simpler representation as efficiently as possible under some operational constraints. In particular, we would like to perform as little computation as we can, but we are usually allowed to revisit the input matrix. Because of the different focus, randomized matrix approximation algorithms require fewer random samples from the matrix and use fewer computational resources than compressive sampling reconstruction algorithms.

2.2. Origins. This section attempts to identify some of the major threads of research that ultimately led to the development of the randomized techniques we discuss in this paper.

2.2.1. Random embeddings. The field of random embeddings is a major precursor to randomized matrix approximation. In a celebrated 1984 paper [75], Johnson and Lindenstrauss showed that the pairwise distances among a collection of N points in a Euclidean space are approximately maintained when the points are mapped randomly to a Euclidean space of dimension $\mathrm{O}(\log N)$. In other words, random embeddings preserve Euclidean geometry. Shortly afterward, Bourgain showed that appropriate random low-dimensional embeddings preserve the geometry of point sets in finite-dimensional ℓ_1 spaces [15].

These observations suggest that we might be able to solve some computational problems of a geometric nature more efficiently by translating them into a lower-dimensional space and solving them there. This idea was cultivated by the theoretical computer science community beginning in the late 1980s, with research flowering in the late 1990s. In particular, nearest-neighbor search can benefit from dimension-reduction techniques [73, 79, 81]. The papers [57, 104] were apparently the first to apply this approach to linear algebra.

Around the same time, researchers became interested in simplifying the form of dimension reduction maps and improving the computational cost of applying the map. Several researchers developed refined results on the performance of a Gaussian matrix as a linear dimension reduction map [35, 73, 95]. Achlioptas demonstrated that discrete random matrices would serve nearly as well [1]. In 2006, Ailon and Chazelle proposed the *fast Johnson–Lindenstrauss transform* [3], which combines the speed of the FFT with the favorable embedding properties of a Gaussian matrix. Subsequent refinements appear in [4, 89]. Sarlós then imported these techniques to study several problems in numerical linear algebra, which has led to some of the fastest algorithms currently available [90, 135].

2.2.2. Data streams. Muthukrishnan argues that a distinguishing feature of modern data is the manner in which it is *presented* to us. The sheer volume of information and the speed at which it must be processed tax our ability to *transmit* the data elsewhere, to *compute* complicated functions on the data, or to *store* a substantial part of the data [101, §3]. As a result, computer scientists have started to develop algorithms that can address familiar computational problems under these novel constraints. The data stream phenomenon is one of the primary justifications cited by [45] for developing pass-efficient methods for numerical linear algebra problems, and it is also the focus of the recent treatment [32].

One of the methods for dealing with massive data sets is to maintain *sketches*, which are small summaries that allow functions of interest to be calculated. In the simplest case, a sketch is simply a random projection of the data, but it might be a more sophisticated object [101, §5.1]. The idea of sketching can be traced to the fundamental work of Alon et al. [5, 6].

2.2.3. Numerical linear algebra. Classically, the field of numerical linear algebra has focused on developing deterministic algorithms that produce highly accurate matrix approximations with provable guarantees. Nevertheless, randomized techniques have appeared in several environments.

One of the original examples is the use of random models for arithmetical errors, which was pioneered by von Neumann and Goldstine [133, 134]. These two papers stand among the first works to study the properties of random matrices. The earliest numerical linear algebra algorithm that uses randomized techniques is apparently Dixon's method for estimating norms and condition numbers [41].

Another situation where randomness commonly arises is the initialization of iterative methods for computing invariant subspaces. For example, most numerical linear algebra texts advocate random selection of the starting vector for the power method because it ensures that the vector has a nonzero component in the direction of a dominant eigenvector. Woźniakowski and coauthors have analyzed the performance of the power method and the Lanczos iteration given a random starting vector [80, 87].

Among other interesting applications of randomness, we mention the work by Parker and Pierce, which applies a randomized FFT to eliminate pivoting in Gaussian elimination [106], work by Demmel et al. who have studied randomization in connection with the stability of fast methods for linear algebra [38], and work by Le and Parker utilizing randomized methods for stabilizing fast linear algebraic computations based on recursive algorithms (such as Strassen's) [82].

2.2.4. Scientific computing. One of the first algorithmic applications of randomness is the method of Monte Carlo integration introduced by Von Neumann and Ulam [97], and its extensions, such as the Metropolis algorithm for simulations in statistical physics. (See [9] for an introduction). The most basic technique is to estimate an integral by sampling m points from the measure and computing an empirical mean of the integrand evaluated at the sample locations:

$$\int f(x)\,\mathrm{d}\mu(x) \approx \frac{1}{m}\sum_{i=1}^{m} f(X_i),$$

where X_i are independent and identically distributed according to the probability measure μ. The law of large numbers (usually) ensures that this approach produces the correct result in the limit as $m \to \infty$. Unfortunately, the approximation error

typically has a standard deviation of $m^{-1/2}$, and the method provides no certificate of success.

The disappointing computational profile of Monte Carlo integration seems to have inspired a distaste for randomized approaches within the scientific computing community. Fortunately, there are many other types of randomized algorithms—such as the ones in this paper—that do not suffer from the same shortcomings.

2.2.5. Geometric functional analysis. There is one more character that plays a central role in our story: the probabilistic method in geometric analysis. Many of the algorithms and proof techniques ultimately come from work in this beautiful but recondite corner of mathematics.

Dvoretsky's theorem [52] states (roughly) that every infinite-dimensional Banach space contains an n-dimensional subspace whose geometry is essentially the same as an n-dimensional Hilbert space, where n is an arbitrary natural number. In 1971, V. D. Milman developed an audacious new proof of this result by showing that a *random* n-dimensional subspace of an N-dimensional Banach space has this property with exceedingly high probability, provided that N is large enough [98]. Milman's article is cited as the debut of the *concentration of measure phenomenon*, which is a geometric interpretation of the classical idea that regular functions of independent random variables rarely deviate far from their mean. This work opened a new era in geometric analysis where the probabilistic method became a basic instrument.

Another prominent example of measure concentration is Kashin's computation of the Gel'fand widths of the ℓ_1 ball [78], subsequently refined in [59]. This work showed that a *random* $(N-n)$-dimensional projection of the N-dimensional ℓ_1 ball has an astonishingly small Euclidean diameter: approximately $\sqrt{(1 + \log(N/n))/n}$. In contrast, a nonzero projection of the ℓ_2 ball always has Euclidean diameter one. This basic geometric fact undergirds recent developments in compressive sampling [23].

We have already described a third class of examples: the randomized embeddings of Johnson–Lindenstrauss [75] and of Bourgain [15].

Finally, we mention Maurey's technique of empirical approximation. The original work was unpublished; one of the earliest applications appears in [27, §1]. Although Maurey's idea has not received as much press as the examples above, it can lead to simple and efficient algorithms for sparse approximation. For some examples in machine learning, consider [8, 86, 109, 119]

The importance of random constructions in the geometric analysis community has led to the development of powerful techniques for studying random matrices. Classical random matrix theory focuses on a detailed asymptotic analysis of the spectral properties of special classes of random matrices. In contrast, geometric analysts know methods for determining the approximate behavior of rather complicated finite-dimensional random matrices. See [37] for a fairly current survey article. We also make special mention of the works of Rudelson [114] and Rudelson–Vershynin [116], which describe powerful tools for studying random matrices drawn from certain discrete distributions. Their papers are rooted deeply in the field of geometric functional analysis, but they reach out toward computational applications. The analysis in the present paper leans heavily on ideas drawn from the references in this paragraph.

3. Linear algebraic preliminaries. This section summarizes the background we need for the detailed description of randomized algorithms in §§4–6 and the analysis in §§8–11. We introduce notation in §3.1, describe some standard matrix decompositions in §3.2, and briefly review standard techniques for computing matrix factorizations in §3.3.

3.1. Basic definitions. The standard Hermitian geometry for $\mathbb{C}^n$ is induced by the inner product

$$\langle \boldsymbol{x},\ \boldsymbol{y} \rangle = \boldsymbol{x} \cdot \boldsymbol{y} = \sum\nolimits_j x_j\, \overline{y}_j.$$

The associated norm is

$$\|\boldsymbol{x}\|^2 = \langle \boldsymbol{x},\ \boldsymbol{x} \rangle = \sum\nolimits_j |x_j|^2.$$

We usually measure the magnitude of a matrix $\boldsymbol{A}$ with the operator norm

$$\|\boldsymbol{A}\| = \max_{\boldsymbol{x} \neq \boldsymbol{0}} \frac{\|\boldsymbol{A}\boldsymbol{x}\|}{\|\boldsymbol{x}\|},$$

which is often referred to as the *spectral norm*. The Frobenius norm is given by

$$\|\boldsymbol{A}\|_{\mathrm{F}} = \left[\sum\nolimits_{jk} |a_{jk}|^2 \right]^{1/2}.$$

The matrix conjugate transpose, or adjoint, of a matrix $\boldsymbol{A}$ is denoted $\boldsymbol{A}^*$. The important identity

$$\|\boldsymbol{A}\|^2 = \|\boldsymbol{A}^*\boldsymbol{A}\| = \|\boldsymbol{A}\boldsymbol{A}^*\|$$

holds for each matrix $\boldsymbol{A}$.

We say that a matrix $\boldsymbol{U}$ is *orthonormal* if its columns form an orthonormal set with respect to the Hermitian inner product. An orthonormal matrix $\boldsymbol{U}$ preserves geometry in the sense that $\|\boldsymbol{U}\boldsymbol{x}\| = \|\boldsymbol{x}\|$ for every vector $\boldsymbol{x}$. A *unitary* matrix is a square orthonormal matrix, and an *orthogonal* matrix is a real unitary matrix. Unitary matrices satisfy the relations $\boldsymbol{U}\boldsymbol{U}^* = \boldsymbol{U}^*\boldsymbol{U} = \mathbf{I}$. Both the operator norm and the Frobenius norm are *unitarily invariant*, which means that

$$\|\boldsymbol{U}\boldsymbol{A}\boldsymbol{V}^*\| = \|\boldsymbol{A}\| \qquad \text{and} \qquad \|\boldsymbol{U}\boldsymbol{A}\boldsymbol{V}^*\|_{\mathrm{F}} = \|\boldsymbol{A}\|_{\mathrm{F}}$$

for every matrix $\boldsymbol{A}$ and all orthonormal matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ with the right dimensions.

3.2. Standard matrix factorizations. This section defines three basic matrix decompositions. Throughout, $\boldsymbol{A}$ is an $m \times n$ matrix with entries a_{ij}. We use the notation of [61] to denote submatrices. If $I = [i_1, i_2, \ldots, i_p]$ and $J = [j_1, j_2, \ldots, j_q]$ are two index vectors, then the associated $p \times q$ submatrix is expressed as

$$\boldsymbol{A}_{(I,J)} = \begin{bmatrix} a_{i_1,j_1} & \cdots & a_{i_1,j_q} \\ \vdots & & \vdots \\ a_{i_p,j_1} & \cdots & a_{i_p,j_q} \end{bmatrix}.$$

For column- and row-submatrices, we use the abbreviations $\boldsymbol{A}_{(:\,,J)} = \boldsymbol{A}_{([1,2,\ldots,m],J)}$ and $\boldsymbol{A}_{(I,:\,)} = \boldsymbol{A}_{(I,[1,2,\ldots,n])}$.

3.2.1. The pivoted QR factorization. Each $m \times n$ matrix $\boldsymbol{A}$ admits a decomposition

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R},$$

where $\boldsymbol{Q}$ is an $m \times \ell$ orthonormal matrix, and $\boldsymbol{R}$ is an $\ell \times n$ *weakly upper-triangular* matrix. That is, there exists a permutation J of the numbers $\{1, 2, \ldots, n\}$ such that $\boldsymbol{R}_{(:\,,J)}$ is upper triangular. Moreover, the diagonal entries of $\boldsymbol{R}_{(:\,,J)}$ are weakly decreasing. See [61, §5.4.1] for details.

3.2.2. The singular value decomposition (SVD). Each $m \times n$ matrix $\boldsymbol{A}$ admits a factorization

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*,$$

where $\boldsymbol{U}$ is an $m \times \ell$ orthonormal matrix, $\boldsymbol{V}$ is an $n \times \ell$ orthonormal matrix, and $\boldsymbol{\Sigma}$ is an $\ell \times \ell$ nonnegative, diagonal matrix

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_\ell \end{bmatrix}.$$

The numbers σ_j are called the *singular values* of $\boldsymbol{A}$. They are arranged in weakly decreasing order:

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_\ell \geq 0.$$

The columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ are called *left singular vectors* and *right singular vectors*, respectively.

Singular values are connected with the approximability of matrices. For each j, the number σ_{j+1} equals the spectral-norm discrepancy between $\boldsymbol{A}$ and an optimal rank-j approximation [99]. That is,

$$\sigma_{j+1} = \min\{\|\boldsymbol{A} - \boldsymbol{B}\| \,:\, \boldsymbol{B} \text{ has rank } j\}. \tag{3.1}$$

In particular, $\sigma_1 = \|\boldsymbol{A}\|$. See [61, §2.5.3 and §5.4.5] for additional details.

3.2.3. The interpolative decomposition (ID). Our final factorization identifies a collection of k columns from a rank-k matrix $\boldsymbol{A}$ that span the range of $\boldsymbol{A}$. To be precise, there exists an index set $J = [j_1, \ldots, j_k]$ such that

$$\boldsymbol{A} = \boldsymbol{A}_{(:\,,J)}\, \boldsymbol{X},$$

where $\boldsymbol{X}$ is a $k \times n$ matrix that contains the $k \times k$ identity matrix $\mathbf{I}_k$. Specifically, $\boldsymbol{X}_{(:\,,J)} = \mathbf{I}_k$. Furthermore, no entry of $\boldsymbol{X}$ has magnitude larger than one.

Computing the ID of a general matrix is NP-hard. Even so, if we relax the restriction on $\boldsymbol{X}$ so that its entries have magnitude bounded by two, then very efficient algorithms are available [29, 68].

3.3. Techniques for computing standard factorizations. This section discusses some established deterministic techniques for computing the factorizations presented in §3.2. The material on pivoted QR and SVD can be located in any major text on numerical linear algebra, such as [61, 131]. References for the ID include [29, 68].

3.3.1. Computing the full decomposition. It is possible to compute the full QR factorization or the full SVD of an $m \times n$ matrix to double-precision accuracy with $\mathrm{O}(mn \min\{m, n\})$ flops. Techniques for computing the SVD are iterative by necessity, but they converge so fast that we can treat them as finite for practical purposes.

3.3.2. Computing partial decompositions. Suppose that an $m \times n$ matrix has numerical rank k, where k is substantially smaller than m and n. In this case, it is possible to produce a structured low-rank decomposition that approximates the matrix well. Sections 4 and 5 describe a set of randomized techniques for obtaining these partial decompositions. This section briefly reviews the classical techniques, which also play a role in developing randomized methods.

To compute a partial QR decomposition, the classical device is the Businger–Golub algorithm, which performs successive orthogonalization with pivoting on the columns of the matrix. We halt the procedure when the Frobenius norm of the remaining columns is less than a computational tolerance ε. Provided that the orthogonality of the basis is maintained scrupulously, the process usually results in a partial factorization

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R} + \boldsymbol{E}, \tag{3.2}$$

where $\boldsymbol{Q}$ is an $m \times k$ orthonormal matrix, $\boldsymbol{R}$ is a $k \times n$ weakly upper-triangular matrix, and $\boldsymbol{E}$ is an error term that satisfies $\|\boldsymbol{E}\|_{\mathrm{F}} \leq \varepsilon$. The computational cost is $\mathrm{O}(kmn)$. In practice, k is typically close to the minimal rank for which this decomposition is possible, but the Businger–Golub algorithm can fail for pathological matrices.

Subsequent research has led to strong rank-revealing QR algorithms that succeed for all matrices. For example, using $\mathrm{O}(kmn)$ flops, the Gu–Eisenstat algorithm produces an QR decomposition of the form (3.2), where

$$\|\boldsymbol{E}\| \leq \sqrt{1 + 4k(n-k)} \cdot \sigma_{k+1}.$$

Recall that σ_{k+1} is the minimal error possible in a rank-k approximation [99]. The Gu–Eisenstat algorithm can also be used to obtain an approximate ID in $\mathrm{O}(kmn)$ flops [29].

To compute an approximate SVD of a general $m \times n$ matrix, the most straightforward technique is to compute the full SVD and truncate it. This procedure is stable and accurate, but it requires $\mathrm{O}(mn \min\{m, n\})$ flops. A more efficient approach is to compute a partial QR factorization and postprocess the factors to obtain a partial SVD using the methods described below in §3.3.3. This scheme takes only $\mathrm{O}(kmn)$ flops. Krylov subspace methods can also compute partial SVDs at a comparable cost of $\mathrm{O}(kmn)$, but they are much less robust.

Note that all the techniques described in this section require extensive random access to the matrix, and they can be very slow when the matrix is stored out-of-core.

3.3.3. Converting from one partial factorization to another. Suppose that we have obtained a partial decomposition of a matrix $\boldsymbol{A}$ by some means:

$$\|\boldsymbol{A} - \boldsymbol{C}\boldsymbol{B}\| \leq \varepsilon,$$

where $\boldsymbol{B}$ and $\boldsymbol{C}$ have rank k. Given this information, we can efficiently compute any of the basic factorizations.

We construct a partial QR factorization using the following three steps:

1. Compute a QR factorization of $\boldsymbol{C}$ so that $\boldsymbol{C} = \boldsymbol{Q}_1\boldsymbol{R}_1$.
2. Form the product $\boldsymbol{D} = \boldsymbol{R}_1\boldsymbol{B}$, and compute a QR factorization: $\boldsymbol{D} = \boldsymbol{Q}_2\boldsymbol{R}$.
3. Form the product $\boldsymbol{Q} = \boldsymbol{Q}_1\boldsymbol{Q}_2$.

The result is an orthonormal matrix $\boldsymbol{Q}$ and a weakly upper-triangular matrix $\boldsymbol{R}$ such that $\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{R}\| \leq \varepsilon$.

An analogous technique yields a partial SVD:

1. Compute a QR factorization of $\boldsymbol{C}$ so that $\boldsymbol{C} = \boldsymbol{Q}_1\boldsymbol{R}_1$.
2. Form the product $\boldsymbol{D} = \boldsymbol{R}_1\boldsymbol{B}$, and compute an SVD: $\boldsymbol{D} = \boldsymbol{U}_2\boldsymbol{\Sigma}\boldsymbol{V}^*$.
3. Form the product $\boldsymbol{U} = \boldsymbol{Q}_1\boldsymbol{U}_2$.

The result is a diagonal matrix $\boldsymbol{\Sigma}$ and orthonormal matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ such that $\|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*\| \leq \varepsilon$.

Converting $\boldsymbol{B}$ and $\boldsymbol{C}$ into a partial ID is a one-step process:

1. Compute J and $\boldsymbol{X}$ such that $\boldsymbol{B} = \boldsymbol{B}_{(:\,,J)}\boldsymbol{X}$.

Then $\boldsymbol{A} \approx \boldsymbol{A}_{(:\,,J)}\boldsymbol{X}$, but the approximation error may deteriorate from the initial estimate. For example, if we use the Gu–Eisenstat algorithm [68] to compute the ID, then $\left\|\boldsymbol{A} - \boldsymbol{A}_{(:\,,J)}\boldsymbol{X}\right\| \leq (1 + \sqrt{1 + 4k(n-k)}) \cdot \varepsilon$. Compare this bound with Lemma 5.1 below.

3.3.4. Krylov-subspace methods. Suppose that the matrix $\boldsymbol{A}$ can be applied rapidly to vectors, as happens when $\boldsymbol{A}$ is sparse or structured. Then Krylov subspace techniques can very effectively and accurately compute partial spectral decompositions. For concreteness, assume that $\boldsymbol{A}$ is square. The concept behind these techniques is to fix a random starting vector $\boldsymbol{\omega}$ and to generate the corresponding *Krylov subspace*

$$\mathcal{V}_q(\boldsymbol{\omega}) = \operatorname{span}\{\boldsymbol{\omega}, \boldsymbol{A}\boldsymbol{\omega}, \boldsymbol{A}^2\boldsymbol{\omega}, \dots, \boldsymbol{A}^{q-1}\boldsymbol{\omega}\}.$$

The algorithms essentially restrict the matrix to the subspace and perform a spectral decomposition of the reduced matrix.

The execution time of Krylov subspace techniques typically satisfies

$$T_{\text{Krylov}} \sim k\,T_{\text{mult}} + k^2(m+n), \tag{3.3}$$

where T_{mult} is the cost of a matrix–vector multiplication. In practice, the performance of Krylov subspace methods depends heavily on the structure of the singular spectrum of the matrix.

Part II: Algorithms

This part of the paper, §§4–7, provides detailed descriptions of randomized algorithms for constructing low-rank approximations to matrices. As discussed in §1.2, we split the problem into two stages. In Stage A, we construct a subspace that captures the action of the input matrix. In Stage B, we use this subspace to help us obtain an approximate factorization of the matrix.

Section 4 develops randomized methods for completing Stage A, and §5 describes deterministic methods for Stage B. Section 6 compares the computational costs of the resulting two-stage algorithm with the classical approaches outlined in §3. Finally, §7 illustrates the performance of the randomized schemes via numerical examples.

4. Stage A: Randomized schemes for approximating the range. This section outlines techniques for constructing a subspace that captures most of the action of a matrix. We begin with a recapitulation of the proto-algorithm that we

introduced in §1.3. We discuss how it can be implemented in practice (§4.1) and then consider the question of how many random samples to acquire (§4.2). Afterward, we present several ways in which the basic scheme can be improved. Sections 4.3 and 4.4 explain how to address the situation where the numerical rank of the input matrix is not known in advance. Section 4.5 shows how to modify the scheme to improve its accuracy when the singular spectrum of the input matrix decays slowly. Finally, §4.6 describes how the scheme can be accelerated by using a structured random matrix.

4.1. The proto-algorithm revisited. The most natural way to implement the proto-algorithm from §1.3 is to draw a random test matrix $\boldsymbol{\Omega}$ from the standard Gaussian distribution. That is, each entry of $\boldsymbol{\Omega}$ is an independent Gaussian random variable with mean zero and variance one. For reference, we formulate the resulting scheme as Algorithm 4.1.

ALGORITHM 4.1

Given an $m \times n$ matrix $\boldsymbol{A}$, and an integer ℓ, this scheme computes an $m \times \ell$ orthonormal matrix $\boldsymbol{Q}$ whose range approximates the range of $\boldsymbol{A}$.

1 Draw an $n \times \ell$ Gaussian random matrix $\boldsymbol{\Omega}$.
2 Form the $m \times \ell$ matrix $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{\Omega}$.
3 Construct an $m \times \ell$ matrix $\boldsymbol{Q}$ whose columns form an orthonormal basis for the range of $\boldsymbol{Y}$.

The number T_{basic} of flops required by Algorithm 4.1 satisfies

$$T_{\text{basic}} \sim \ell n\, T_{\text{rand}} + \ell\, T_{\text{mult}} + \ell^2 n \tag{4.1}$$

where T_{rand} is the cost of generating a Gaussian random number and T_{mult} is the cost of multiplying $\boldsymbol{A}$ by a vector. The three terms in (4.1) correspond directly with the three steps of Algorithm 4.1.

Empirically, we have found that the performance of Algorithm 4.1 depends very little on the quality of the random number generator used in Step 1.

The actual cost of Step 2 depends substantially on the matrix $\boldsymbol{A}$ and the computational environment that we are working in. The estimate (4.1) suggests that Algorithm 4.1 is especially efficient when the matrix–vector product $\boldsymbol{x} \mapsto \boldsymbol{A}\boldsymbol{x}$ can be evaluated rapidly. In particular, the scheme is appropriate for approximating sparse or structured matrices. Turn to §6 for more details.

The most important implementation issue arises when performing the basis calculation in Step 3. Typically, the columns of the sample matrix $\boldsymbol{Y}$ are almost linearly dependent, so it is imperative to use stable methods for performing the orthonormalization. We have found that the Gram–Schmidt procedure, augmented with the *double orthogonalization* described in [12], is both convenient and reliable. Methods based on Householder reflectors or Givens rotations also work very well. Note that very little is gained by pivoting because the columns of the random matrix $\boldsymbol{Y}$ are independent samples drawn from the same distribution.

4.2. The number of samples required. The goal of Algorithm 4.1 is to produce an orthonormal matrix $\boldsymbol{Q}$ with few columns that achieves

$$\|(\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^*)\boldsymbol{A}\| \leq \varepsilon, \tag{4.2}$$

where ε is a specified computational tolerance. The number of columns ℓ that the algorithm needs to reach this threshold is usually slightly larger than the minimal rank k of the smallest basis that verifies (4.2). We call the discrepancy $p = \ell - k$ the *oversampling parameter*. The size of the oversampling parameter depends on several factors:

The matrix dimensions. Very large matrices may require more oversampling.

The singular spectrum. The more rapid the decay of the singular values, the less oversampling is needed. In the extreme case that the matrix has exact rank k, it is not necessary to oversample.

The random test matrix. Gaussian matrices succeed with very little oversampling, but they can be difficult to generate precisely. The structured random matrices discussed in §4.6 may require substantial oversampling, but they still yield computational gains in certain settings.

The theoretical results in Part III provide detailed information about how the behavior of randomized schemes depends on these factors. For the moment, we limit ourselves to some general remarks on implementation issues.

For Gaussian test matrices, it is adequate to choose the oversampling parameter to be a small constant, such as $p = 5$ or $p = 10$. There is rarely any advantage to select $p > k$. This observation from demonstrates that a Gaussian test matrix results in a negligible amount of extra computation [94].

In practice, the target rank k is rarely known in advance. Randomized algorithms are usually implemented to increase the number of samples adaptively until the desired tolerance is met. In other words, the user never *chooses* the oversampling parameter. Theoretical results that bound the amount of oversampling are valuable primarily as aids for designing algorithms. We develop an adaptive approach in §§4.3–4.4.

The computational bottleneck in Algorithm 4.1 is usually the formation of the product $\boldsymbol{A\Omega}$. As a result, it often pays to draw a larger number ℓ of samples than necessary because the user can minimize the cost of the matrix multiplication with tools such as blocking of operations, high-level linear algebra subroutines, parallel processors, etc. This approach may lead to an ill-conditioned sample matrix $\boldsymbol{Y}$, but the orthogonalization in Step 3 of Algorithm 4.1 can easily identify the numerical rank of the sample matrix and ignore the excess samples. Furthermore, Stage B of the matrix approximation process succeeds even when the basis matrix $\boldsymbol{Q}$ has a larger dimension than necessary.

4.3. A posteriori error estimation. Algorithm 4.1 is designed for solving the fixed-rank problem, where the target rank of the input matrix is specified in advance. To handle the fixed-precision problem, where the parameter is the computational tolerance, we need a scheme for estimating how well a putative basis matrix $\boldsymbol{Q}$ captures the action of the matrix $\boldsymbol{A}$. To do so, we develop a probabilistic error estimator. These methods are inspired by work of Dixon [41]; our treatment follows [90, 135].

The exact approximation error is $\|(\mathbf{I} - \boldsymbol{QQ}^*)\boldsymbol{A}\|$. It is intuitively plausible that we can obtain some information about this quantity by computing $\|(\mathbf{I} - \boldsymbol{QQ}^*)\boldsymbol{A\omega}\|$, where $\boldsymbol{\omega}$ is a standard Gaussian vector. This notion leads to the following method. Draw a sequence $\{\boldsymbol{\omega}^{(i)} : i = 1, 2, \dots, r\}$ of standard Gaussian vectors, where r is a small integer that balances computational cost and reliability. Then

$$\|(\mathbf{I} - \boldsymbol{QQ}^*)\boldsymbol{A}\| \le 10\sqrt{\frac{2}{\pi}} \max_{i=1,\dots,r} \left\|(\mathbf{I} - \boldsymbol{QQ}^*)\boldsymbol{A\omega}^{(i)}\right\| \tag{4.3}$$

except with probability 10^{-r}. This statement follows by setting $\boldsymbol{B} = (\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^*)\boldsymbol{A}$ and $\alpha = 1/10$ in the following lemma, whose proof appears in [135, §3.4].

LEMMA 4.1. *Let $\boldsymbol{B}$ be a real $m \times n$ matrix. Fix a positive integer r and a real number $\alpha \in (0, 1)$. Draw an independent family $\{\boldsymbol{\omega}^{(i)} : i = 1, 2, \dots, r\}$ of standard Gaussian vectors. Then*

$$\|\boldsymbol{B}\| \le \frac{1}{\alpha}\sqrt{\frac{2}{\pi}} \max_{i=1,\dots,r} \left\|\boldsymbol{B}\boldsymbol{\omega}^{(i)}\right\|$$

except with probability α^r.

The critical point is that the error estimate (4.3) requires only a small number of matrix–vector products, hence is computationally cheap. Therefore, we can make a lowball guess for the numerical rank of $\boldsymbol{A}$ and add more samples if the error estimate is too large. The asymptotic cost of Algorithm 4.1 is preserved if we double our guess for the rank at each step. For example, we can start with 32 samples, compute another 32, then another 64, etc.

REMARK 4.1. The estimate (4.3) is actually somewhat crude. We can obtain a better estimate at a similar computational cost by initializing a power iteration with a random vector and repeating the process several times [90].

4.4. Error estimation (almost) for free. The error estimate described in §4.3 can be combined with any method for constructing an approximate basis for the range of a matrix. In this section, we explain how the error estimator can be incorporated into Algorithm 4.1 at almost no additional cost.

To be precise, let us suppose that $\boldsymbol{A}$ is an $m \times n$ matrix and ε is a computational tolerance. We seek an integer ℓ and an $m \times \ell$ orthonormal matrix $\boldsymbol{Q}^{(\ell)}$ such that

$$\left\|\left(\mathbf{I} - \boldsymbol{Q}^{(\ell)}(\boldsymbol{Q}^{(\ell)})^*\right)\boldsymbol{A}\right\| \le \varepsilon. \tag{4.4}$$

The size ℓ of the basis will typically be slightly larger than the size k of the smallest basis that achieves this error.

The basic observation behind the adaptive scheme is that we can generate the basis in Step 3 of Algorithm 4.1 incrementally. Starting with an empty basis matrix $\boldsymbol{Q}^{(0)}$, the following scheme generates an orthonormal matrix whose range captures the action of $\boldsymbol{A}$:

for $i = 1, 2, 3, \dots$
  Draw an $n \times 1$ Gaussian random vector $\boldsymbol{\omega}^{(i)}$, and set $\boldsymbol{y}^{(i)} = \boldsymbol{A}\boldsymbol{\omega}^{(i)}$.
  Compute $\hat{\boldsymbol{q}}^{(i)} = \left(\mathbf{I} - \boldsymbol{Q}^{(i-1)}(\boldsymbol{Q}^{(i-1)})^*\right)\boldsymbol{y}^{(i)}$.
  Normalize $\boldsymbol{q}^{(i)} = \hat{\boldsymbol{q}}^{(i)}/\left\|\hat{\boldsymbol{q}}^{(i)}\right\|$, and form $\boldsymbol{Q}^{(i)} = [\boldsymbol{Q}^{(i-1)}\ \boldsymbol{q}^{(i)}]$.
end for

How do we know when we have reached a basis $\boldsymbol{Q}^{(\ell)}$ that verifies (4.4)? The answer becomes apparent once we observe that the vectors $\hat{\boldsymbol{q}}^{(i)}$ are precisely the vectors that appear in the error bound (4.3). The resulting rule is that we break the loop once we observe r consecutive vectors $\hat{\boldsymbol{q}}^{(i)}$ whose norms are smaller than $\varepsilon/(10\sqrt{2/\pi})$.

A potential complication is that the vectors $\hat{\boldsymbol{q}}^{(i)}$ become small as the basis starts to capture most of the action of $\boldsymbol{A}$. In finite-precision arithmetic, their direction is extremely unreliable. To address this problem, we simply reproject the normalized vector $\boldsymbol{q}^{(i)}$ onto $\text{range}(\boldsymbol{Q}^{(i-1)})^{\perp}$.

A formal description of the resulting algorithm appears as Algorithm 4.2. Its CPU time requirement is essentially identical with that of Algorithm 4.1. Note that the algorithm computes the last few samples purely to obtain the error estimate; this apparent extra cost is offset by the fact that Algorithm 4.1 always includes an oversampling factor. The failure probability stated for this scheme is pessimistic because it is derived from a simple argument using the union bound. In practice, the error estimator is reliable in a range of circumstances when we take $r = 10$.

Algorithm 4.2

Given an $m \times n$ matrix $\boldsymbol{A}$, a tolerance ε, and an integer r, the following scheme computes an orthonormal matrix $\boldsymbol{Q}$ such that (1.3) *holds with probability at least* $1 - \min\{m, n\}10^{-r}$.

Draw standard Gaussian vectors $\boldsymbol{\omega}^{(1)}, \dots, \boldsymbol{\omega}^{(r)}$ of length n.
For $i = 1, 2, \dots, r$, compute $\boldsymbol{y}^{(i)} = \boldsymbol{A}\boldsymbol{\omega}^{(i)}$.
$j = 0$.
$\boldsymbol{Q}^{(0)} = [\,]$, the $m \times 0$ empty matrix.
while $\max\left\{ \left\|\boldsymbol{y}^{(j+1)}\right\|, \left\|\boldsymbol{y}^{(j+2)}\right\|, \dots, \left\|\boldsymbol{y}^{(j+r)}\right\| \right\} > \varepsilon/(10\sqrt{2/\pi})$,
    $j = j + 1$.
    Overwrite $\boldsymbol{y}^{(j)}$ by $\left(\mathbf{I} - \boldsymbol{Q}^{(j-1)}(\boldsymbol{Q}^{(j-1)})^*\right)\boldsymbol{y}^{(j)}$.
    $\boldsymbol{q}^{(j)} = \boldsymbol{y}^{(j)} / \left\|\boldsymbol{y}^{(j)}\right\|$.
    $\boldsymbol{Q}^{(j)} = [\boldsymbol{Q}^{(j-1)}\ \boldsymbol{q}^{(j)}]$.
    Draw a standard Gaussian vector $\boldsymbol{\omega}^{(j+r)}$ of length n.
    $\boldsymbol{y}^{(j+r)} = \left(\mathbf{I} - \boldsymbol{Q}^{(j)}(\boldsymbol{Q}^{(j)})^*\right)\boldsymbol{A}\boldsymbol{\omega}^{(j+r)}$.
    for $i = (j+1), (j+2), \dots, (j+r-1)$,
        Overwrite $\boldsymbol{y}^{(i)}$ by $\boldsymbol{y}^{(i)} - \boldsymbol{q}^{(j)}\left\langle \boldsymbol{q}^{(j)},\ \boldsymbol{y}^{(i)} \right\rangle$.
    end for
end while
$\boldsymbol{Q} = \boldsymbol{Q}^{(j)}$.

Remark 4.2. The calculations in Algorithm 4.2 can be organized so that each iteration processes a block of samples simultaneously. This revision can lead to dramatic improvements in speed because it allows us to exploit BLAS3 (i.e., higher-level linear algebra subroutines) or parallel processors. Although blocking can lead to the generation of unnecessary samples, this outcome is generally harmless, as noted in §4.2.

4.5. A modified scheme for matrices whose singular values decay slowly. The techniques described in §4.1 and §4.4 work well for matrices whose singular values exhibit some decay, but they may result in a poor basis when the input matrix has a flat singular spectrum or when the input matrix is very large. Our theoretical analysis in §10 quantifies these tradeoffs precisely.

In this section, we develop a remedy suggested in [67, 111] that is related to earlier work of [113]. This approach incorporates Krylov-subspace ideas to produce a basis with near-optimal accuracy. The intuition is that the small singular vectors interfere with the calculation, so we reduce their weight relative to the dominant singular vectors by means of a power iteration.

More precisely, we can apply the randomized sampling scheme to the matrix

$\boldsymbol{B} = (\boldsymbol{A}\boldsymbol{A}^*)^q \boldsymbol{A}$, where q is a small integer. The matrix $\boldsymbol{B}$ has the same singular vectors as the input matrix $\boldsymbol{A}$, but its singular values decay much more quickly:

$$\sigma_j(\boldsymbol{B}) = \sigma_j(\boldsymbol{A})^{2q+1}, \quad j = 1, 2, 3, \dots.$$

This approach requires $2q+1$ times as many matrix–vector multiplies as Algorithm 4.1, but it is far more accurate. A good heuristic is that, when the original scheme produces a basis whose approximation error is within a factor C of the optimum, the power scheme produces an approximation error within $C^{1/(2q+1)}$ of the optimum. In other words, the power iteration drives the approximation gap to one exponentially fast. See Theorem 9.2 and §10.4 for the details.

In practice, we must adapt this idea to account for the fact that calculations are performed in finite-precision arithmetic. In this case, it is advantageous to use the sample matrix formed when we retain the results of intermediate calculations:

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{A}\boldsymbol{\Omega} & \big| & (\boldsymbol{A}\boldsymbol{A}^*)\boldsymbol{A}\boldsymbol{\Omega} & \big| & \dots & \big| & (\boldsymbol{A}\boldsymbol{A}^*)^q\boldsymbol{A}\boldsymbol{\Omega} \end{bmatrix}. \tag{4.5}$$

The resulting scheme is summarized as Algorithm 4.3. Corollary 9.3 and [111] provide additional analysis.

ALGORITHM 4.3

Given an $m \times n$ matrix $\boldsymbol{A}$ and numbers ℓ, q, this algorithm computes an $m \times \ell$ orthonormal matrix $\boldsymbol{Q}$ whose range approximates the range of $\boldsymbol{A}$.

1 Draw an $n \times \ell$ standard Gaussian matrix $\boldsymbol{\Omega}$.

2 Compute the $n \times (q+1)\ell$ sample matrix $\boldsymbol{W} = \big[\boldsymbol{Y}^{(0)}, \boldsymbol{Y}^{(1)}, \dots, \boldsymbol{Y}^{(q)}\big]$, where the matrices $\boldsymbol{Y}^{(i)}$ are defined by induction: $\boldsymbol{Y}^{(0)} = \boldsymbol{A}\boldsymbol{\Omega}$, and $\boldsymbol{Y}^{(i)} = \boldsymbol{A}\boldsymbol{A}^*\boldsymbol{Y}^{(i-1)}$ for $i = 1, 2, \dots, q$.

3 Form the $n \times \ell$ matrix $\boldsymbol{Q}$ by taking the first ℓ steps of a rank-revealing QR factorization of $\boldsymbol{Z}$.

Algorithm 4.3 targets the fixed-rank problem. To address the fixed-precision problem, we can incorporate the error estimators described in §4.3 to obtain an adaptive scheme analogous with Algorithm 4.2. In situations where it is critical to achieve near-optimal approximation errors, one can increase the oversampling beyond our standard recommendation $\ell = k + 5$ all the way to $\ell = 2k$ without changing the scaling of the asymptotic computational cost. A supporting analysis appears in Corollary 10.10.

4.6. An accelerated technique for general dense matrices. This section describes a set of techniques that allow us to compute an approximate rank-ℓ factorization of a general dense $m \times n$ matrix in roughly $\mathrm{O}(mn \log(\ell))$ flops, in contrast to the asymptotic cost $\mathrm{O}(mn\ell)$ required by earlier methods. We can tailor this scheme for the real or complex case, but we focus on the conceptually simpler complex case. These algorithms were introduced in [135]; similar techniques were proposed in [118].

The first step toward this accelerated technique is to observe that the bottleneck in Algorithm 4.1 is the computation of the matrix product $\boldsymbol{A}\boldsymbol{\Omega}$. When the test matrix $\boldsymbol{\Omega}$ is standard Gaussian, the cost of this multiplication is $\mathrm{O}(mn\ell)$, the same as a rank-revealing QR algorithm [68]. The key idea is to use a *structured* random matrix that allows us to compute the product in $\mathrm{O}(mn \log(\ell))$ flops.

The *subsampled random Fourier transform*, or SRFT, is perhaps the simplest example of a structured random matrix that meets our goals. An SRFT is an $n \times \ell$ matrix of the form

$$\boldsymbol{\Omega} = \sqrt{\frac{n}{\ell}}\, \boldsymbol{D}\boldsymbol{F}\boldsymbol{R}, \tag{4.6}$$

where

- $\boldsymbol{D}$ is an $n \times n$ diagonal matrix whose entries are independent random variables uniformly distributed on the complex unit circle,
- $\boldsymbol{F}$ is the $n \times n$ unitary discrete Fourier transform, whose entries take the values $f_{pq} = n^{-1/2}\, \mathrm{e}^{-2\pi \mathrm{i}(p-1)(q-1)/n}$ for $p, q = 1, 2, \ldots, n$, and
- $\boldsymbol{R}$ is an $n \times \ell$ matrix that samples ℓ coordinates from n uniformly at random, i.e., its ℓ columns are drawn randomly without replacement from the columns of the $n \times n$ identity matrix.

When $\boldsymbol{\Omega}$ is defined by (4.6), we can compute the sample matrix $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{\Omega}$ using $\mathrm{O}(mn \log(\ell))$ flops via a subsampled FFT [135]. Then we form the basis $\boldsymbol{Q}$ by orthonormalizing the columns of $\boldsymbol{Y}$, as documented in §4.1. The total number T_{struct} of flops required by this procedure is

$$T_{\mathrm{struct}} \sim mn \log(\ell) + \ell^2 n \tag{4.7}$$

This scheme is analogous with Algorithm 4.1. Note that if ℓ is substantially larger than the numerical rank k of the input matrix, we can perform the orthogonalization with $\mathrm{O}(k\ell n)$ flops because the columns of the sample matrix are almost linearly dependent.

The test matrix (4.6) is just one choice among many possibilities. Other suggestions that appear in the literature include subsampled Hadamard transforms, chains of Givens rotations acting on randomly chosen coordinates, and many more. See [88] and its bibliography. Empirically, we have found that the transform summarized in Remark 4.5 below performs very well in a variety of environments [112].

At this point, it is not well understood how to quantify and compare the behavior of structured random transforms. One reason for this uncertainty is that it has been difficult to analyze the amount of oversampling that various transforms require. Section 11 establishes that the random matrix (4.6) can be used to identify a near-optimal basis for a rank-k matrix using $\ell \sim (k + \log(n)) \log(k)$ samples. In practice, the transforms (4.6) and (4.8) typically require no more over-sampling than do a Gaussian random matrix. (For a numerical example, see §7.5.) In consequence, setting $\ell = k + 10$ or $\ell = k + 20$ is typically more than adequate. Further research on these questions would be valuable.

REMARK 4.3. The structured random matrices discussed in this section do not adapt readily to the fixed-precision problem, where the computational tolerance is specified, because the samples from the range are usually computed in bulk. Fortunately, these schemes are sufficiently inexpensive that we can progressively increase the number of samples computed starting with $\ell = 32$, say, and then proceeding to $\ell = 64, 128, 256, \ldots$ until we achieve the desired tolerance.

REMARK 4.4. When using the SRFT (4.6) for matrix approximation, we have a choice whether to use a subsampled FFT or a full FFT. The complete FFT is so inexpensive that it often pays to construct an extended sample matrix $\boldsymbol{Y}_{\mathrm{large}} = \boldsymbol{A}\boldsymbol{D}\boldsymbol{F}$

and then generate the actual samples by drawing columns at random from $\boldsymbol{Y}_{\text{large}}$ and rescaling as needed. The asymptotic cost increases very slightly to $\mathrm{O}(mn \log(n))$ flops, but the full FFT is actually faster for moderate problem sizes because the constant suppressed by the big-O notation is so small. Adaptive rank determination is easy because we just examine extra samples as needed.

REMARK 4.5. Among the structured random matrices that we have tried, one of the strongest candidates involves sequences of random Givens rotations [112]. This matrix takes the form

$$\boldsymbol{\Omega} = \boldsymbol{D}'' \, \boldsymbol{\Theta}' \, \boldsymbol{D}' \, \boldsymbol{\Theta} \, \boldsymbol{D} \, \boldsymbol{F} \, \boldsymbol{R}, \tag{4.8}$$

where the prime symbol $'$ indicates an independent realization of a random matrix. The matrices $\boldsymbol{R}$, $\boldsymbol{F}$, and $\boldsymbol{D}$ are defined after (4.6). The matrix $\boldsymbol{\Theta}$ is a chain of random Givens rotations:

$$\boldsymbol{\Theta} = \boldsymbol{\Pi} \, \boldsymbol{G}(1, 2; \theta_1) \, \boldsymbol{G}(2, 3; \theta_2) \cdots \boldsymbol{G}(n-1, n; \theta_{n-1})$$

where $\boldsymbol{\Pi}$ is a random $n \times n$ permutation matrix; where $\theta_1, \dots, \theta_{n-1}$ are independent random variables uniformly distributed on the interval $[0, 2\pi]$; and where $\boldsymbol{G}(i, j; \theta)$ denotes a rotation on $\mathbb{R}^n$ by the angle θ in the (i, j) coordinate plane [61, §5.1.8].

5. Stage B: Construction of standard factorizations. The algorithms for Stage A described in §4 produce an orthonormal matrix $\boldsymbol{Q}$ whose range captures the action of an input matrix $\boldsymbol{A}$:

$$\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^* \boldsymbol{A}\| \leq \varepsilon, \tag{5.1}$$

where ε is a computational tolerance. This section describes methods for approximating standard factorizations of $\boldsymbol{A}$ using the information in the basis $\boldsymbol{Q}$.

To accomplish this task, we pursue the idea from §3.3.3 that any low-rank factorization $\boldsymbol{A} \approx \boldsymbol{C}\boldsymbol{B}$ can be manipulated to produce a standard decomposition. When the bound (5.1) holds, the low-rank factors are simply $\boldsymbol{C} = \boldsymbol{Q}$ and $\boldsymbol{B} = \boldsymbol{Q}^* \boldsymbol{A}$. The simplest scheme (§5.1) computes the factor $\boldsymbol{B}$ directly with a matrix product to ensure a minimal error in the final approximation. An alternative approach (§5.2) constructs $\boldsymbol{B}$ using only the information in the basis $\boldsymbol{Q}$. The second method can be faster but typically results in larger errors. Both schemes can be streamlined for an Hermitian input matrix (§5.3). Finally, we develop single-pass algorithms that exploit other information generated in Stage A to avoid revisiting the input matrix (§5.4).

Throughout this section, $\boldsymbol{A}$ denotes an $m \times n$ matrix, and $\boldsymbol{Q}$ is an $m \times k$ orthonormal matrix that verifies (5.1). For purposes of exposition, we concentrate on methods for constructing the partial SVD.

5.1. Factorizations based on forming $\boldsymbol{Q}^* \boldsymbol{A}$ directly. The relation (5.1) implies that $\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{B}\| \leq \varepsilon$, where $\boldsymbol{B} = \boldsymbol{Q}^* \boldsymbol{A}$. Once we have computed $\boldsymbol{B}$, we can produce any standard factorization using the methods of §3.3.3. Algorithm 5.1 illustrates how to build an approximate SVD.

The factors produced by Algorithm 5.1 satisfy

$$\|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*\| \leq \varepsilon. \tag{5.2}$$

Therefore, the approximation error does not degrade.

ALGORITHM 5.1

Given matrices $\boldsymbol{A}$ *and* $\boldsymbol{Q}$ *such that* (5.1) *holds, this procedure computes an approximate SVD* $\boldsymbol{A} \approx \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$.

1 Form the matrix $\boldsymbol{B} = \boldsymbol{Q}^*\boldsymbol{A}$.
2 Compute an SVD of the small matrix: $\boldsymbol{B} = \widehat{\boldsymbol{U}}\boldsymbol{\Sigma}\boldsymbol{V}^*$.
3 Form the product $\boldsymbol{U} = \boldsymbol{Q}\widehat{\boldsymbol{U}}$.

The cost of Algorithm 5.1 is generally dominated by the cost of the product $\boldsymbol{Q}^*\boldsymbol{A}$ in Step 1, which takes $\mathrm{O}(kmn)$ flops for a general dense matrix. Note that this scheme is particularly well suited to environments where we have a fast method for computing the matrix–vector product $\boldsymbol{x} \mapsto \boldsymbol{A}^*\boldsymbol{x}$, for example when $\boldsymbol{A}$ is sparse or structured. This approach retains a strong advantage over Krylov-subspace methods and rank-revealing QR because Step 1 can be accelerated using BLAS3, parallel processors, and so forth. Steps 2 and 3 require $\mathrm{O}(k^2n)$ and $\mathrm{O}(k^2m)$ flops respectively.

5.2. Postprocessing via row extraction. Given a matrix $\boldsymbol{Q}$ such that (5.1) holds, we can obtain a rank-k factorization

$$\boldsymbol{A} \approx \boldsymbol{X}\boldsymbol{B}, \tag{5.3}$$

where $\boldsymbol{B}$ is a $k \times n$ matrix consisting of k rows extracted from $\boldsymbol{A}$. The approximation (5.3) can be produced without computing any matrix–matrix products, which makes this approach to postprocessing very fast. The drawback comes because the error $\|\boldsymbol{A} - \boldsymbol{X}\boldsymbol{B}\|$ is usually larger than the initial error $\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\|$, especially when the dimensions of $\boldsymbol{A}$ are large. See Remark 5.2 for more discussion.

To obtain the factorization (5.3), we simply construct the interpolative decomposition (§3.2.3) of the matrix $\boldsymbol{Q}$:

$$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{Q}_{(J,:)}. \tag{5.4}$$

The index vector J marks k rows of $\boldsymbol{Q}$ that span the row space of $\boldsymbol{Q}$, and $\boldsymbol{X}$ is an $m \times k$ matrix whose entries are bounded in magnitude by 2 and contains the $k \times k$ identity as a submatrix: $\boldsymbol{X}_{(J,:)} = \mathbf{I}_k$. Combining (5.4) and (5.1), we reach

$$\boldsymbol{A} \approx \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A} = \boldsymbol{X}\boldsymbol{Q}_{(J,:)}\boldsymbol{Q}^*\boldsymbol{A}. \tag{5.5}$$

Since $\boldsymbol{X}_{(J,:)} = \mathbf{I}_k$, equation (5.5) implies that $\boldsymbol{A}_{(J,:)} \approx \boldsymbol{Q}_{(J,:)}\boldsymbol{Q}^*\boldsymbol{A}$. Therefore, (5.3) follows when we put $\boldsymbol{B} = \boldsymbol{A}_{(J,:)}$.

Provided with the factorization (5.3), we can obtain any standard factorization using the techniques of §3.3.3. Algorithm 5.2 illustrates an SVD calculation. This procedure requires $\mathrm{O}(k^2(m+n))$ flops. The following lemma guarantees the accuracy of the computed factors.

LEMMA 5.1. *Let* $\boldsymbol{A}$ *be an* $m \times n$ *matrix, and let* $\boldsymbol{Q}$ *be an* $m \times k$ *matrix that satisfy* (5.1). *Suppose that* $\boldsymbol{U}$, $\boldsymbol{\Sigma}$, *and* $\boldsymbol{V}$ *are the matrices constructed by Algorithm 5.2. Then*

$$\|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*\| \leq \left[1 + \sqrt{1 + 4k(n-k)}\right]\varepsilon. \tag{5.6}$$

ALGORITHM 5.2

Given matrices $\boldsymbol{A}$ *and* $\boldsymbol{Q}$ *such that* (5.1) *holds, this algorithm computes an approximate SVD* $\boldsymbol{A} \approx \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$.

1 Compute an ID $\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{Q}_{(J,:)}$.
2 Extract $\boldsymbol{A}_{(J,:)}$, and compute a QR factorization $\boldsymbol{A}_{(J,:)} = \boldsymbol{R}^*\widehat{\boldsymbol{V}}^*$.
3 Form the product $\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{R}^*$.
4 Compute an SVD $\boldsymbol{Z} = \boldsymbol{U}\boldsymbol{\Sigma}\widetilde{\boldsymbol{V}}^*$.
5 Form the product $\boldsymbol{V} = \widehat{\boldsymbol{V}}\widetilde{\boldsymbol{V}}$.

Proof. The factors $\boldsymbol{U}$, $\boldsymbol{\Sigma}$, $\boldsymbol{V}$ constructed by the algorithm satisfy

$$\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^* = \boldsymbol{U}\boldsymbol{\Sigma}\widetilde{\boldsymbol{V}}^*\widehat{\boldsymbol{V}}^* = \boldsymbol{Z}\widehat{\boldsymbol{V}}^* = \boldsymbol{X}\boldsymbol{R}^*\widehat{\boldsymbol{V}}^* = \boldsymbol{X}\boldsymbol{A}_{(J,:)}.$$

Define

$$\widetilde{\boldsymbol{A}} = \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}. \tag{5.7}$$

Since $\widetilde{\boldsymbol{A}} = \boldsymbol{X}\boldsymbol{Q}_{(J,:)}\boldsymbol{Q}^*\boldsymbol{A}$ and since $\boldsymbol{X}_{(J,:)} = \mathbf{I}_k$, it must be that $\widetilde{\boldsymbol{A}}_{(J,:)} = \boldsymbol{Q}_{(J,:)}\boldsymbol{Q}^*\boldsymbol{A}$. Consequently,

$$\widetilde{\boldsymbol{A}} = \boldsymbol{X}\widetilde{\boldsymbol{A}}_{(J,:)}.$$

We have the chain of relations

$$\begin{aligned}
\|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*\| &= \big\|\boldsymbol{A} - \boldsymbol{X}\boldsymbol{A}_{(J,:)}\big\| \\
&= \big\|\big(\boldsymbol{A} - \boldsymbol{X}\widetilde{\boldsymbol{A}}_{(J,:)}\big) + \big(\boldsymbol{X}\widetilde{\boldsymbol{A}}_{(J,:)} - \boldsymbol{X}\boldsymbol{A}_{(J,:)}\big)\big\| \\
&\le \big\|\boldsymbol{A} - \widetilde{\boldsymbol{A}}\big\| + \big\|\boldsymbol{X}\widetilde{\boldsymbol{A}}_{(J,:)} - \boldsymbol{X}\boldsymbol{A}_{(J,:)}\big\| \\
&\le \big\|\boldsymbol{A} - \widetilde{\boldsymbol{A}}\big\| + \|\boldsymbol{X}\|\,\big\|\boldsymbol{A}_{(J,:)} - \widetilde{\boldsymbol{A}}_{(J,:)}\big\|.
\end{aligned} \tag{5.8}$$

Inequality (5.1) ensures that $\big\|\boldsymbol{A} - \widetilde{\boldsymbol{A}}\big\| \le \varepsilon$. Since $\boldsymbol{A}_{(J,:)} - \widetilde{\boldsymbol{A}}_{(J,:)}$ is a submatrix of $\boldsymbol{A} - \widetilde{\boldsymbol{A}}$, we must also have $\big\|\boldsymbol{A}_{(J,:)} - \widetilde{\boldsymbol{A}}_{(J,:)}\big\| \le \varepsilon$. Thus, (5.8) reduces to

$$\|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*\| \le (1 + \|\boldsymbol{X}\|)\,\varepsilon. \tag{5.9}$$

The bound (5.6) follows from (5.9) after we observe that $\boldsymbol{X}$ contains a $k \times k$ identity matrix and that the entries of the remaining $(n-k) \times k$ submatrix are bounded in magnitude by two. □

REMARK 5.1. To maintain a unified presentation, we have formulated all the postprocessing techniques so they take an orthonormal matrix $\boldsymbol{Q}$ as input. Recall that, in Stage A of our framework, we construct the matrix $\boldsymbol{Q}$ by orthonormalizing the columns of the sample matrix $\boldsymbol{Y}$. With finite-precision arithmetic, it is preferable to adapt Algorithm 5.2 to start directly from the sample matrix $\boldsymbol{Y}$. To be precise, we modify Step 1 to compute $\boldsymbol{X}$ and J so that $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{Y}_{(J,:)}$. This revision is recommended even when $\boldsymbol{Q}$ is available from the adaptive rank determination of Algorithm 4.2.

REMARK 5.2. As the inequality (5.6) suggests, the factorization produced by Algorithm 5.2 is potentially less accurate than the basis that it uses as input. This

loss of accuracy is problematic when ε is not so small or when kn is large. In such cases, it is important to use Algorithm 5.1, which is more costly but does not amplify the error, as shown in (5.2).

5.3. Postprocessing of an Hermitian matrix. When $\boldsymbol{A}$ is Hermitian, the postprocessing becomes particularly elegant. In this case, the columns of $\boldsymbol{Q}$ form a good basis for both the column space *and* the row space of $\boldsymbol{A}$ so that we have $\boldsymbol{A} \approx \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{Q}\boldsymbol{Q}^*$. More precisely, when (5.1) is in force, we have

$$\begin{aligned}\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{Q}\boldsymbol{Q}^*\| &= \|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A} + \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{Q}\boldsymbol{Q}^*\| \\ &\le \|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\| + \left\|\boldsymbol{Q}\boldsymbol{Q}^*\left(\boldsymbol{A} - \boldsymbol{A}\boldsymbol{Q}\boldsymbol{Q}^*\right)\right\| \le 2\varepsilon. \quad (5.10)\end{aligned}$$

The last inequality relies on the facts that $\|\boldsymbol{Q}\boldsymbol{Q}^*\| = 1$ and that

$$\|\boldsymbol{A} - \boldsymbol{A}\boldsymbol{Q}\boldsymbol{Q}^*\| = \left\|(\boldsymbol{A} - \boldsymbol{A}\boldsymbol{Q}\boldsymbol{Q}^*)^*\right\| = \|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\|.$$

For Hermitian $\boldsymbol{A}$, it is more common to compute an eigenvalue decomposition than an SVD. We can accomplish this goal by adapting the scheme in §5.1. Form $\boldsymbol{B} = \boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{Q}$, and compute its eigendecomposition $\boldsymbol{B} = \widehat{\boldsymbol{U}}\boldsymbol{\Lambda}\widehat{\boldsymbol{U}}^*$. Finally, construct the product $\boldsymbol{U} = \boldsymbol{Q}\widehat{\boldsymbol{U}}$ to obtain an approximate eigenvalue decomposition that satisfies $\|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^*\| \le 2\varepsilon$. This scheme requires $\mathrm{O}(kn^2)$ flops.

We can also pursue the approach from §5.2, which is faster but less accurate. First, compute an ID of $\boldsymbol{Q}$ to obtain a factorization of the form (5.3). This step results in the symmetric decomposition $\boldsymbol{A} \approx \boldsymbol{X}\boldsymbol{A}_{(J,J)}\boldsymbol{X}^*$, which can be manipulated to obtain an approximate eigenvalue decomposition. The total cost is $\mathrm{O}(k^2n)$ flops.

5.4. Single-pass algorithms. The techniques described in §§5.1–5.3 all require us to revisit the input matrix. This may not be feasible in environments where the matrix is too large to be stored. In this section, we develop a method that requires just one pass over the matrix to construct not only an approximate basis but also a complete factorization. Similar techniques appear in [135] and [32].

For motivation, we begin with the case where $\boldsymbol{A}$ is Hermitian. Let us recall the proto-algorithm from §1.3.3: Draw a random test matrix $\boldsymbol{\Omega}$; form the sample matrix $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{\Omega}$; then construct a basis $\boldsymbol{Q}$ for the range of $\boldsymbol{Y}$. It turns out that the matrices $\boldsymbol{\Omega}$, $\boldsymbol{Y}$, and $\boldsymbol{Q}$ contain all the information we need to approximate $\boldsymbol{A}$.

To see why, imagine that we could form the matrix $\boldsymbol{B} = \boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{Q}$. Postmultiplying this equation by $\boldsymbol{Q}^*\boldsymbol{\Omega}$, we obtain the identity $\boldsymbol{B}\boldsymbol{Q}^*\boldsymbol{\Omega} = \boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{\Omega}$. The relationships $\boldsymbol{A}\boldsymbol{Q}\boldsymbol{Q}^* \approx \boldsymbol{A}$ and $\boldsymbol{A}\boldsymbol{\Omega} = \boldsymbol{Y}$ show that $\boldsymbol{B}$ must satisfy

$$\boldsymbol{B}\boldsymbol{Q}^*\boldsymbol{\Omega} \approx \boldsymbol{Q}^*\boldsymbol{Y}. \quad (5.11)$$

All three matrices $\boldsymbol{\Omega}$, $\boldsymbol{Y}$, and $\boldsymbol{Q}$ are available, so we can solve (5.11) to obtain the matrix $\boldsymbol{B}$. Then the low-rank factorization $\boldsymbol{A} \approx \boldsymbol{Q}\boldsymbol{B}\boldsymbol{Q}^*$ can be converted to an eigenvalue decomposition via familiar techniques. Algorithm 5.3 summarizes this scheme.

The cost of this algorithm is $\mathrm{O}(k^2n)$ flops. The following theorem demonstrates that the accuracy of Algorithm 5.3 hinges on the conditioning of the matrix $\boldsymbol{Q}^*\boldsymbol{\Omega}$.

THEOREM 5.2. *Let $\boldsymbol{A}$ be an $n \times n$ Hermitian matrix, let $\boldsymbol{\Omega}$ be an $n \times \ell$ matrix, and let $\boldsymbol{Q}$ be an orthonormal matrix that verifies* (5.1)*. In Step 1 of Algorithm 5.3, suppose that we compute an approximate solution $\boldsymbol{B}_{\mathrm{approx}}$ that minimizes the operator-norm*

ALGORITHM 5.3

Given an Hermitian matrix $\boldsymbol{A}$ and an orthonormal matrix $\boldsymbol{Q}$ that verify (5.1)*, a random test matrix $\boldsymbol{\Omega}$, and a sample matrix $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{\Omega}$, this algorithm computes an approximate eigenvalue decomposition $\boldsymbol{A} \approx \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^*$.*

1 Use a standard least-squares solver to find an Hermitian matrix $\boldsymbol{B}_{\text{approx}}$ that approximately satisfies the equation $\boldsymbol{B}_{\text{approx}}(\boldsymbol{Q}^*\boldsymbol{\Omega}) \approx \boldsymbol{Q}^*\boldsymbol{Y}$.
2 Compute the eigenvalue decomposition $\boldsymbol{B}_{\text{approx}} = \widehat{\boldsymbol{U}}\boldsymbol{\Lambda}\widehat{\boldsymbol{U}}^*$.
3 Form the product $\boldsymbol{U} = \boldsymbol{Q}\widehat{\boldsymbol{U}}$.

residual. If $\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^*$ is the eigenvalue decomposition of $\boldsymbol{A}$ produced by the algorithm, then

$$\|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^*\| \le 2\left(1 + \frac{\|\boldsymbol{\Omega}\|}{\sigma_{\min}}\right)\varepsilon, \tag{5.12}$$

where $\sigma_{\min}$ denotes the smallest singular value of $\boldsymbol{Q}^*\boldsymbol{\Omega}$.

Proof. Define $\boldsymbol{B} = \boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{Q}$. Relation (5.10) implies that $\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{B}\boldsymbol{Q}^*\| \le 2\varepsilon$. Since $\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^* = \boldsymbol{Q}\boldsymbol{B}_{\text{approx}}\boldsymbol{Q}^*$, we find that

$$\begin{aligned}
\|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^*\| &= \|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{B}_{\text{approx}}\boldsymbol{Q}^*\| \\
&= \|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{B}\boldsymbol{Q}^* + \boldsymbol{Q}(\boldsymbol{B} - \boldsymbol{B}_{\text{approx}})\boldsymbol{Q}^*\| \\
&\le \|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{B}\boldsymbol{Q}^*\| + \|\boldsymbol{B} - \boldsymbol{B}_{\text{approx}}\| \\
&\le 2\varepsilon + \|\boldsymbol{B} - \boldsymbol{B}_{\text{approx}}\|\,.
\end{aligned} \tag{5.13}$$

It remains to bound the second term on the right-hand side of (5.13).

Let $\boldsymbol{F}$ denote the residual error in solving the relation in Step 1 of Algorithm 5.3 so that

$$\boldsymbol{Q}^*\boldsymbol{Y} = \boldsymbol{B}_{\text{approx}}\boldsymbol{Q}^*\boldsymbol{\Omega} + \boldsymbol{F}.$$

We have the chain of relations

$$\begin{aligned}
\|\boldsymbol{B} - \boldsymbol{B}_{\text{approx}}\| &\le \frac{1}{\sigma_{\min}}\,\|(\boldsymbol{B} - \boldsymbol{B}_{\text{approx}})\boldsymbol{Q}^*\boldsymbol{\Omega}\| \\
&= \frac{1}{\sigma_{\min}}\,\|\boldsymbol{B}\boldsymbol{Q}^*\boldsymbol{\Omega} - \boldsymbol{Q}^*\boldsymbol{Y} + \boldsymbol{F}\| \\
&= \frac{1}{\sigma_{\min}}\,\|\boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{\Omega} - \boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{\Omega} + \boldsymbol{F}\| \\
&\le \frac{1}{\sigma_{\min}}\,(\|\boldsymbol{Q}^*\|\,\|\boldsymbol{A}\boldsymbol{Q}\boldsymbol{Q}^* - \boldsymbol{A}\|\,\|\boldsymbol{\Omega}\| + \|\boldsymbol{F}\|) \\
&\le \frac{1}{\sigma_{\min}}\,(\varepsilon\,\|\boldsymbol{\Omega}\| + \|\boldsymbol{F}\|)\,.
\end{aligned} \tag{5.14}$$

In order to bound $\|\boldsymbol{F}\|$, recall that $\boldsymbol{B}_{\text{approx}}$ is a minimum-residual solution to the relation $\boldsymbol{B}_{\text{approx}}(\boldsymbol{Q}^*\boldsymbol{\Omega}) \approx \boldsymbol{Q}^*\boldsymbol{Y}$. Therefore,

$$\begin{aligned}
\|\boldsymbol{F}\| &= \|\boldsymbol{Q}^*\boldsymbol{Y} - \boldsymbol{B}_{\text{approx}}\boldsymbol{Q}^*\boldsymbol{\Omega}\| \\
&\le \|\boldsymbol{Q}^*\boldsymbol{Y} - \boldsymbol{B}\boldsymbol{Q}^*\boldsymbol{\Omega}\| = \|\boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{\Omega} - \boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{\Omega}\| \\
&= \|\boldsymbol{Q}^*\boldsymbol{A}(\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^*)\boldsymbol{\Omega}\| \le \|\boldsymbol{A}(\boldsymbol{I} - \boldsymbol{Q}\boldsymbol{Q}^*)\|\,\|\boldsymbol{\Omega}\| \le \varepsilon\,\|\boldsymbol{\Omega}\|\,.
\end{aligned} \tag{5.15}$$

Together, (5.13), (5.14), and (5.15) imply the result (5.12). ∎

We have checked empirically that Algorithm 5.3 is effective when the test matrix $\boldsymbol{\Omega}$ is Gaussian and the number of samples ℓ is much smaller than the dimension n of the input matrix $\boldsymbol{A}$. In this situation, $\|\boldsymbol{\Omega}\| \sim \sqrt{n}$, and experiments indicate that $\sigma_{\min} \sim \sqrt{n}$, so the error bound (5.12) is quite strong. We have not established a rigorous result on the size of this minimum singular value, but this type of estimate is not essential in practice because we can inexpensively compute $\sigma_{\min}$ and $\|\boldsymbol{\Omega}\|$ to evaluate the bound on the approximation error.

If one encounters a situation where $\sigma_{\min}$ is small, then Algorithm 5.3 may fail. In this case, it may be necessary to use one of the schemes from §5.3 to produce the low-rank factorization or to start the approximation process again from scratch. In a streaming environment, unfortunately, the game is over.

When $\boldsymbol{A}$ is not Hermitian, it is still possible to devise single-pass algorithms, but we must modify the initial Stage A of the approximation framework to simultaneously construct bases for the ranges of $\boldsymbol{A}$ and $\boldsymbol{A}^*$:

1. Generate random matrices $\boldsymbol{\Omega}$ and $\widetilde{\boldsymbol{\Omega}}$.
2. Compute $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{\Omega}$ and $\widetilde{\boldsymbol{Y}} = \boldsymbol{A}^*\widetilde{\boldsymbol{\Omega}}$ in a single pass over $\boldsymbol{A}$.
3. Compute QR factorizations $\boldsymbol{Y} = \boldsymbol{Q}\boldsymbol{R}$ and $\widetilde{\boldsymbol{Y}} = \widetilde{\boldsymbol{Q}}\widetilde{\boldsymbol{R}}$.

This procedure results in matrices $\boldsymbol{Q}$ and $\widetilde{\boldsymbol{Q}}$ such that $\boldsymbol{A} \approx \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\widetilde{\boldsymbol{Q}}\widetilde{\boldsymbol{Q}}^*$. The reduced matrix we must approximate is $\boldsymbol{B} = \boldsymbol{Q}^*\boldsymbol{A}\widetilde{\boldsymbol{Q}}$. In analogy with (5.11), we find that

$$\boldsymbol{Q}^*\boldsymbol{Y} = \boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{\Omega} \approx \boldsymbol{Q}^*\boldsymbol{A}\widetilde{\boldsymbol{Q}}\widetilde{\boldsymbol{Q}}^*\boldsymbol{\Omega} = \boldsymbol{B}\widetilde{\boldsymbol{Q}}^*\boldsymbol{\Omega}. \tag{5.16}$$

An analogous calculation shows that $\boldsymbol{B}$ should also satisfy

$$\tilde{\boldsymbol{Q}}^*\widetilde{\boldsymbol{Y}} \approx \boldsymbol{B}^*\boldsymbol{Q}^*\widetilde{\boldsymbol{\Omega}}. \tag{5.17}$$

Now, the reduced matrix $\boldsymbol{B}_{\text{approx}}$ can be determined by finding a minimum-residual solution to the system of relations (5.16) and (5.17).

6. Computational costs. So far, we have postponed a detailed discussion of the computational cost of randomized matrix approximation algorithms because it is necessary to account for both the first stage, where we compute an approximate basis for the range (§4), and the second stage, where we postprocess the basis to complete the factorization (§5). We are now prepared to compare the cost of the two-stage scheme with the cost of traditional techniques.

Choosing an appropriate algorithm, whether classical or randomized, requires us to consider the properties of the input matrix. To provide a nuanced picture, we consider three different computational environments in §6.1–6.3. We close with some comments on parallel implementations in §6.4.

For concreteness, we focus on the problem of computing an approximate SVD of an $m \times n$ matrix $\boldsymbol{A}$ with numerical rank k. The costs for other factorizations are similar.

6.1. General matrices that fit in core memory. Suppose that $\boldsymbol{A}$ is a general matrix presented as an array of numbers that fits in core memory. In this case, the appropriate method for Stage A is to use a structured random matrix (§4.6), which allows us to find a basis that captures the action of the matrix using $\mathrm{O}(mn\log(k) + k^2m)$ flops. For Stage B, we apply the row-extraction technique (§5.2), which costs

an additional $\mathrm{O}(k^2(m+n))$ flops. The total number of operations T_{random} for this approach satisfies

$$T_{\text{random}} \sim mn\log(k) + k^2(m+n).$$

A *heuristic* error bound for this algorithm is

$$\|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*\| \lesssim n \cdot \sigma_{k+1}, \tag{6.1}$$

where σ_{k+1} is the $(k+1)$th singular value of $\boldsymbol{A}$. The estimate (6.1), which follows from Theorem 11.2 and Lemma 5.1, reflects the worst-case scenario; actual errors are usually smaller.

This algorithm should be compared with modern deterministic techniques, such as rank-revealing QR followed by postprocessing (§3.3.2) which requires

$$T_{\text{RRQR}} \sim kmn$$

operations to achieve a comparable error.

In this setting, the randomized algorithm is faster than classical techniques for problems of moderate size, say $m, n \sim 10^3$ and $k \sim 10^2$. See §7.5 for numerical evidence.

REMARK 6.1. In case row extraction is impractical, there is an alternative $\mathrm{O}(mn\log(k))$ technique described in [135, §5.2]. When the error (6.1) is unacceptably large, we can use the direct method (§5.1) for Stage B, which brings the total cost to $\mathrm{O}(kmn)$ flops.

6.2. Matrices for which matrix–vector products can be rapidly evaluated. In many problems in data mining and scientific computing, the cost T_{mult} of performing the matrix–vector multiplication $\boldsymbol{x} \mapsto \boldsymbol{A}\boldsymbol{x}$ is substantially smaller than the nominal cost $\mathrm{O}(mn)$ for the dense case. It is not uncommon that $\mathrm{O}(m+n)$ flops suffice. Standard examples include (i) very sparse matrices; (ii) structured matrices, such as Töplitz operators, that can be applied using the FFT or other means; and (iii) matrices that arise from physical problems, such as discretized integral operators, that can be applied via, e.g., the fast multipole method [66].

Suppose that both $\boldsymbol{A}$ and $\boldsymbol{A}^*$ admit fast multiplies. The appropriate randomized approach for this scenario completes Stage A using Algorithm 4.1 with p constant (for the fixed-rank problem) or Algorithm 4.2 (for the fixed-precision problem) at a cost of $\mathrm{O}(k\,T_{\text{mult}} + k^2 m)$ flops. For Stage B, we invoke Algorithm 5.1, which requires $\mathrm{O}(k\,T_{\text{mult}} + k^2(m+n))$ flops. The total cost T_{sparse} satisfies

$$T_{\text{sparse}} \sim k\,T_{\text{mult}} + k^2(m+n). \tag{6.2}$$

A *heuristic* error bound is

$$\|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*\| \lesssim \sqrt{kn} \cdot \sigma_{k+1}.$$

This estimate follows from Corollary 10.9 and the discussion in §5.1.

When the singular spectrum of $\boldsymbol{A}$ decays slowly, we can incorporate q iterations of the power method (Algorithm 4.3) to obtain superior solutions to the fixed-rank problem. The computational profile is similar to (6.2), but the *heuristic* error bound improves to

$$\|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*\| \lesssim (kn)^{1/2(2q+1)} \cdot \sigma_{k+1}.$$

This estimate takes into account the discussion in §10.4. The power scheme can also be adapted for the fixed-precision problem (§4.5).

In this setting, the classical technique for obtaining a partial SVD employs a Krylov-subspace method (§3.3.4), whose computational cost matches (6.2), assuming proper convergence. Unfortunately, the performance of these methods can depend substantially on whether the singular values of the matrix are tightly clustered (bad), spread out (good), or repeated (better).

Sampling algorithms have two advantages over traditional techniques. First, the performance of the randomized scheme does not depend on the fine structure of the singular spectrum. More important, we have tremendous freedom to organize the computation. For the fixed-rank problem, we can compute all $\mathrm{O}(k)$ matrix–vector products simultaneously; for the fixed-precision problem, we can achieve substantial savings by blocking the computation. Similar economies obtain in the postprocessing of Stage B. In contrast, Krylov methods necessarily perform all the matrix–vector multiplies in serial.

REMARK 6.2. The power scheme, Algorithm 4.3, is conceptually related to Krylov-subspace methods initialized with a random starting vector. To explain this point succinctly, we assume that $\boldsymbol{A}$ is square. Krylov techniques *implicitly* compress the matrix to the subspace

$$\mathcal{V}_q(\boldsymbol{\omega}) = \operatorname{span}\{\boldsymbol{\omega}, \boldsymbol{A}\boldsymbol{\omega}, \boldsymbol{A}^2\boldsymbol{\omega}, \dots, \boldsymbol{A}^{q-1}\boldsymbol{\omega}\}$$

and perform spectral computations on the reduced matrix. In comparison, the power scheme *explicitly* compresses the matrix to a subspace formed using many random starting vectors

$$\mathcal{W}_q(\boldsymbol{\omega}) = \operatorname{span}\big\{\mathcal{V}_q(\boldsymbol{\omega}^{(1)}), \mathcal{V}_q(\boldsymbol{\omega}^{(2)}), \dots, \mathcal{V}_q(\boldsymbol{\omega}^{(\ell)})\big\}$$

and performs spectral computations on the reduced matrix. In the power scheme, the exponent q is usually much smaller than for traditional Krylov methods, while the number ℓ of random vectors is substantial.

We view the power scheme as a hybrid that enjoys the best features of both randomized algorithms and Krylov-subspace methods. Exploring this connection may lead to further advances. See [80, 87] for an analysis of Krylov-subspace methods with a random start and [111, 113] for work on the randomized power scheme.

6.3. General matrices stored in slow memory or streamed. The traditional metric for numerical algorithms is the number of floating-point operations they require. When the data does not fit in fast memory, however, the computational time is often dominated by the cost of memory access. In this setting, a more appropriate measure of algorithmic performance is *pass-efficiency*, which counts how many times the data needs to be cycled through fast memory. Flop counts become largely irrelevant.

All the classical matrix factorization techniques that we discuss in §3.2—including dense SVD, rank-revealing QR, Krylov methods, and so forth— require at least k passes over the the matrix, which is prohibitively expensive for huge data matrices. A desire to reduce the pass count of matrix approximation algorithms served as one of the early motivations for developing randomized schemes [46, 58, 105]. Detailed recent work appears in [32].

For many matrices, randomized techniques can produce an accurate approximation using just one pass over the data. For Hermitian matrices, we obtain a single-pass

algorithm by combining Algorithm 4.1 to construct an approximate basis with Algorithm 5.3, which produces an eigenvalue decomposition without any additional access to the matrix. Section 5.4 describes the analogous technique for general matrices.

For the huge matrices that arise in information science problems, it is common that the singular spectrum decays slowly. Relevant applications include image processing (see §§7.3–7.4 for numerical examples), statistical data analysis, and network monitoring. To compute approximate factorizations in these environments, it is crucial to enhance the accuracy of the randomized approach using the power scheme, Algorithm 4.3, or some other device. This approach increases the pass count somewhat, but four to eight passes are usually sufficient.

6.4. Gains from parallelization. As mentioned in §§6.2–6.3, randomized methods often outperform classical techniques not because they involve fewer floating-point operations but rather because they allow us to reorganize the calculations to exploit the matrix properties and the computer architecture more fully. In addition, these methods are well suited for parallel implementation. For example, in Algorithm 4.1, the computational bottleneck is the evaluation of the matrix product $\boldsymbol{A\Omega}$, which is embarrassingly parallelizable.

7. Numerical examples. At this point, a question presents itself. Do these randomized matrix approximation algorithms actually work in practice? In this section, we attempt to address this concern by illustrating how the algorithms perform on a diverse collection of test cases.

Let us begin with some problems from the physical sciences. First, we consider two matrices with exponentially decaying spectra that arise from discretizations of integral operators (§7.1). Then we attempt to approximate the inverse of a finite-difference operator, where our only access to the matrix is a "matrix–vector multiply" that is accomplished by solving a linear system (§7.2).

Afterward, we turn to some examples from the information sciences. Section 7.3 investigates how well we can approximate a large graph Laplacian that describes the local geometry of an image. We also evaluate the pass-efficient algorithms on an enormous database of face images that requires nearly 5.6 GB of storage in its uncompressed form (§7.4).

Finally, we investigate the performance of randomized methods based on structured random matrices (§7.5).

Sections 7.1–7.4 focus on the algorithms for Stage A that we presented in §4 because we wish to isolate the performance of the randomized step. The eigenface problem, however, is so massive that the methods in §5 for Stage B provide the only practical way to assess the performance!

7.1. Matrices associated with compact integral operators. We study the behavior of the adaptive range approximation method, Algorithm 4.2, by applying it to two matrices we obtained by discretizing compact integral operators that arise in potential theory. The singular spectrum of each operator decays exponentially fast, which is a very favorable environment for random sampling schemes.

We first consider a 200×200 matrix $\boldsymbol{A}$ resulting from the discretization of the following single-layer operator associated with the Laplace equation:

$$[S\sigma](x) = \text{const} \cdot \int_{\Gamma_1} \log|x - y| \; \sigma(y) \, \mathrm{d}A(y), \qquad x \in \Gamma_2, \tag{7.1}$$

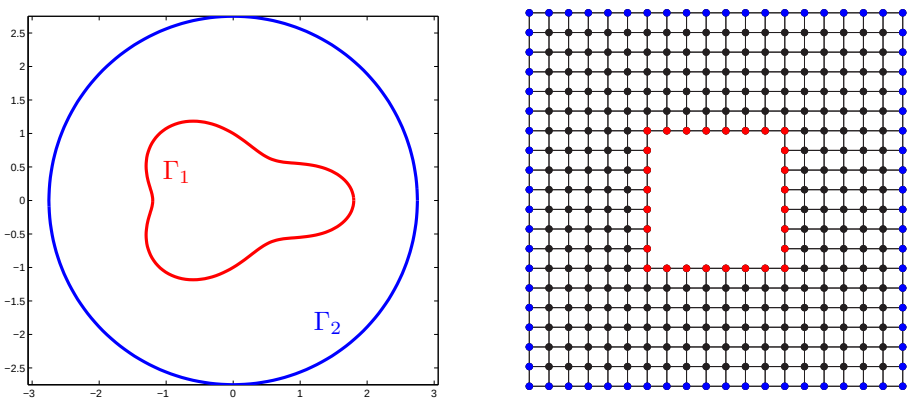

FIG. 7.1. Configurations for physical problems. (a) *The contours* Γ_1 *(red) and* Γ_2 *(blue) for the integral operators* (7.1) *and* (7.2). (b) *Geometry of the lattice problem described in* §7.2.

where Γ_1 and Γ_2 are the two contours in $\mathbb{R}^2$ illustrated in Figure 7.1(a). We approximate the integral with the trapezoidal rule, which converges superalgebraically because the kernel is smooth. In the absence of floating-point errors, we estimate that the discretization error would be less than 10^{-20} for a smooth source σ. The leading constant is selected so the matrix $\boldsymbol{A}$ has unit operator norm.

We implement Algorithm 4.2 in Matlab v6.5. Gaussian test matrices are generated using the `randn` command. For each number ℓ of samples, we compare the following three quantities:

1. The minimum rank-ℓ approximation error $\sigma_{\ell+1}$ is determined using `svd`.
2. The actual error $e_\ell = \left\| \left(\mathbf{I} - \boldsymbol{Q}^{(\ell)}(\boldsymbol{Q}^{(\ell)})^*\right)\boldsymbol{A} \right\|$ is computed with `norm`.
3. A random estimator f_ℓ for the actual error e_ℓ is obtained from (4.3), with the parameter r set to 5.

Note that any values less than 10^{-15} should be considered numerical artifacts.

Figure 7.2 tracks a characteristic execution of Algorithm 4.2. We make three observations: (i) The error e_ℓ incurred by the algorithm is remarkably close to the theoretical minimum $\sigma_{\ell+1}$. (ii) The error estimate always produces an upper bound for the actual error. Without the built-in $10\times$ safety margin, the estimate would track the actual error almost exactly. (iii) The basis constructed by the algorithm essentially reaches full double-precision accuracy.

How typical is the trial documented in Figure 7.2? To answer this question, we examine the empirical performance of the algorithm over 2,000 independent trials. Figure 7.3 charts the error estimate versus the actual error at four points during the course of execution: $\ell = 25, 50, 75, 100$. We offer four observations: (i) The initial run detailed in Figure 7.2 is entirely typical. (ii) Both the actual and estimated error concentrate about their mean value. (iii) The actual error drifts slowly away from the optimal error as the number ℓ of samples increases. (iv) The error estimator is *always* pessimistic by a factor of about ten, which means that the algorithm *never* produces a basis with lower accuracy than requested. The only effect of selecting an unlucky sample matrix $\boldsymbol{\Omega}$ is that the algorithm proceeds for a few additional steps.

We repeat the same experiments for a 400×400 complex matrix $\boldsymbol{B}$ obtained when

we discretize an integral operator inspired by acoustic scattering theory:

$$[S\sigma](x) = \text{const} \cdot \int_{\Gamma_1} H_0^{(1)}(k\,|x-y|)\,\sigma(y)\,\mathrm{d}A(y), \qquad x \in \Gamma_2, \tag{7.2}$$

where $H_0^{(1)}$ is the zeroth-order Hankel function of the first kind. The contours Γ_1 and Γ_2 are depicted in Figure 7.1(a). As the diagram shows, the diameter L of the inner contour Γ_2 slightly exceeds two, and we set the Helmholtz parameter $k = 30$. The leading constant is chosen so that the resulting matrix $\boldsymbol{B}$ has unit operator norm. The physics of the problem ensure that the operator S has $\mathrm{O}(kL)$ singular values of order one, after which the spectrum decays rapidly.

Figure 7.4 documents a typical test run, and Figure 7.5 compiles statistics for 2,000 trials. Qualitatively, the algorithm behaves the same as for the Laplace problem.

7.2. Approximating the inverse of a finite-difference operator. Next, we consider a situation where the input matrix $\boldsymbol{A}$ is (a restriction of) the inverse of a very large, sparse matrix obtained from a finite-difference approximation to a continuous differential operator. We have access to the matrix only through the "matrix–vector multiply" $\boldsymbol{\omega} \mapsto \boldsymbol{A}\boldsymbol{\omega}$, which is computed by solving a sparse linear system.

More precisely, we model an electrostatics problem on the square lattice network shown in Figure 7.1(b). The electric potential at the inner boundary nodes (red) is fixed at the values specified in the data vector $\boldsymbol{u}_{\text{inner}}$. The outer boundary (blue) is insulated. The electric potential of each internal node (black) is the average of the potentials at its four nearest neighbors. The mathematical model for this system is the discrete Laplace equation (using the five-point stencil), coupled with a zero Neumann boundary condition on the exterior nodes and a Dirichlet boundary condition specified by $\boldsymbol{u}_{\text{inner}}$ on the interior nodes. The matrix $\boldsymbol{A}$ that we seek to approximate comes from the linear map

$$\boldsymbol{A} : \boldsymbol{u}_{\text{inner}} \longmapsto \boldsymbol{u}_{\text{outer}}$$

that describes the potential induced on the exterior nodes by the interior potential.

For our experiments, we fix a number n and form the lattice with $4n$ nodes along the inner boundary and $12n$ nodes along the outer boundary. We form the discrete Laplace operator for the resulting donut-shaped lattice with roughly $8n^2$ nodes, and we complete each evaluation $\boldsymbol{\omega} \mapsto \boldsymbol{A}\boldsymbol{\omega}$ using the Matlab `backslash` operator, which yields a residual error below 10^{-15}.

We apply Algorithm 4.2 to the operator $\boldsymbol{A}$ obtained when $n = 133$, and we repeat the experimental regimen outlined in §7.1. The results of a typical trial appear in Figure 7.6. Qualitatively, the performance matches the experiments in §7.1.

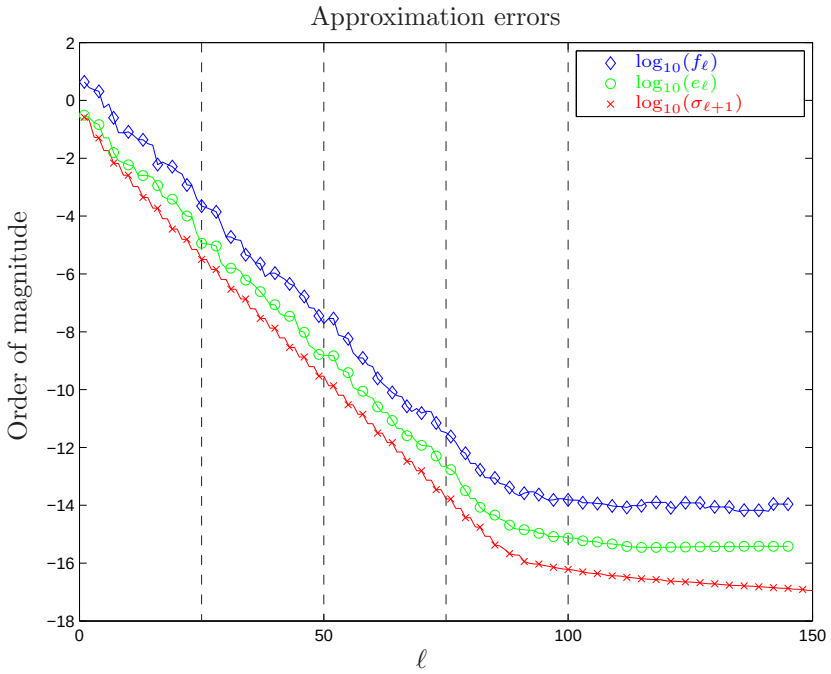

FIG. 7.2. Approximating a Laplace integral operator. *One execution of Algorithm 4.2 for the* 200×200 *input matrix* $\boldsymbol{A}$ *described in §7.1. The number* ℓ *of random samples varies along the horizontal axis; the vertical axis measures the base-10 logarithm of error magnitudes. The dashed vertical lines mark the points during execution at which Figure 7.3 provides additional statistics.*

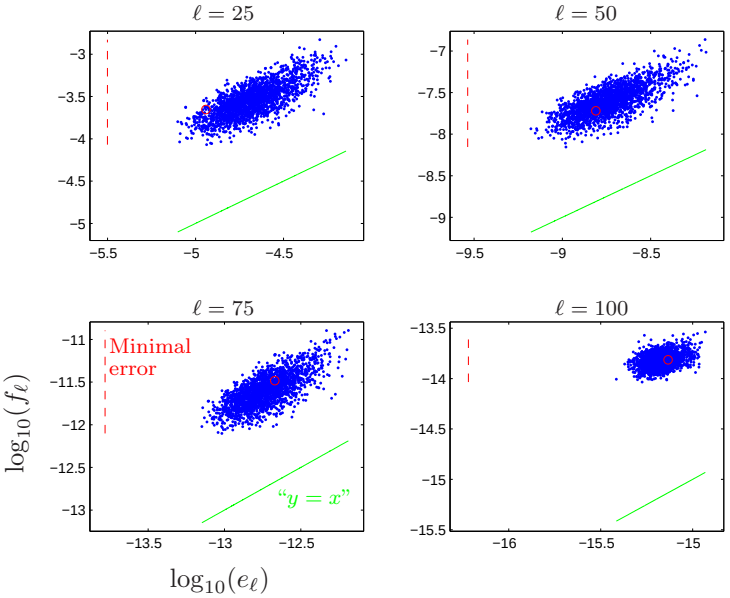

FIG. 7.3. Error statistics for approximating a Laplace integral operator. *2,000 trials of Algorithm 4.2 applied to a* 200×200 *matrix approximating the integral operator* (7.1). *The panels isolate the moments at which* $\ell = 25, 50, 75, 100$ *random samples have been drawn. Each solid point compares the estimated error* f_ℓ *versus the actual error* e_ℓ *in one trial; the open circle indicates the trial detailed in Figure 7.2. The dashed line identifies the minimal error* $\sigma_{\ell+1}$*, and the solid line marks the contour where the error estimator would equal the actual error.*

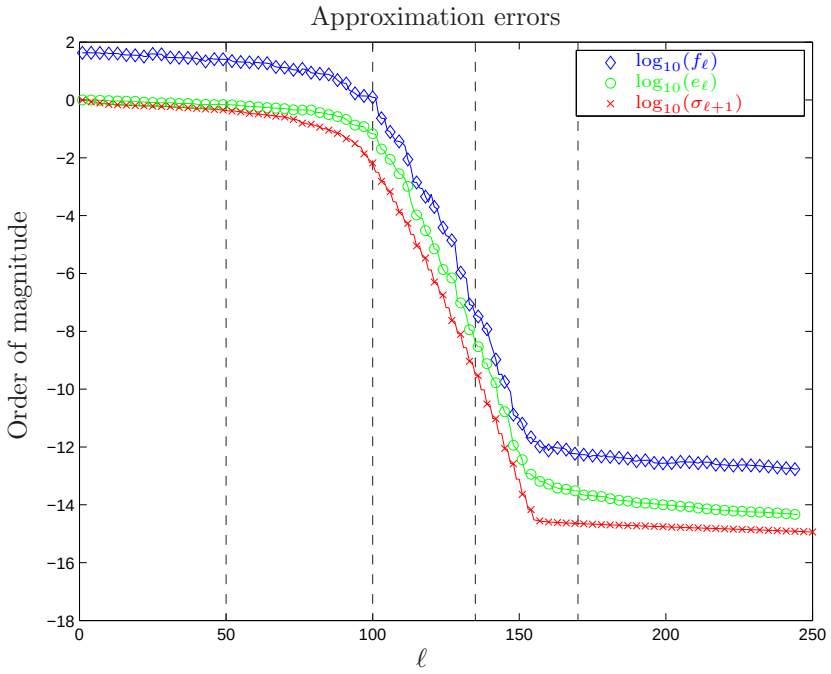

FIG. 7.4. Approximating a Helmholtz integral operator. *One execution of Algorithm 4.2 for the* 400×400 *input matrix* $\boldsymbol{B}$ *described in §7.1. See Figure 7.2 for notations.*

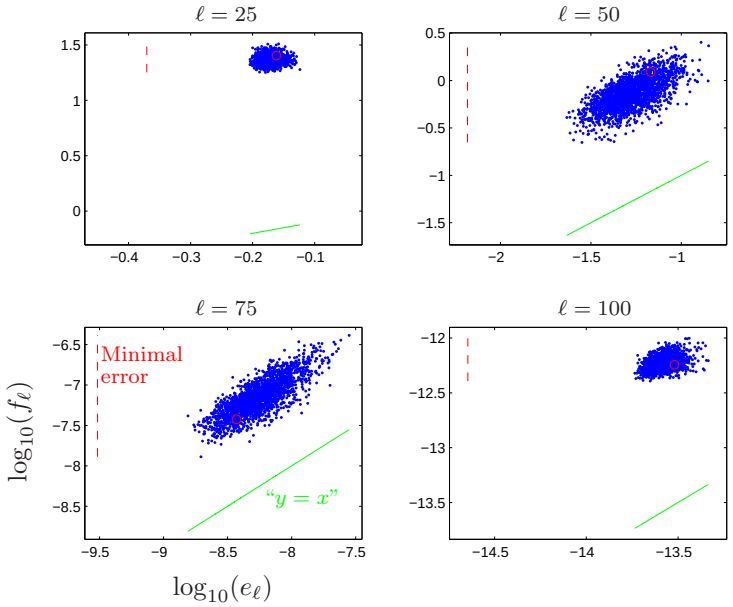

FIG. 7.5. Error statistics for approximating a Helmholtz integral operator. *2,000 trials of Algorithm 4.2 applied to a* 400×400 *matrix approximation the integral operator* (7.2). *See Figure 7.3 for notations.*

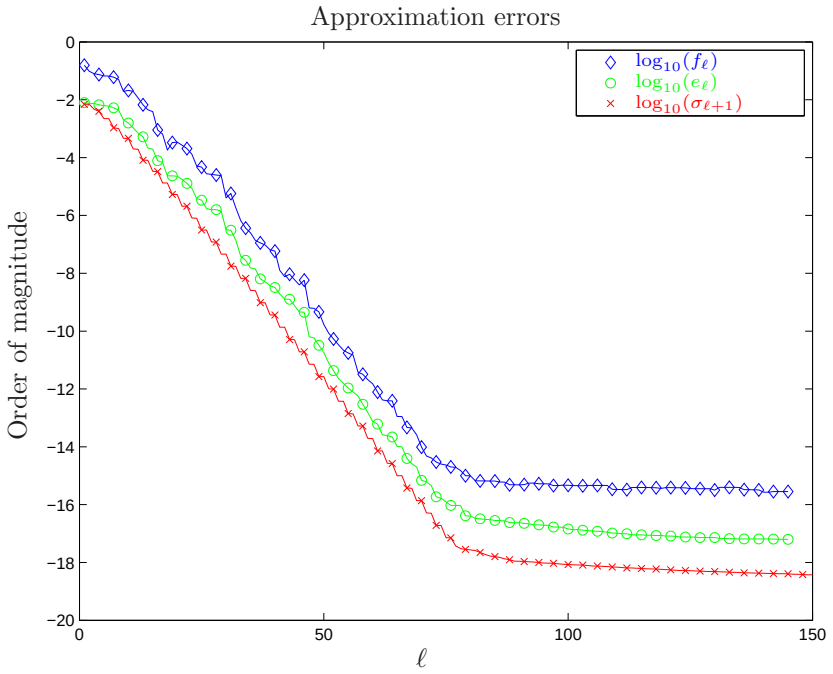

FIG. 7.6. Approximating the inverse of a discrete Laplacian. *One execution of Algorithm 4.2 for the* 1596×532 *input matrix* $\boldsymbol{A}$ *described in §7.2. See Figure 7.2 for notations.*

7.3. A large, sparse, noisy matrix arising in image processing. Our next example involves a matrix that arises in image processing. A recent line of work uses information about the local geometry of an image to develop promising new algorithms for standard tasks, such as denoising, inpainting, and so forth. These methods are based on approximating a *graph Laplacian* associated with the image. The dominant eigenvectors of this matrix provide "coordinates" that help us smooth out noisy image patches [120, 130].

We begin with a 95×95 pixel grayscale image. The intensity of each pixel is represented as an integer in the range 0 to $4,095$. We form for each pixel i a vector $\boldsymbol{x}^{(i)} \in \mathbb{R}^{25}$ by gathering the 25 intensities of the pixels in a 5×5 neighborhood centered at pixel i (with appropriate modifications near the edges). Next, we form the $9,025 \times 9,025$ *weight matrix* $\widetilde{\boldsymbol{W}}$ that reflects the similarities between patches:

$$\widetilde{w}_{ij} = \exp\big\{ - \big\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\big\|^2 / \sigma^2 \big\},$$

where the parameter $\sigma = 50$ controls the level of sensitivity. We obtain a sparse weight matrix $\boldsymbol{W}$ by zeroing out all entries in $\widetilde{\boldsymbol{W}}$ except the seven largest ones in each row. The object is then to construct the low frequency eigenvectors of the graph Laplacian matrix

$$\boldsymbol{L} = \mathbf{I} - \boldsymbol{D}^{-1/2} \boldsymbol{W} \boldsymbol{D}^{-1/2},$$

where $\boldsymbol{D}$ is the diagonal matrix with entries $d_{ii} = \sum_j w_{ij}$. These are the eigenvectors associated with the dominant eigenvalues of the auxiliary matrix $\boldsymbol{A} = \boldsymbol{D}^{-1/2} \boldsymbol{W} \boldsymbol{D}^{-1/2}$.

The matrix $\boldsymbol{A}$ is very large, and its eigenvalues decay slowly so we must use the power scheme summarized in Algorithm 4.3 to approximate it. Figure 7.7(left) illustrates how the approximation error e_ℓ declines as the number ℓ of samples increases.

When we set the exponent $q = 0$, which corresponds with the basic Algorithm 4.1, the approximation is rather poor. The graph illustrates that increasing the exponent q slightly results in a tremendous improvement in the accuracy of the power scheme.

Next, we illustrate the results of using the two-stage approach to approximate the eigenvalues of $\boldsymbol{A}$. In Stage A, we construct a basis for $\boldsymbol{A}$ using Algorithm 4.3 with $\ell = 100$ samples for different values of q. In Stage B, we apply the Hermitian variant of Algorithm 5.1 described in §5.3 to compute an approximate eigenvalue decomposition. Figure 7.7(right) shows how the approximate eigenvalues compare with the actual eigenvalues of $\boldsymbol{A}$. Once again, we see that the minimal exponent $q = 0$ produces miserable results, but the largest eigenvalues are already quite accurate when we $q = 1$.

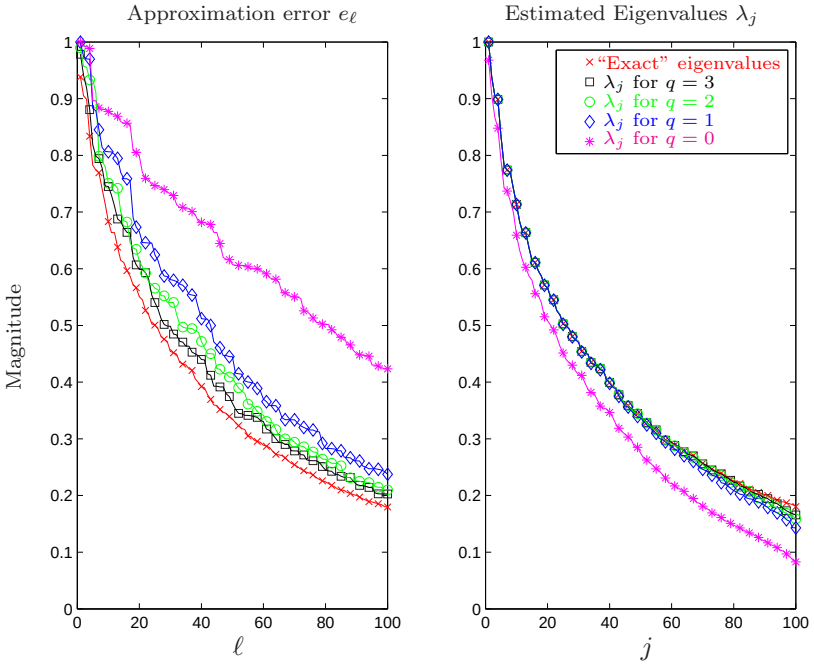

FIG. 7.7. Approximating a graph Laplacian. *For varying exponent q, one trial of the power scheme, Algorithm 4.3, applied to the $10,000 \times 10,000$ matrix $\boldsymbol{A}$ described in §7.3.* (Left) *Approximation errors as a function of the number ℓ of random samples.* (Right) *Estimates for the 100 largest eigenvalues given $\ell = 100$ random samples compared with the 100 largest eigenvalues of $\boldsymbol{A}$.*

7.4. Eigenfaces. Our next example involves a large, dense matrix derived from the FERET databank of face images [107, 108]. A simple method for performing face recognition is to identify the principal directions of the image data, which are called *eigenfaces*. Each of the original photographs can be summarized by its components along these principal directions. To identify the subject in a new picture, we compute its decomposition in this basis and use a classification technique, such as nearest neighbors, to select the closest image in the database [122].

We construct a data matrix $\boldsymbol{A}$ in the following manner. The FERET database contains $7,254$ photographs, and each 384×256 image contains $98,304$ pixels. First, we build an auxiliary $98,304 \times 7,254$ matrix $\widetilde{\boldsymbol{A}}$ whose columns are the images. We form $\boldsymbol{A}$ by centering each column of $\widetilde{\boldsymbol{A}}$ and scaling it to unit norm, so that the images are roughly comparable. The eigenfaces are the dominant left singular vectors of this

matrix.

Our goal then is to compute an approximate SVD of the matrix $\boldsymbol{A}$. Represented as an array of double-precision real numbers, $\boldsymbol{A}$ would require 5.4 GB of storage, which does not fit within the fast memory of many machines. It is possible to compress the database down to at 57 MB or less (in JPEG format), but then the data would have to be uncompressed with each sweep over the matrix. Furthermore, the matrix $\boldsymbol{A}$ has slowly decaying singular values, so we need to use the power scheme, Algorithm 4.3, to capture the range of the matrix accurately.

To address these concerns, we implemented the power scheme to run in a pass-efficient manner. An additional difficulty arises because the size of the data makes it prohibitively expensive to calculate the actual error e_ℓ incurred by the approximation or to determine the minimal error $\sigma_{\ell+1}$. To estimate the errors, we use the technique described in Remark 4.1.

Figure 7.8 describes the behavior of the power scheme, which is similar to its performance for the graph Laplacian in §7.3. When the exponent $q = 0$, the approximation of the data matrix is very poor, but it improves quickly as q increases. Likewise, the estimate for the spectrum of $\boldsymbol{A}$ appears to converge rapidly; the largest singular values are already quite accurate when $q = 1$. We see essentially no improvement in the estimates after the first 3–5 passes over the matrix.

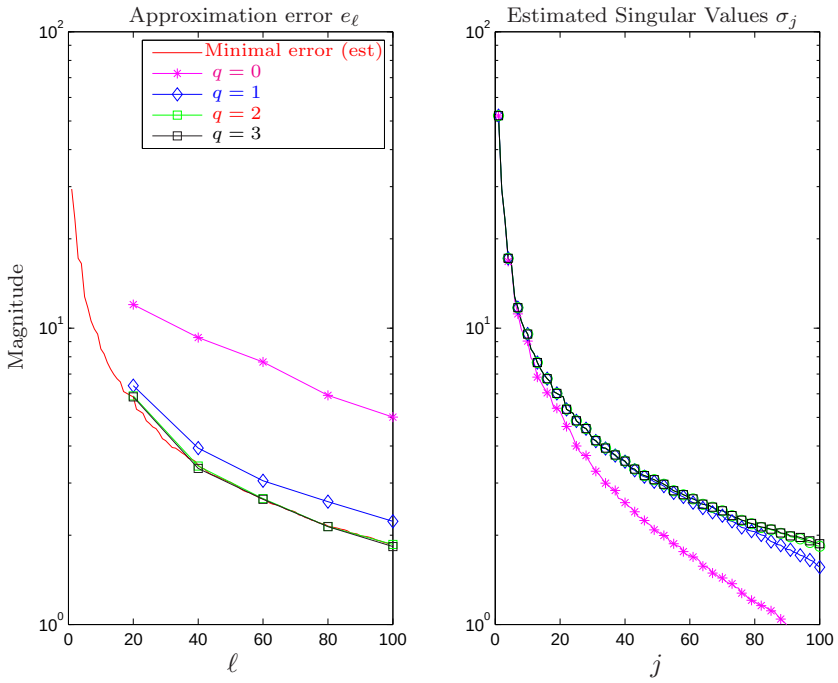

FIG. 7.8. Computing eigenfaces. *For varying exponent q, one trial of the power scheme, Algorithm 4.3, applied to the $98,304 \times 7,254$ matrix $\boldsymbol{A}$ described in §7.4.* (Left) *Approximation errors as a function of the number ℓ of random samples. The red line indicates the minimal errors as estimated by the singular values computed using $\ell = 100$ and $q = 3$.* (Right) *Estimates for the 100 largest eigenvalues given $\ell = 100$ random samples.*

7.5. Performance of structured random matrices. Our final set of experiments illustrates that the structured random matrices described in §4.6 lead to matrix approximation algorithms that are both fast and accurate.

First, we compare the computational speeds of four methods for computing an approximation to the ℓ dominant terms in the SVD of an $n \times n$ Gaussian matrix.

Method	**Stage A**	**Stage B**
`direct`	Rank-revealing QR executed using column pivoting and Householder reflectors	Algorithm 5.1
`gauss`	Algorithm 4.1 with a Gaussian random matrix	Algorithm 5.1
`srft`	Algorithm 4.1 with the modified SRFT (4.8)	Algorithm 5.2
`svd`	Full SVD with LAPACK routine `dgesdd`	Truncate to ℓ terms

Applying these algorithms to approximate a Gaussian matrix represents no loss of generality because the size of the input matrix is the only factor relevant to the runtime. See Remark 7.1 for implementation details.

Table 7.1 lists the measured runtime of a single execution of each algorithm for various choices of the dimension n of the input matrix and the rank ℓ of the approximation. Of course, the cost of the full SVD does not depend on the number ℓ of components required.

A more informative way to look at the runtime data is to compare the *relative* cost of the algorithms. The `direct` method is the best deterministic approach for dense matrices, so we calculate the factor by which the randomized methods improve on this benchmark. Figure 7.9 displays the results. We make two observations: (i) Using an SRFT often leads to a dramatic speed-up over classical techniques, even for moderate problem sizes. (ii) Using a standard Gaussian test matrix typically leads to a moderate speed-up over classical methods, primarily because matrix–matrix multiplication is more efficient than rank-revealing QR.

Second, we investigate how the choice of random test matrix influences the error in approximating an input matrix. For these experiments, we return to the 200×200 matrix $\boldsymbol{A}$ defined in Section 7.1. Consider variations of Algorithm 4.1 obtained when the random test matrix $\boldsymbol{\Omega}$ is drawn from the following four distributions:

Gauss: The standard Gaussian distribution.

Ortho: The uniform distribution on $n \times \ell$ orthonormal matrices.

SRFT: The SRFT distribution defined in (4.6).

GSRFT: The modified SRFT distribution defined in (4.8).

Intuitively, we expect that **Ortho** should provide the best performance.

For each distribution, we perform 100,000 trials of the following experiment. Apply the corresponding version of Algorithm 4.1 to the matrix $\boldsymbol{A}$, and calculate the approximation error $e_\ell = \|\boldsymbol{A} - \boldsymbol{Q}_\ell \boldsymbol{Q}_\ell^* \boldsymbol{A}\|$. Figure 7.10 displays the empirical probability density function for the error e_ℓ obtained with each algorithm. We offer three observations: (i) The SRFT actually performs slightly *better* than a Gaussian random matrix for this example. (ii) The standard SRFT and the modified SRFT have essentially identical errors. (iii) There is almost no difference between the Gaussian random matrix and the random orthonormal matrix in the first three plots, while the fourth plot shows that the random orthonormal matrix performs better. This behavior occurs because, with high probability, a tall Gaussian matrix is well conditioned and a (nearly) square Gaussian matrix is not.

TABLE 7.1
Computational times for a partial SVD. *The time, in seconds, required to compute the ℓ leading components in the SVD of an $n \times n$ matrix using each of the methods from §7.5. The last row indicates the time needed to obtain a full SVD.*

	$n = 1,024$			$n = 2,048$			$n = 4,096$		
ℓ	`direct`	`gauss`	`srft`	`direct`	`gauss`	`srft`	`direct`	`gauss`	`srft`
10	1.08e-1	5.63e-2	9.06e-2	4.22e-1	2.16e-1	3.56e-1	1.70e 0	8.94e-1	1.45e 0
20	1.97e-1	9.69e-2	1.03e-1	7.67e-1	3.69e-1	3.89e-1	3.07e 0	1.44e 0	1.53e 0
40	3.91e-1	1.84e-1	1.27e-1	1.50e 0	6.69e-1	4.33e-1	6.03e 0	2.64e 0	1.63e 0
80	7.84e-1	4.00e-1	2.19e-1	3.04e 0	1.43e 0	6.64e-1	1.20e 1	5.43e 0	2.08e 0
160	1.70e 0	9.92e-1	6.92e-1	6.36e 0	3.36e 0	1.61e 0	2.46e 1	1.16e 1	3.94e 0
320	3.89e 0	2.65e 0	2.98e 0	1.34e 1	7.45e 0	5.87e 0	5.00e 1	2.41e 1	1.21e 1
640	1.03e 1	8.75e 0	1.81e 1	3.14e 1	2.13e 1	2.99e 1	1.06e 2	5.80e 1	5.35e 1
1280	—	—	—	7.97e 1	6.69e 1	3.13e 2	2.40e 2	1.68e 2	4.03e 2
`svd`	1.19e 1			8.77e 1			6.90e 2		

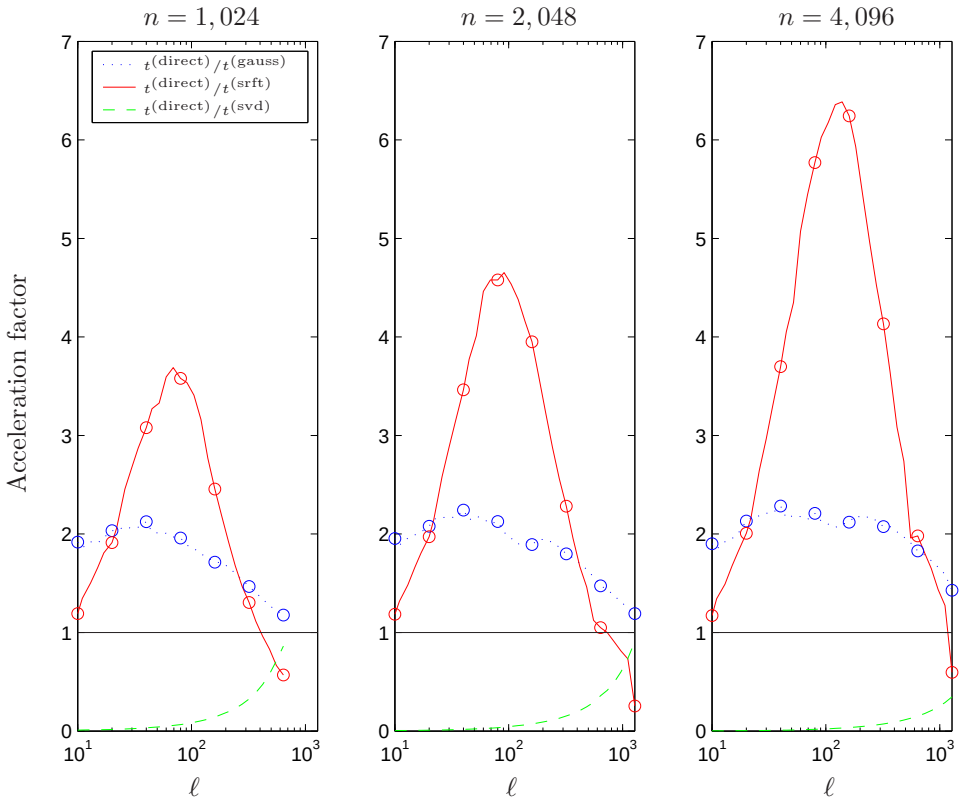

FIG. 7.9. Acceleration factor. *The relative cost of computing an ℓ-term partial SVD of an $n \times n$ Gaussian matrix using* `direct`*, a benchmark classical algorithm, versus each of the three competitors described in §7.5. The solid red curve shows the speedup using an SRFT test matrix, and the dotted blue curve shows the speedup with a Gaussian test matrix. The dashed green curve indicates that a full SVD computation using classical methods is substantially* slower. *Table 7.1 reports the absolute runtimes that yield the circled data points.*

REMARK 7.1. The running times reported in Table 7.1 and in Figure 7.9 depend strongly on both the computer hardware and the coding of the algorithms. The experiments reported here were performed on a standard office desktop with a 3.2 GHz Pentium IV processor and 2 GB of RAM. The algorithms were implemented in Fortran 90 and compiled with the Lahey compiler. The Lahey versions of BLAS and LAPACK were used to accelerate all matrix–matrix multiplications, as well as the SVD computations in Algorithms 5.1 and 5.2. We used the code for the modified SRFT (4.8) provided in the publicly available software package `id_dist` [93].

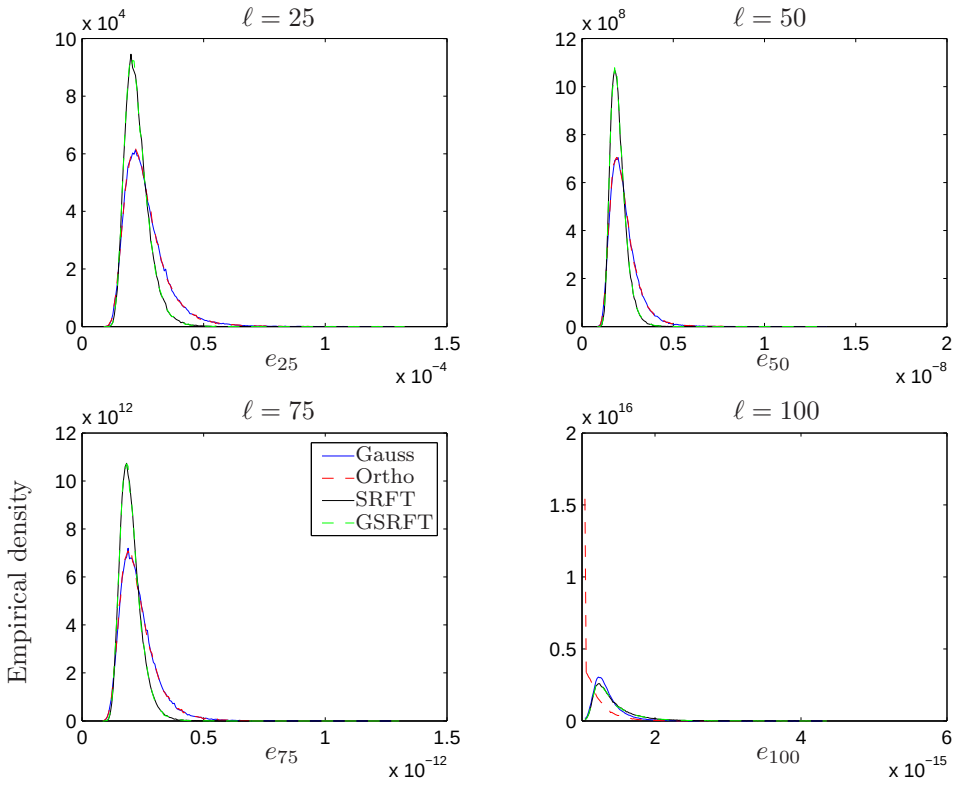

FIG. 7.10. Empirical probability density functions for the error in Algorithm 4.1. *As described in §7.5, the algorithm is implemented with four distributions for the random test matrix and used to approximate the* 200×200 *input matrix obtained by discretizing the integral operator* (7.1). *The four panels capture the empirical error distribution for each version of the algorithm at the moment when* $\ell = 25, 50, 75, 100$ *random samples have been drawn.*

Part III: Theory

This part of the paper, §§8–11, provides a detailed analysis of randomized sampling schemes for constructing an approximate basis for the range of a matrix, the task we refer to as Stage A in the framework of §1.2. More precisely, we assess the quality of the basis $\boldsymbol{Q}$ that the proto-algorithm of §1.3 produces by establishing rigorous bounds for the approximation error

$$\|\!|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}|\!\|, \tag{7.3}$$

where $\|\!|\cdot|\!\|$ denotes either the spectral norm or the Frobenius norm. The difficulty in developing these bounds is that the matrix $\boldsymbol{Q}$ is random, and its distribution is a complicated nonlinear function of the input matrix $\boldsymbol{A}$ and the random test matrix $\boldsymbol{\Omega}$. Naturally, any estimate for the approximation error must depend on the properties of the input matrix and the distribution of the test matrix.

To address these challenges, we split the argument into two pieces. The first part exploits techniques from linear algebra to deliver a generic bound for the error that depends on the interaction between the test matrix $\boldsymbol{\Omega}$ and the right singular vectors of the input matrix $\boldsymbol{A}$, as well as the tail singular values of $\boldsymbol{A}$. In the second part of the argument, we take into account the distribution of the random matrix to estimate the error for specific instantiations of the proto-algorithm. This bipartite proof is common in the literature on randomized linear algebra, but our argument is most similar in spirit to [17].

Section 8 surveys the basic linear algebraic tools we need. Section 9 uses these methods to derive a generic error bound. Afterward, we specialize this results to case where the test matrix is Gaussian (§10) and the case where the test matrix is a subsampled random Fourier transform (§11).

8. Theoretical preliminaries. We proceed with some additional background from linear algebra. Section 8.1 sets out properties of positive-semidefinite matrices, and §8.2 offers some results for orthogonal projectors. Standard references for this material include [11, 72].

8.1. Positive semidefinite matrices. An Hermitian matrix $\boldsymbol{M}$ is *positive semidefinite* (briefly, *psd*) when $\boldsymbol{u}^*\boldsymbol{M}\boldsymbol{u} \geq 0$ for all $\boldsymbol{u} \neq \boldsymbol{0}$. If the inequalities are strict, $\boldsymbol{M}$ is *positive definite* (briefly, *pd*). The psd matrices form a convex cone, which induces a partial ordering on the linear space of Hermitian matrices: $\boldsymbol{M} \preccurlyeq \boldsymbol{N}$ if and only if $\boldsymbol{N} - \boldsymbol{M}$ is psd. This ordering allows us to write $\boldsymbol{M} \succcurlyeq \boldsymbol{0}$ to indicate that the matrix $\boldsymbol{M}$ is psd.

Alternatively, we can define a psd (resp., pd) matrix as an Hermitian matrix with nonnegative (resp., positive) eigenvalues. In particular, each psd matrix is diagonalizable, and the inverse of a pd matrix is also pd. The spectral norm of a psd matrix $\boldsymbol{M}$ has the variational characterization

$$\|\boldsymbol{M}\| = \max_{\boldsymbol{u}\neq\boldsymbol{0}} \frac{\boldsymbol{u}^*\boldsymbol{M}\boldsymbol{u}}{\boldsymbol{u}^*\boldsymbol{u}}, \tag{8.1}$$

according to the Rayleigh–Ritz theorem [72, Thm. 4.2.2]. It follows that

$$\boldsymbol{M} \preccurlyeq \boldsymbol{N} \quad\Longrightarrow\quad \|\boldsymbol{M}\| \leq \|\boldsymbol{N}\|. \tag{8.2}$$

A fundamental fact is that conjugation preserves the psd property.

LEMMA 8.1 (Conjugation Rule). *Suppose that* $\boldsymbol{M} \succcurlyeq \mathbf{0}$. *For every* $\boldsymbol{A}$, *the matrix* $\boldsymbol{A}^*\boldsymbol{M}\boldsymbol{A} \succcurlyeq \mathbf{0}$. *In particular,*

$$\boldsymbol{M} \preccurlyeq \boldsymbol{N} \quad \Longrightarrow \quad \boldsymbol{A}^*\boldsymbol{M}\boldsymbol{A} \preccurlyeq \boldsymbol{A}^*\boldsymbol{N}\boldsymbol{A}.$$

Our argument invokes the conjugation rule repeatedly. As a first application, we establish a perturbation bound for the matrix inverse near the identity matrix.

PROPOSITION 8.2 (Perturbation of Inverses). *Suppose that* $\boldsymbol{M} \succcurlyeq \mathbf{0}$. *Then*

$$\mathbf{I} - (\mathbf{I} + \boldsymbol{M})^{-1} \preccurlyeq \boldsymbol{M}$$

Proof. Define $\boldsymbol{R} = \boldsymbol{M}^{1/2}$, the psd square root of $\boldsymbol{M}$ promised by [72, Thm. 7.2.6]. We have the chain of relations

$$\mathbf{I} - (\mathbf{I} + \boldsymbol{R}^2)^{-1} = (\mathbf{I} + \boldsymbol{R}^2)^{-1}\boldsymbol{R}^2 = \boldsymbol{R}(\mathbf{I} + \boldsymbol{R}^2)^{-1}\boldsymbol{R} \preccurlyeq \boldsymbol{R}^2.$$

The first equality can be verified algebraically. The second holds because rational functions of a diagonalizable matrix, such as $\boldsymbol{R}$, commute. The last relation follows from the conjugation rule because $(\mathbf{I} + \boldsymbol{R}^2)^{-1} \preccurlyeq \mathbf{I}$. ∎

Next, we present a generalization of the fact that the spectral norm of a psd matrix is controlled by its trace.

PROPOSITION 8.3. *We have* $\|\boldsymbol{M}\| \le \|\boldsymbol{A}\| + \|\boldsymbol{C}\|$ *for each partitioned psd matrix*

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{B}^* & \boldsymbol{C} \end{bmatrix}.$$

Proof. The variational characterization (8.1) of the spectral norm implies that

$$\begin{aligned} \|\boldsymbol{M}\| &= \sup_{\|\boldsymbol{x}\|^2 + \|\boldsymbol{y}\|^2 = 1} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix}^* \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{B}^* & \boldsymbol{C} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} \\ &\le \sup_{\|\boldsymbol{x}\|^2 + \|\boldsymbol{y}\|^2 = 1} \big(\|\boldsymbol{A}\| \, \|\boldsymbol{x}\|^2 + 2 \, \|\boldsymbol{B}\| \, \|\boldsymbol{x}\| \, \|\boldsymbol{y}\| + \|\boldsymbol{C}\| \, \|\boldsymbol{y}\|^2 \big). \end{aligned}$$

The block generalization of Hadamard's psd criterion [72, Thm. 7.7.7] states that $\|\boldsymbol{B}\|^2 \le \|\boldsymbol{A}\| \, \|\boldsymbol{C}\|$. Thus,

$$\|\boldsymbol{M}\| \le \sup_{\|\boldsymbol{x}\|^2 + \|\boldsymbol{y}\|^2 = 1} \big(\|\boldsymbol{A}\|^{1/2} \, \|\boldsymbol{x}\| + \|\boldsymbol{C}\|^{1/2} \, \|\boldsymbol{y}\| \big)^2 = \|\boldsymbol{A}\| + \|\boldsymbol{C}\|.$$

This point completes the argument. ∎

8.2. Orthogonal projectors. An *orthogonal projector* is an Hermitian matrix $\boldsymbol{P}$ that satisfies the polynomial $\boldsymbol{P}^2 = \boldsymbol{P}$. This identity implies $\mathbf{0} \preccurlyeq \boldsymbol{P} \preccurlyeq \mathbf{I}$. An orthogonal projector is completely determined by its range. For a given matrix $\boldsymbol{M}$, we write $\boldsymbol{P}_{\boldsymbol{M}}$ for the unique orthogonal projector with $\operatorname{range}(\boldsymbol{P}_{\boldsymbol{M}}) = \operatorname{range}(\boldsymbol{M})$. When $\boldsymbol{M}$ has full column rank, we can express this projector explicitly:

$$\boldsymbol{P}_{\boldsymbol{M}} = \boldsymbol{M}(\boldsymbol{M}^*\boldsymbol{M})^{-1}\boldsymbol{M}^*. \tag{8.3}$$

The orthogonal projector onto the complementary subspace, $\operatorname{range}(\boldsymbol{P})^{\perp}$, is the matrix $\mathbf{I} - \boldsymbol{P}$. Our argument hinges on several other facts about orthogonal projectors.

PROPOSITION 8.4. *Suppose $\boldsymbol{U}$ is unitary. Then $\boldsymbol{U}^*\boldsymbol{P}_{\boldsymbol{M}}\boldsymbol{U} = \boldsymbol{P}_{\boldsymbol{U}^*\boldsymbol{M}}$.*

Proof. Abbreviate $\boldsymbol{P} = \boldsymbol{U}^*\boldsymbol{P}_{\boldsymbol{M}}\boldsymbol{U}$. It is clear that $\boldsymbol{P}$ is an orthogonal projector since it is Hermitian and $\boldsymbol{P}^2 = \boldsymbol{P}$. Evidently,

$$\operatorname{range}(\boldsymbol{P}) = \boldsymbol{U}^* \operatorname{range}(\boldsymbol{M}) = \operatorname{range}(\boldsymbol{U}^*\boldsymbol{M}).$$

Since the range determines the orthogonal projector, we conclude $\boldsymbol{P} = \boldsymbol{P}_{\boldsymbol{U}^*\boldsymbol{M}}$. ∎

PROPOSITION 8.5. *Suppose* $\operatorname{range}(\boldsymbol{N}) \subset \operatorname{range}(\boldsymbol{M})$. *Then, for each matrix $\boldsymbol{A}$, it holds that $\|\boldsymbol{P}_{\boldsymbol{N}}\boldsymbol{A}\| \le \|\boldsymbol{P}_{\boldsymbol{M}}\boldsymbol{A}\|$ and that $\|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{M}})\boldsymbol{A}\| \le \|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{N}})\boldsymbol{A}\|$.*

Proof. The projector $\boldsymbol{P}_{\boldsymbol{N}} \preccurlyeq \mathbf{I}$, so the conjugation rule yields $\boldsymbol{P}_{\boldsymbol{M}}\boldsymbol{P}_{\boldsymbol{N}}\boldsymbol{P}_{\boldsymbol{M}} \preccurlyeq \boldsymbol{P}_{\boldsymbol{M}}$. The hypothesis $\operatorname{range}(\boldsymbol{N}) \subset \operatorname{range}(\boldsymbol{M})$ implies that $\boldsymbol{P}_{\boldsymbol{M}}\boldsymbol{P}_{\boldsymbol{N}} = \boldsymbol{P}_{\boldsymbol{N}}$, which results in

$$\boldsymbol{P}_{\boldsymbol{M}}\boldsymbol{P}_{\boldsymbol{N}}\boldsymbol{P}_{\boldsymbol{M}} = \boldsymbol{P}_{\boldsymbol{N}}\boldsymbol{P}_{\boldsymbol{M}} = (\boldsymbol{P}_{\boldsymbol{M}}\boldsymbol{P}_{\boldsymbol{N}})^* = \boldsymbol{P}_{\boldsymbol{N}}.$$

In summary, $\boldsymbol{P}_{\boldsymbol{N}} \preccurlyeq \boldsymbol{P}_{\boldsymbol{M}}$. The conjugation rule shows that $\boldsymbol{A}^*\boldsymbol{P}_{\boldsymbol{N}}\boldsymbol{A} \preccurlyeq \boldsymbol{A}^*\boldsymbol{P}_{\boldsymbol{M}}\boldsymbol{A}$. We conclude from (8.2) that

$$\|\boldsymbol{P}_{\boldsymbol{N}}\boldsymbol{A}\|^2 = \|\boldsymbol{A}^*\boldsymbol{P}_{\boldsymbol{N}}\boldsymbol{A}\| \le \|\boldsymbol{A}^*\boldsymbol{P}_{\boldsymbol{M}}\boldsymbol{A}\| = \|\boldsymbol{P}_{\boldsymbol{M}}\boldsymbol{A}\|^2.$$

The second statement follows from the first by taking orthogonal complements. ∎

Finally, we need an unusual variation on the spectral radius formula.

PROPOSITION 8.6. *Let $\boldsymbol{P}$ be an orthogonal projector, and let $\boldsymbol{M}$ be a matrix. For each nonnegative q,*

$$\|\boldsymbol{P}\boldsymbol{M}\| \le \|\boldsymbol{P}(\boldsymbol{M}\boldsymbol{M}^*)^q\boldsymbol{M}\|^{1/(2q+1)}. \tag{8.4}$$

Proof. Suppose that $\boldsymbol{R}$ is an orthogonal projector, $\boldsymbol{D}$ is a nonnegative diagonal matrix, and $t \ge 1$. We claim that

$$\|\boldsymbol{R}\boldsymbol{D}\boldsymbol{R}\|^t \le \|\boldsymbol{R}\boldsymbol{D}^t\boldsymbol{R}\|. \tag{8.5}$$

Granted this inequality, we quickly complete the proof. Using an SVD $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$, we compute

$$\begin{aligned}\|\boldsymbol{P}\boldsymbol{M}\|^{2(2q+1)} &= \|\boldsymbol{P}\boldsymbol{M}\boldsymbol{M}^*\boldsymbol{P}\|^{2q+1} = \|(\boldsymbol{U}^*\boldsymbol{P}\boldsymbol{U})\cdot\boldsymbol{\Sigma}^2\cdot(\boldsymbol{U}^*\boldsymbol{P}\boldsymbol{U})\|^{2q+1} \\ &\le \|(\boldsymbol{U}^*\boldsymbol{P}\boldsymbol{U})\cdot\boldsymbol{\Sigma}^{2(2q+1)}\cdot(\boldsymbol{U}^*\boldsymbol{P}\boldsymbol{U})\| = \|\boldsymbol{P}(\boldsymbol{M}\boldsymbol{M}^*)^{2(2q+1)}\boldsymbol{P}\| \\ &= \|\boldsymbol{P}(\boldsymbol{M}\boldsymbol{M}^*)^q\boldsymbol{M}\cdot\boldsymbol{M}^*(\boldsymbol{M}\boldsymbol{M}^*)^q\boldsymbol{P}\| = \|\boldsymbol{P}(\boldsymbol{M}\boldsymbol{M}^*)^q\boldsymbol{M}\|^2.\end{aligned}$$

We have used the unitary invariance of the spectral norm in the second and fourth relation. The inequality (8.5) applies because $\boldsymbol{U}^*\boldsymbol{P}\boldsymbol{U}$ is an orthogonal projector. Take a square root to finish the argument.

Now, we turn to the claim (8.5). This relation follows immediately from [11, Thm. IX.2.10], but we offer a direct argument based on more elementary considerations. Let $\boldsymbol{x}$ be a unit vector at which

$$\boldsymbol{x}^*(\boldsymbol{R}\boldsymbol{D}\boldsymbol{R})\boldsymbol{x} = \|\boldsymbol{R}\boldsymbol{D}\boldsymbol{R}\|.$$

We must have $\boldsymbol{R}\boldsymbol{x} = \boldsymbol{x}$. Otherwise, $\|\boldsymbol{R}\boldsymbol{x}\| < 1$ because $\boldsymbol{R}$ is an orthogonal projector, which implies that the unit vector $\boldsymbol{y} = \boldsymbol{R}\boldsymbol{x} / \|\boldsymbol{R}\boldsymbol{x}\|$ verifies

$$\boldsymbol{y}^*(\boldsymbol{R}\boldsymbol{D}\boldsymbol{R})\boldsymbol{y} = \frac{(\boldsymbol{R}\boldsymbol{x})^*(\boldsymbol{R}\boldsymbol{D}\boldsymbol{R})(\boldsymbol{R}\boldsymbol{x})}{\|\boldsymbol{R}\boldsymbol{x}\|^2} = \frac{\boldsymbol{x}^*(\boldsymbol{R}\boldsymbol{D}\boldsymbol{R})\boldsymbol{x}}{\|\boldsymbol{R}\boldsymbol{x}\|^2} > \boldsymbol{x}^*(\boldsymbol{R}\boldsymbol{D}\boldsymbol{R})\boldsymbol{x}.$$

Writing x_j for the entries of $\boldsymbol{x}$ and d_j for the diagonal entries of $\boldsymbol{D}$, we find that

$$\begin{aligned}\|\boldsymbol{R}\boldsymbol{D}\boldsymbol{R}\|^t = [\boldsymbol{x}^*(\boldsymbol{R}\boldsymbol{D}\boldsymbol{R})\boldsymbol{x}]^t = [\boldsymbol{x}^*\boldsymbol{D}\boldsymbol{x}]^t &= \left[\sum\nolimits_j d_j x_j^2\right]^t \\ &\le \left[\sum\nolimits_j d_j^t x_j^2\right] = \boldsymbol{x}^*\boldsymbol{D}^t\boldsymbol{x} = (\boldsymbol{R}\boldsymbol{x})^*\boldsymbol{D}^t(\boldsymbol{R}\boldsymbol{x}) \le \left\|\boldsymbol{R}\boldsymbol{D}^t\boldsymbol{R}\right\|.\end{aligned}$$

The inequality is Jensen's, which applies because $\sum x_j^2 = 1$ and the function $z \mapsto |z|^t$ is convex for $t \ge 1$. □

9. Error bounds via linear algebra. We are now prepared to develop a deterministic error analysis for the proto-algorithm described in §1.3. To begin, we must introduce some notation. Afterward, we establish the key error bound, which strengthens a result from the literature [17, Lem. 2]. Finally, we explain why the power method can be used to improve the performance of the proto-algorithm.

9.1. Setup. Let $\boldsymbol{A}$ be an $m \times n$ matrix with singular value decomposition $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$, as described in Section 3.2.2. The algorithms described construct approximations to the space spanned by the first k left singular vectors, where k for now is a given number. To analyze the situation, we first partition the singular value decomposition of $\boldsymbol{A}$ as follows:

$$\boldsymbol{A} = \boldsymbol{U} \begin{array}{c} \begin{matrix} k & n-k \end{matrix} \\ \begin{bmatrix} \boldsymbol{\Sigma}_1 & \\ & \boldsymbol{\Sigma}_2 \end{bmatrix} \end{array} \begin{array}{c} \begin{matrix} \\ \end{matrix} \\ \begin{bmatrix} \boldsymbol{V}_1^* \\ \boldsymbol{V}_2^* \end{bmatrix} \end{array} \begin{array}{c} \begin{matrix} k \\ n-k \end{matrix} \end{array} \tag{9.1}$$

Let $\boldsymbol{\Omega}$ be an $n \times \ell$ test matrix, where we assume only that $\ell \ge k$. Decompose the test matrix in the coordinate system determined by the right unitary factor of $\boldsymbol{A}$:

$$\boldsymbol{\Omega}_1 = \boldsymbol{V}_1^*\,\boldsymbol{\Omega} \quad \text{and} \quad \boldsymbol{\Omega}_2 = \boldsymbol{V}_2^*\,\boldsymbol{\Omega}. \tag{9.2}$$

The singular spectra of $\boldsymbol{\Omega}_1$ and $\boldsymbol{\Omega}_2$ play a critical role in the analysis of the algorithm. Indeed, we ultimately set $\ell = k + p$ because we need to improve the conditioning of the block $\boldsymbol{\Omega}_1$. With this notation, the sample matrix $\boldsymbol{Y}$ can be expressed as

$$\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{\Omega} = \boldsymbol{U} \begin{array}{c} \begin{matrix} \ell \end{matrix} \\ \begin{bmatrix} \boldsymbol{\Sigma}_1\boldsymbol{\Omega}_1 \\ \boldsymbol{\Sigma}_2\boldsymbol{\Omega}_2 \end{bmatrix} \end{array} \begin{array}{c} \begin{matrix} k \\ n-k \end{matrix} \end{array} \tag{9.3}$$

Intuitively, the top block reflects the gross behavior of $\boldsymbol{A}$, while the second block represents a perturbation.

9.2. A deterministic error bound for the proto-algorithm. The proto-algorithm constructs an orthonormal basis $\boldsymbol{Q}$ for the range of the sample matrix $\boldsymbol{Y}$, and our goal is to quantify how well this basis captures the action of the input $\boldsymbol{A}$. Since $\boldsymbol{Q}\boldsymbol{Q}^* = \boldsymbol{P}_{\boldsymbol{Y}}$, the challenge is to obtain bounds on the approximation error

$$\left\|\left\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\right\|\right\| = \left\|\left\|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Y}})\boldsymbol{A}\right\|\right\|.$$

The following theorem shows that the behavior of the proto-algorithm depends on the interaction between the test matrix and the right singular vectors of the input matrix, as well as the singular spectrum of the input matrix.

THEOREM 9.1 (Deterministic error bound). *Let $\boldsymbol{A}$ be an $m \times n$ matrix with singular value decomposition $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$, and fix $k \geq 0$. Choose a test matrix $\boldsymbol{\Omega}$, and construct the sample matrix $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{\Omega}$. Partition $\boldsymbol{\Sigma}$ as specified in* (9.1)*, and define $\boldsymbol{\Omega}_1$ and $\boldsymbol{\Omega}_2$ via* (9.2)*. Assuming that $\boldsymbol{\Omega}_1$ has full row rank, the approximation error satisfies*

$$\left\|\!\left|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Y}})\boldsymbol{A}\right|\!\right\|^2 \leq \left\|\!\left|\boldsymbol{\Sigma}_2\right|\!\right\|^2 + \left\|\!\left|\boldsymbol{\Sigma}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^{\dagger}\right|\!\right\|^2, \tag{9.4}$$

where $\|\!|\cdot|\!\|$ denotes either the spectral norm or the Frobenius norm.

Theorem 9.1 sharpens [17, Lem. 2] by means of a different argument. Our proof strategy is inspired by the perturbation theory of orthogonal projectors [124].

Proof. We establish the bound for the spectral-norm error. The bound for the Frobenius-norm error follows from an analogous argument that is slightly easier.

As a preliminary step, we check that $\boldsymbol{U}$ plays no essential role in the argument. In effect, we execute the proof for the auxiliary matrix

$$\boldsymbol{A}_0 = \boldsymbol{U}^*\boldsymbol{A} = \boldsymbol{\Sigma}\boldsymbol{V}^*.$$

Owing to the unitary invariance of the spectral norm and to Proposition 8.4, we have the identity

$$\|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Y}})\boldsymbol{A}\| = \|\boldsymbol{U}^*(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Y}})\boldsymbol{U}\boldsymbol{A}_0\| = \|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{U}^*\boldsymbol{Y}})\boldsymbol{A}_0\|. \tag{9.5}$$

In words, we need to determine how well the range of $\boldsymbol{U}^*\boldsymbol{Y}$ explains $\boldsymbol{A}_0$.

The key idea is to identify a matrix with full column rank whose range lies within the range of $\boldsymbol{U}^*\boldsymbol{Y}$. This step allows us to invoke (8.3), which yields an analytic expression for the orthogonal projector. In consideration of (9.3), set

$$\boldsymbol{Z} = (\boldsymbol{U}^*\boldsymbol{Y})\boldsymbol{\Omega}_1^{\dagger}\boldsymbol{\Sigma}_1^{-1},$$

where we may assume that $\boldsymbol{\Sigma}_1$ is invertible by a perturbation argument. This construction ensures that $\operatorname{range}(\boldsymbol{Z}) \subset \operatorname{range}(\boldsymbol{U}^*\boldsymbol{Y})$. Proposition 8.5 implies

$$\|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{U}^*\boldsymbol{Y}})\boldsymbol{A}_0\| \leq \|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Z}})\boldsymbol{A}_0\|.$$

That is, we can better approximate the matrix $\boldsymbol{A}_0$ in the range of $\boldsymbol{U}^*\boldsymbol{Y}$ than in the range of $\boldsymbol{Z}$. In view of (9.5), it holds that

$$\|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Y}})\boldsymbol{A}\|^2 \leq \|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Z}})\boldsymbol{A}_0\|^2 = \|\boldsymbol{A}_0^*(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Z}})\boldsymbol{A}_0\| = \|\boldsymbol{\Sigma}^*(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Z}})\boldsymbol{\Sigma}\|, \tag{9.6}$$

where the last identity follows from the unitary invariance of the spectral norm. Our goal now is to bound the right-hand side of (9.6).

To continue, we need a detailed expression for the projector $\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Z}}$. Referring back to (9.3) and invoking the assumption that $\boldsymbol{\Omega}_1$ has full row rank, we obtain

$$\boldsymbol{Z} = \begin{bmatrix} \boldsymbol{\Sigma}_1\boldsymbol{\Omega}_1 \\ \boldsymbol{\Sigma}_2\boldsymbol{\Omega}_2 \end{bmatrix} \boldsymbol{\Omega}_1^{\dagger}\boldsymbol{\Sigma}_1^{-1} = \begin{bmatrix} \mathbf{I} \\ \boldsymbol{F} \end{bmatrix},$$

where $\boldsymbol{F} = \boldsymbol{\Sigma}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\boldsymbol{\Sigma}_1^{-1}$ and $\mathbf{I}$ represents a conformal identity matrix, here with dimension $k \times k$. In particular, this expression shows that $\boldsymbol{Z}$ has full column rank, so we can apply (8.3) to see that the projector

$$\boldsymbol{P}_{\boldsymbol{Z}} = \boldsymbol{Z}(\boldsymbol{Z}^*\boldsymbol{Z})^{-1}\boldsymbol{Z}^*.$$

Expanding this formula, we determine that the complementary projector satisfies

$$\begin{aligned}\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Z}} &= \mathbf{I} - \begin{bmatrix}\mathbf{I}\\ \boldsymbol{F}\end{bmatrix}(\mathbf{I} + \boldsymbol{F}^*\boldsymbol{F})^{-1}\begin{bmatrix}\mathbf{I}\\ \boldsymbol{F}\end{bmatrix}^* \\ &= \begin{bmatrix}\mathbf{I} - (\mathbf{I} + \boldsymbol{F}^*\boldsymbol{F})^{-1} & -(\mathbf{I} + \boldsymbol{F}^*\boldsymbol{F})^{-1}\boldsymbol{F}^* \\ -\boldsymbol{F}(\mathbf{I} + \boldsymbol{F}^*\boldsymbol{F})^{-1} & \mathbf{I} - \boldsymbol{F}(\mathbf{I} + \boldsymbol{F}^*\boldsymbol{F})^{-1}\boldsymbol{F}^*\end{bmatrix}. \end{aligned} \tag{9.7}$$

The partitioning here is conformal with the partitioning of $\boldsymbol{\Sigma}$. As a result, when we conjugate the matrix by $\boldsymbol{\Sigma}$, copies of $\boldsymbol{\Sigma}_1^{-1}$, presently hidden in the top-left block, will cancel to happy effect.

The latter point may not seem obvious, owing to the complicated form of (9.7). In reality, the block matrix is less fearsome than it looks. Proposition 8.2, on the perturbation of inverses, shows that the top-left block verifies

$$\mathbf{I} - (\mathbf{I} + \boldsymbol{F}^*\boldsymbol{F})^{-1} \preccurlyeq \boldsymbol{F}^*\boldsymbol{F}.$$

The bottom-right block satisfies

$$\mathbf{I} - \boldsymbol{F}(\mathbf{I} + \boldsymbol{F}^*\boldsymbol{F})^{-1}\boldsymbol{F}^* \preccurlyeq \mathbf{I},$$

because the conjugation rule guarantees that $\boldsymbol{F}(\mathbf{I} + \boldsymbol{F}^*\boldsymbol{F})^{-1}\boldsymbol{F}^* \succcurlyeq \mathbf{0}$. We abbreviate the off-diagonal blocks with the symbol $\boldsymbol{B} = -(\mathbf{I} + \boldsymbol{F}^*\boldsymbol{F})^{-1}\boldsymbol{F}^*$. In summary,

$$\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Z}} \preccurlyeq \begin{bmatrix}\boldsymbol{F}^*\boldsymbol{F} & \boldsymbol{B}\\ \boldsymbol{B}^* & \mathbf{I}\end{bmatrix}.$$

This relation exposes the key structural properties of the projector.

Moving toward the final estimate, we conjugate the last inequality by $\boldsymbol{\Sigma}$ to obtain

$$\boldsymbol{\Sigma}^*(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Z}})\boldsymbol{\Sigma} \preccurlyeq \begin{bmatrix}\boldsymbol{\Sigma}_1^*\boldsymbol{F}^*\boldsymbol{F}\boldsymbol{\Sigma}_1 & \boldsymbol{\Sigma}_1^*\boldsymbol{B}\boldsymbol{\Sigma}_2\\ \boldsymbol{\Sigma}_2^*\boldsymbol{B}^*\boldsymbol{\Sigma}_1 & \boldsymbol{\Sigma}_2^*\boldsymbol{\Sigma}_2\end{bmatrix}.$$

The conjugation rule demonstrates that the matrix on the left-hand side is psd, so the matrix on the right-hand side is too. Proposition 8.3 results in the norm bound

$$\|\boldsymbol{\Sigma}^*(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Z}})\boldsymbol{\Sigma}\| \le \|\boldsymbol{\Sigma}_1^*\boldsymbol{F}^*\boldsymbol{F}\boldsymbol{\Sigma}_1\| + \|\boldsymbol{\Sigma}_2^*\boldsymbol{\Sigma}_2\| = \|\boldsymbol{F}\boldsymbol{\Sigma}_1\|^2 + \|\boldsymbol{\Sigma}_2\|^2.$$

Recall that $\boldsymbol{F} = \boldsymbol{\Sigma}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\boldsymbol{\Sigma}_1^{-1}$, so the factor $\boldsymbol{\Sigma}_1$ cancels neatly. Therefore,

$$\|\boldsymbol{\Sigma}^*(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Z}})\boldsymbol{\Sigma}\| \le \left\|\boldsymbol{\Sigma}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\right\|^2 + \|\boldsymbol{\Sigma}_2\|^2.$$

Looking back to (9.6), we discover that the proof is complete. □

9.3. Analysis of the power scheme. Theorem 9.1 suggests that the performance of the proto-algorithm depends strongly on the relationship between the large singular values of $\boldsymbol{A}$ listed in $\boldsymbol{\Sigma}_1$ and the small singular values listed in $\boldsymbol{\Sigma}_2$. When a substantial proportion of the mass of $\boldsymbol{A}$ appears in the small singular values, the constructed basis $\boldsymbol{Q}$ may have low accuracy. Conversely, when the large singular values dominate, it is much easier to identify a good low-rank basis.

To improve the performance of the proto-algorithm, we can run it with a closely related input matrix whose singular values decay more rapidly [67, 111]. Fix a nonnegative integer q, and set

$$\boldsymbol{B} = (\boldsymbol{A}\boldsymbol{A}^*)^q \boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}^{2q+1}\boldsymbol{V}^*.$$

We apply the proto-algorithm to $\boldsymbol{B}$, which generates a sample matrix $\boldsymbol{Z} = \boldsymbol{B}\boldsymbol{\Omega}$ and constructs a basis $\boldsymbol{Q}$ for the range of $\boldsymbol{Z}$. Section 4.5 elaborates on the implementation details. The following result describes how well we can approximate the *original* matrix $\boldsymbol{A}$ within the range of $\boldsymbol{Z}$.

THEOREM 9.2 (Power scheme). *Let $\boldsymbol{A}$ be an $m \times n$ matrix, and let $\boldsymbol{\Omega}$ be an $n \times \ell$ matrix. Fix a nonnegative integer q, form $\boldsymbol{B} = (\boldsymbol{A}^*\boldsymbol{A})^q\boldsymbol{A}$, and compute the sample matrix $\boldsymbol{Z} = \boldsymbol{B}\boldsymbol{\Omega}$. Then*

$$\|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Z}})\boldsymbol{A}\| \le \|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Z}})\boldsymbol{B}\|^{1/(2q+1)}.$$

Proof. We can estimate that

$$\|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Z}})\boldsymbol{A}\| \le \|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Z}})(\boldsymbol{A}\boldsymbol{A}^*)^q\boldsymbol{A}\|^{1/(2q+1)} = \|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Z}})\boldsymbol{B}\|^{1/(2q+1)}$$

as a simple consequence of Proposition 8.6. ∎

Let us illustrate how the power scheme interacts with the main error bound (9.4). Fix a matrix $\boldsymbol{A}$, and let σ_{k+1} denote its $(k+1)$th singular value. First, suppose we approximate $\boldsymbol{A}$ in the range of the sample matrix $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{\Omega}$. Since $\|\boldsymbol{\Sigma}_2\| = \sigma_{k+1}$, Theorem 9.1 implies that

$$\|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Y}})\boldsymbol{A}\| \le \left(1 + \left\|\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\right\|^2\right)^{1/2} \sigma_{k+1}. \tag{9.8}$$

Now, define $\boldsymbol{B} = (\boldsymbol{A}\boldsymbol{A}^*)^q\boldsymbol{A}$, and suppose that we approximate $\boldsymbol{A}$ within the range of the sample matrix $\boldsymbol{Z} = \boldsymbol{B}\boldsymbol{\Omega}$. Together, Theorem 9.2 and Theorem 9.1 imply that

$$\|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Z}})\boldsymbol{A}\| \le \|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Z}})\boldsymbol{B}\|^{1/(4q+2)} \le \left(1 + \left\|\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\right\|^2\right)^{1/(4q+2)} \sigma_{k+1}$$

because σ_{k+1}^{2q+1} is the $(k+1)$th singular value of $\boldsymbol{B}$. In effect, the power scheme drives down the suboptimality of the bound (9.8) exponentially fast as the power q increases. In principle, we can make the extra factor as close to one as we like, although this increases the cost of the algorithm.

REMARK 9.1. When we apply the power scheme numerically, we obtain additional Krylov information that can be exploited to improve the numerical accuracy of the orthogonalization step. The resulting procedure appears as Algorithm 4.3. Disregarding numerical effects, we can develop an error bound for this scheme as an immediate corollary of Theorem 9.2. For a rounding-error analysis, see [111].

COROLLARY 9.3 (Extended power scheme). *Instate the hypotheses and notation of Theorem 9.2. Define the matrices* $\boldsymbol{Y}^{(i)} = (\boldsymbol{A}\boldsymbol{A}^*)^i\boldsymbol{A}\boldsymbol{\Omega}$ *for* $i = 0, 1, \dots, q$, *and construct the extended sample matrix*

$$\boldsymbol{W} = \left[\boldsymbol{Y}^{(0)}, \boldsymbol{Y}^{(1)}, \dots, \boldsymbol{Y}^{(q)}\right]. \tag{9.9}$$

Then

$$\|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{W}})\boldsymbol{A}\| \leq \|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Z}})\boldsymbol{B}\|^{1/(2q+1)}.$$

Proof. Observe that range($\boldsymbol{Z}$) $\subset$ range($\boldsymbol{W}$). Then invoke Proposition 8.5 and Theorem 9.2. □

10. Gaussian test matrices. The error bound in Theorem 9.1 shows that the performance of the proto-algorithm depends on the interaction between the test matrix $\boldsymbol{\Omega}$ and the right singular vectors of the input matrix $\boldsymbol{A}$. Algorithm 4.1 is a particularly simple version of the proto-algorithm that draws the test matrix according to the standard Gaussian distribution. The literature contains a wealth of information about these matrices, which results in a sharp error analysis.

We focus on the real case in this section. Analogous results hold in the complex case, where the algorithm even exhibits superior performance.

10.1. Technical background. A *standard Gaussian matrix* is a random matrix whose entries are independent standard normal variables. The distribution of a standard Gaussian matrix is rotationally invariant: If $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal matrices, then $\boldsymbol{U}^*\boldsymbol{G}\boldsymbol{V}$ also has the standard Gaussian distribution.

Our analysis requires detailed information about the properties of Gaussian matrices. In particular, we must understand how the norm of a Gaussian matrix and its pseudoinverse vary. We summarize the relevant results and citations here, reserving the details for Appendix A.

PROPOSITION 10.1 (Expected norm of a scaled Gaussian matrix). *Fix matrices* $\boldsymbol{S}, \boldsymbol{T}$, *and draw a standard Gaussian matrix* $\boldsymbol{G}$. *Then*

$$\left(\mathbb{E}\,\|\boldsymbol{S}\boldsymbol{G}\boldsymbol{T}^*\|_{\mathrm{F}}^2\right)^{1/2} = \|\boldsymbol{S}\|_{\mathrm{F}}\,\|\boldsymbol{T}\|_{\mathrm{F}} \tag{10.1}$$

$$\mathbb{E}\,\|\boldsymbol{S}\boldsymbol{G}\boldsymbol{T}^*\| \leq \|\boldsymbol{S}\|\,\|\boldsymbol{T}\|_{\mathrm{F}} + \|\boldsymbol{S}\|_{\mathrm{F}}\,\|\boldsymbol{T}\|. \tag{10.2}$$

The identity (10.1) follows from a direct calculation. The second bound (10.2) relies on methods developed by Gordon [62, 63]. See Propositions A.1 and A.2.

PROPOSITION 10.2 (Expected norm of a pseudo-inverted Gaussian matrix). *Draw a* $k \times (k+p)$ *standard Gaussian matrix* $\boldsymbol{G}$ *with* $p \geq 2$. *Then*

$$\left(\mathbb{E}\left\|\boldsymbol{G}^\dagger\right\|_{\mathrm{F}}^2\right)^{1/2} = \sqrt{\frac{k}{p-1}} \tag{10.3}$$

$$\mathbb{E}\left\|\boldsymbol{G}^\dagger\right\| \leq \frac{\mathrm{e}\sqrt{k+p}}{p}. \tag{10.4}$$

The first identity is a standard result from multivariate statistics [100, p. 96]. The second follows from work of Chen and Dongarra [28]. See Proposition A.4 and A.5.

To study the probability that Algorithm 4.1 produces a large error, we rely on tail bounds for functions of Gaussian matrices. The next proposition rephrases a well-known result on concentration of measure [14, Thm. 4.5.7]. See also [85, §1.1] and [84, §5.1].

PROPOSITION 10.3 (Concentration for functions of a Gaussian matrix). *Suppose that h is a Lipschitz function on matrices:*

$$|h(\boldsymbol{X}) - h(\boldsymbol{Y})| \le L \|\boldsymbol{X} - \boldsymbol{Y}\|_{\mathrm{F}} \quad \text{for all } \boldsymbol{X}, \boldsymbol{Y}.$$

Draw a standard Gaussian matrix $\boldsymbol{G}$. Then

$$\mathbb{P}\{h(\boldsymbol{G}) \ge \mathbb{E}\, h(\boldsymbol{G}) + Lt\} \le \mathrm{e}^{-t^2/2}.$$

Finally, we state some large deviation bounds for the norm of a pseudo-inverted Gaussian matrix.

PROPOSITION 10.4 (Norm bounds for a pseudo-inverted Gaussian matrix). *Let $\boldsymbol{G}$ be a $k \times (k+p)$ Gaussian matrix where $p \ge 4$. For all $t \ge 1$,*

$$\mathbb{P}\left\{ \left\|\boldsymbol{G}^{\dagger}\right\|_{\mathrm{F}} \ge \sqrt{\frac{12k}{p}} \cdot t \right\} \le 4t^{-p}, \quad \text{and} \tag{10.5}$$

$$\mathbb{P}\left\{ \left\|\boldsymbol{G}^{\dagger}\right\| \ge \frac{\mathrm{e}\sqrt{k+p}}{p+1} \cdot t \right\} \le t^{-(p+1)}. \tag{10.6}$$

Compare these estimates with Proposition 10.2. It seems that (10.5) is new; we were unable to find a comparable analysis in the random matrix literature. Although the form of (10.5) is not optimal, it allows us to produce more transparent results than a fully detailed estimate. The bound (10.6) essentially appears in the work of Chen and Dongarra [28]. See Propositions A.3 and Theorem A.6 for more information.

10.2. Average-case analysis of Algorithm 4.1. We separate our analysis into two pieces. First, we present information about expected values. Afterward, we provide bounds on the likelihood of a large deviation.

We begin with the simplest result, which provides an estimate for the expected approximation error in the Frobenius norm. All proofs are postponed to the end of the section.

THEOREM 10.5 (Average Frobenius error). *Suppose that $\boldsymbol{A}$ is a* real *$m \times n$ matrix with singular values $\sigma_1 \ge \sigma_2 \ge \sigma_3 \ge \dots$. Choose a target rank k and an oversampling parameter $p \ge 2$, where $k + p \le \min\{m, n\}$. Draw an $n \times (k+p)$ standard Gaussian matrix $\boldsymbol{\Omega}$, and construct the sample matrix $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{\Omega}$. Then the expected approximation error*

$$\mathbb{E}\, \|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Y}})\boldsymbol{A}\|_{\mathrm{F}} \le \left(1 + \frac{k}{p-1}\right)^{1/2} \left(\sum\nolimits_{j>k} \sigma_j^2\right)^{1/2}.$$

This theorem encapsulates several distinct—and intriguing—behaviors of Algorithm 4.1. The Eckart–Young theorem [53] shows that $(\sum_{j>k} \sigma_j^2)^{1/2}$ is the minimal Frobenius-norm error when approximating $\boldsymbol{A}$ with a rank-k matrix. This quantity is the appropriate benchmark for the performance of the algorithm. If the small singular values of $\boldsymbol{A}$ are very flat, the series may be as large as $\sigma_{k+1}\sqrt{\min\{m,n\}-k}$. On the other hand, when the singular values exhibit some decay, the error may be on the same order as σ_{k+1}.

The error bound always exceeds this baseline error, but it may be polynomially larger, depending on the ratio between the target rank k and the oversampling parameter p. For p small (say, less than five), the error is actually quite variable because the small singular values of a nearly square Gaussian matrix are very unstable. As the oversampling increases, the performance improves quickly. When $p \sim k$, the error is already within a constant factor of the baseline.

The error bound for the spectral norm is somewhat more complicated, but it reveals some interesting new features.

THEOREM 10.6 (Average spectral error). *Under the hypotheses of Theorem 10.5,*

$$\mathbb{E}\,\|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Y}})\boldsymbol{A}\| \leq \left(1 + \sqrt{\frac{k}{p-1}}\right)\sigma_{k+1} + \frac{\mathrm{e}\sqrt{k+p}}{p}\left(\sum\nolimits_{j>k}\sigma_j^2\right)^{1/2}.$$

Mirsky [99] has shown that the quantity σ_{k+1} is the minimum spectral-norm error when approximating $\boldsymbol{A}$ with a rank-k matrix, so the first term in Theorem 10.6 is analogous with the error bound in Theorem 10.5. The second term represents a new phenomenon: we also pay for the Frobenius-norm error in approximating $\boldsymbol{A}$. Note that, as the amount p of oversampling increases, the polynomial factor in the second term declines much more quickly than the factor in the first term. Consider what happens when $p \sim k$.

Finally, we remark that the bound in Theorem 10.6 implies

$$\mathbb{E}\,\|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Y}})\boldsymbol{A}\| \leq \left[1 + \sqrt{\frac{k}{p-1}} + \frac{\mathrm{e}\sqrt{k+p}}{p}\cdot\sqrt{\min\{m,n\}-k}\right]\sigma_{k+1},$$

so the average spectral-norm error always lies within a small polynomial factor of the baseline σ_{k+1}.

We continue with the proofs of these results.

Proof. [Theorem 10.5] Let $\boldsymbol{V}$ be the right unitary factor of $\boldsymbol{A}$. Partition $\boldsymbol{V} = [\boldsymbol{V}_1 \mid \boldsymbol{V}_2]$ into blocks containing, respectively, k and $n-k$ columns. Recall that

$$\boldsymbol{\Omega}_1 = \boldsymbol{V}_1^*\boldsymbol{\Omega} \quad \text{and} \quad \boldsymbol{\Omega}_2 = \boldsymbol{V}_2^*\boldsymbol{\Omega}.$$

The Gaussian distribution is rotationally invariant, so $\boldsymbol{V}^*\boldsymbol{\Omega}$ is also a standard Gaussian matrix. Observe that $\boldsymbol{\Omega}_1$ and $\boldsymbol{\Omega}_2$ are *disjoint* submatrices of $\boldsymbol{V}^*\boldsymbol{\Omega}$, so these two matrices are not only standard Gaussian but also stochastically independent. Furthermore, the rows of a (fat) Gaussian matrix are almost surely in general position, so the $k \times (k+p)$ matrix $\boldsymbol{\Omega}_1$ has full row rank with probability one.

Hölder's inequality and Theorem 9.1 together imply that

$$\mathbb{E}\,\|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Y}})\boldsymbol{A}\|_{\mathrm{F}} \leq \left(\mathbb{E}\,\|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Y}})\boldsymbol{A}\|_{\mathrm{F}}^2\right)^{1/2} \leq \left(\left\|\boldsymbol{\Sigma}_2\right\|_{\mathrm{F}}^2 + \mathbb{E}\left\|\boldsymbol{\Sigma}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^{\dagger}\right\|_{\mathrm{F}}^2\right)^{1/2}.$$

We compute this expectation by conditioning on the value of $\boldsymbol{\Omega}_1$ and applying Proposition 10.1 to the scaled Gaussian matrix $\boldsymbol{\Omega}_2$. Thus,

$$\begin{aligned}\mathbb{E}\left\|\boldsymbol{\Sigma}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\right\|_{\mathrm{F}}^2 = \mathbb{E}\left(\mathbb{E}\left[\left\|\boldsymbol{\Sigma}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\right\|_{\mathrm{F}}^2 \mid \boldsymbol{\Omega}_1\right]\right) &= \mathbb{E}\left(\|\boldsymbol{\Sigma}_2\|_{\mathrm{F}}^2\left\|\boldsymbol{\Omega}_1^\dagger\right\|_{\mathrm{F}}^2\right)\\ &= \|\boldsymbol{\Sigma}_2\|_{\mathrm{F}}^2 \cdot \mathbb{E}\left\|\boldsymbol{\Omega}_1^\dagger\right\|_{\mathrm{F}}^2 = \frac{k}{p-1}\cdot\|\boldsymbol{\Sigma}_2\|_{\mathrm{F}}^2,\end{aligned}$$

where the last expectation follows from relation (10.3) of Proposition 10.2. In summary,

$$\mathbb{E}\,\|(\mathbf{I}-\boldsymbol{P}_{\boldsymbol{Y}})\boldsymbol{A}\|_{\mathrm{F}} \le \left(1+\frac{k}{p-1}\right)^{1/2}\|\boldsymbol{\Sigma}_2\|_{\mathrm{F}}.$$

Observe that $\|\boldsymbol{\Sigma}_2\|_{\mathrm{F}}^2 = \sum_{j>k}\sigma_j^2$ to complete the proof. □

Proof. [Theorem 10.6] The argument is similar the proof of Theorem 10.5. First, Theorem 9.1 and the (deterministic) Hölder inequality imply that

$$\mathbb{E}\,\|(\mathbf{I}-\boldsymbol{P}_{\boldsymbol{Y}})\boldsymbol{A}\| \le \mathbb{E}\left(\|\boldsymbol{\Sigma}_2\|^2 + \left\|\boldsymbol{\Sigma}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\right\|^2\right)^{1/2} \le \|\boldsymbol{\Sigma}_2\| + \mathbb{E}\left\|\boldsymbol{\Sigma}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\right\|.$$

We condition on $\boldsymbol{\Omega}_1$ and apply Proposition 10.1 to bound the expectation with respect to $\boldsymbol{\Omega}_2$. Thus,

$$\begin{aligned}\mathbb{E}\left\|\boldsymbol{\Sigma}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\right\| &\le \mathbb{E}\left(\|\boldsymbol{\Sigma}_2\|\left\|\boldsymbol{\Omega}_1^\dagger\right\|_{\mathrm{F}} + \|\boldsymbol{\Sigma}_2\|_{\mathrm{F}}\left\|\boldsymbol{\Omega}_1^\dagger\right\|\right)\\ &\le \|\boldsymbol{\Sigma}_2\|\left(\mathbb{E}\left\|\boldsymbol{\Omega}_1^\dagger\right\|_{\mathrm{F}}^2\right)^{1/2} + \|\boldsymbol{\Sigma}_2\|_{\mathrm{F}}\cdot\mathbb{E}\left\|\boldsymbol{\Omega}_1^\dagger\right\|.\end{aligned}$$

where the second relation requires Hölder's inequality. Applying both parts of Proposition 10.2, we obtain

$$\mathbb{E}\left\|\boldsymbol{\Sigma}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\right\| \le \sqrt{\frac{k}{p-1}}\,\|\boldsymbol{\Sigma}_2\| + \frac{\mathrm{e}\sqrt{k+p}}{p}\,\|\boldsymbol{\Sigma}_2\|_{\mathrm{F}}.$$

Note that $\|\boldsymbol{\Sigma}_2\| = \sigma_{k+1}$ to wrap up. □

10.3. Probabilistic error bounds for Algorithm 4.1. We can develop tail bounds for the approximation error, which demonstrate that the average performance of the algorithm is representative of the actual performance. We begin with the Frobenius norm because the result is somewhat simpler.

THEOREM 10.7 (Deviation bounds for the Frobenius error). *Frame the hypotheses of Theorem 10.5. Assume further that $p \ge 4$. For all $u, t \ge 1$,*

$$\|(\mathbf{I}-\boldsymbol{P}_{\boldsymbol{Y}})\boldsymbol{A}\|_{\mathrm{F}} \le \left(1+t\cdot\sqrt{12k/p}\right)\left(\sum\nolimits_{j>k}\sigma_j^2\right)^{1/2} + ut\cdot\frac{\mathrm{e}\sqrt{k+p}}{p+1}\cdot\sigma_{k+1},$$

except with probability $5t^{-p} + 2\mathrm{e}^{-u^2/2}$.

To parse this theorem, observe that the first term in the error bound corresponds with the expected approximation error in Theorem 10.5. The second term represents a deviation above the mean. In general, the mean dominates the size of a deviation,

but the deviations can be substantially smaller when the oversampling parameter p is large or the singular values of the input matrix decay slowly.

An analogous result holds for the spectral norm.

THEOREM 10.8 (Deviation bounds for the spectral error). *Frame the hypotheses of Theorem 10.5. Assume further that $p \geq 4$. For all $u, t \geq 1$,*

$$\begin{aligned}&\|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Y}})\boldsymbol{A}\| \\ &\quad \leq \left[\left(1 + t \cdot \sqrt{12k/p}\right)\sigma_{k+1} + t \cdot \frac{\mathrm{e}\sqrt{k+p}}{p+1}\left(\sum\nolimits_{j>k}\sigma_j^2\right)^{1/2}\right] + ut \cdot \frac{\mathrm{e}\sqrt{k+p}}{p+1}\sigma_{k+1},\end{aligned}$$

except with probability $5t^{-p} + \mathrm{e}^{-u^2/2}$.

The bracket corresponds with the expected spectral-norm error while the remaining term represents a deviation above the mean. Neither the numerical constants nor the precise form of the bound are optimal because of the slackness in Proposition 10.4. Nevertheless, the theorem gives a fairly good picture of what is actually happening.

We acknowledge that the current form of Theorem 10.8 is complicated. To produce more transparent results, we make appropriate selections for the parameters u, t and compute the numerical constants.

COROLLARY 10.9 (Simplified deviation bounds for the spectral error). *Frame the hypotheses of Theorem 10.5, and assume further that $p \geq 4$. Then*

$$\|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Y}})\boldsymbol{A}\| \leq \left(1 + 17\sqrt{1 + k/p}\right)\sigma_{k+1} + \frac{8\sqrt{k+p}}{p+1}\left(\sum\nolimits_{j>k}\sigma_j^2\right)^{1/2},$$

except with probability $6\mathrm{e}^{-p}$. Moreover,

$$\|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Y}})\boldsymbol{A}\| \leq \left(1 + 8\sqrt{(k+p)\cdot p\log p}\right)\sigma_{k+1} + 3\sqrt{k+p}\left(\sum\nolimits_{j>k}\sigma_j^2\right)^{1/2},$$

except with probability $6p^{-p}$.

Proof. The first part of the result follows from the choices $t = \mathrm{e}$ and $u = \sqrt{2p}$, and the second emerges when $t = p$ and $u = \sqrt{2p\log p}$. Another interesting parameter selection is $t = p^{c/p}$ and $u = \sqrt{2c\log p}$, which yields a failure probability $6p^{-c}$. ∎

Corollary 10.9 should be compared with [94, Obs. 4.4–4.5]. Although our result contains sharper error estimates, the failure probabilities are usually worse.

We continue with a proof of Theorem 10.8. The same argument can be used to obtain a bound for the Frobenius-norm error, but we omit a detailed account.

Proof. [Theorem 10.8] Since $\boldsymbol{\Omega}_1$ and $\boldsymbol{\Omega}_2$ are independent from each other, we can study how the error depends on the matrix $\boldsymbol{\Omega}_2$ by conditioning on the event that $\boldsymbol{\Omega}_1$ is not too irregular. To that end, we define a (parameterized) event on which the spectral and Frobenius norms of the matrix $\boldsymbol{\Omega}_1^\dagger$ are both controlled. For $t \geq 1$, let

$$E_t = \left\{\boldsymbol{\Omega}_1 : \left\|\boldsymbol{\Omega}_1^\dagger\right\| \leq \frac{\mathrm{e}\sqrt{k+p}}{p+1}\cdot t \quad \text{and} \quad \left\|\boldsymbol{\Omega}_1^\dagger\right\|_{\mathrm{F}} \leq \sqrt{\frac{12k}{p}}\cdot t\right\}.$$

Invoking both parts of Proposition 10.4, we find that

$$\mathbb{P}\left(E_t^c\right) \leq t^{-(p+1)} + 4t^{-p} \leq 5t^{-p}.$$

Consider the function $h(\boldsymbol{X}) = \left\|\boldsymbol{\Sigma}_2 \boldsymbol{X} \boldsymbol{\Omega}_1^\dagger\right\|$. We quickly compute its Lipschitz constant L with the lower triangle inequality and some standard norm estimates:

$$\begin{aligned}|h(\boldsymbol{X}) - h(\boldsymbol{Y})| &\le \left\|\boldsymbol{\Sigma}_2(\boldsymbol{X} - \boldsymbol{Y})\boldsymbol{\Omega}_1^\dagger\right\| \\ &\le \|\boldsymbol{\Sigma}_2\| \, \|\boldsymbol{X} - \boldsymbol{Y}\| \left\|\boldsymbol{\Omega}_1^\dagger\right\| \le \|\boldsymbol{\Sigma}_2\| \left\|\boldsymbol{\Omega}_1^\dagger\right\| \|\boldsymbol{X} - \boldsymbol{Y}\|_{\mathrm{F}}.\end{aligned}$$

Therefore, $L \le \|\boldsymbol{\Sigma}_2\| \left\|\boldsymbol{\Omega}_1^\dagger\right\|$. Relation (10.2) of Proposition 10.1 implies that

$$\mathbb{E}[h(\boldsymbol{\Omega}_2) \mid \boldsymbol{\Omega}_1] \le \|\boldsymbol{\Sigma}_2\| \left\|\boldsymbol{\Omega}_1^\dagger\right\|_{\mathrm{F}} + \|\boldsymbol{\Sigma}_2\|_{\mathrm{F}} \left\|\boldsymbol{\Omega}_1^\dagger\right\|.$$

Applying the concentration of measure inequality, Proposition 10.3, conditionally to the random variable $h(\boldsymbol{\Omega}_2) = \left\|\boldsymbol{\Sigma}_2 \boldsymbol{\Omega}_2 \boldsymbol{\Omega}_1^\dagger\right\|_{\mathrm{F}}$ results in

$$\mathbb{P}\left\{ \left\|\boldsymbol{\Sigma}_2 \boldsymbol{\Omega}_2 \boldsymbol{\Omega}_1^\dagger\right\|_{\mathrm{F}} > \|\boldsymbol{\Sigma}_2\| \left\|\boldsymbol{\Omega}_1^\dagger\right\|_{\mathrm{F}} + \|\boldsymbol{\Sigma}_2\|_{\mathrm{F}} \left\|\boldsymbol{\Omega}_1^\dagger\right\| + \|\boldsymbol{\Sigma}_2\| \left\|\boldsymbol{\Omega}_1^\dagger\right\| \cdot u \;\middle|\; E_t \right\} \le \mathrm{e}^{-u^2/2}.$$

Under the event E_t, we have explicit bounds on the norms of $\boldsymbol{\Omega}_1^\dagger$, so

$$\mathbb{P}\left\{ \left\|\boldsymbol{\Sigma}_2 \boldsymbol{\Omega}_2 \boldsymbol{\Omega}_1^\dagger\right\|_{\mathrm{F}} > \|\boldsymbol{\Sigma}_2\| \sqrt{\frac{12k}{p}} \cdot t + \|\boldsymbol{\Sigma}_2\|_{\mathrm{F}} \frac{\mathrm{e}\sqrt{k+p}}{p+1} \cdot t + \|\boldsymbol{\Sigma}_2\| \frac{\mathrm{e}\sqrt{k+p}}{p+1} \cdot ut \;\middle|\; E_t \right\} \le \mathrm{e}^{-u^2/2}.$$

Use the fact $\mathbb{P}(E_t^c) \le 5t^{-p}$ to remove the conditioning. Therefore,

$$\mathbb{P}\left\{ \left\|\boldsymbol{\Sigma}_2 \boldsymbol{\Omega}_2 \boldsymbol{\Omega}_1^\dagger\right\|_{\mathrm{F}} > \|\boldsymbol{\Sigma}_2\| \sqrt{\frac{12k}{p}} \cdot t + \|\boldsymbol{\Sigma}_2\|_{\mathrm{F}} \frac{\mathrm{e}\sqrt{k+p}}{p+1} \cdot t + \|\boldsymbol{\Sigma}_2\| \frac{\mathrm{e}\sqrt{k+p}}{p+1} \cdot ut \right\} \le 5t^{-p} + \mathrm{e}^{-u^2/2}.$$

Insert the expressions for the norms of $\boldsymbol{\Sigma}_2$ into this result to complete the probability bound. Finally, introduce this estimate into the error bound from Theorem 9.1. □

10.4. Analysis of the power scheme. Theorem 10.6 makes it clear that the performance of the randomized approximation scheme, Algorithm 4.1, depends heavily on the singular spectrum of the input matrix. The power scheme outlined in Algorithm 4.3 addresses this problem by enhancing the decay of spectrum. We can combine our analysis of Algorithm 4.1 with Theorem 9.2 and Corollary 9.3 to obtain a detailed report on the behavior of the performance of the power scheme using a Gaussian matrix.

COROLLARY 10.10 (Average spectral error for the power scheme). *Frame the hypotheses of Theorem 10.5. Define $\boldsymbol{B} = (\boldsymbol{A}\boldsymbol{A}^*)^q \boldsymbol{A}$ for a nonnegative integer q, and construct the sample matrix $\boldsymbol{Z} = \boldsymbol{B}\boldsymbol{\Omega}$. Then*

$$\mathbb{E}\,\|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Z}})\boldsymbol{A}\| \le \left[\left(1 + \sqrt{\frac{k}{p-1}}\right) \sigma_{k+1}^{2q+1} + \frac{\mathrm{e}\sqrt{k+p}}{p} \left(\sum\nolimits_{j>k} \sigma_j^{2(2q+1)} \right)^{1/2} \right]^{1/(2q+1)}.$$

The same bound holds when we replace $\boldsymbol{Z}$ by the extended sample matrix $\boldsymbol{W}$ from (9.9).

Proof. By Hölder's inequality and Theorem 9.2,

$$\mathbb{E}\,\|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Z}})\boldsymbol{A}\| \le \left(\mathbb{E}\,\|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Z}})\boldsymbol{A}\|^{2q+1}\right)^{1/(2q+1)} \le \left(\mathbb{E}\,\|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Z}})\boldsymbol{B}\|\right)^{1/(2q+1)}.$$

Invoke Theorem 10.6 to bound the right-hand side, noting that $\sigma_j(\boldsymbol{B}) = \sigma_j^{2q+1}$. The result for the extended sample matrix follows if we use Corollary 9.3 in place of Theorem 9.2. ∎

The real message of Corollary 10.10 emerges if we bound the series using its largest term σ_{k+1}^{4q+2} and draw the factor σ_{k+1} out of the bracket:

$$\mathbb{E}\,\|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Z}})\boldsymbol{A}\| \le \left[1 + \sqrt{\frac{k}{p-1}} + \frac{\mathrm{e}\sqrt{k+p}}{p} \cdot \sqrt{\min\{m,n\} - k}\right]^{1/(2q+1)} \sigma_{k+1}.$$

In words, as we increase the exponent q, the power scheme drives the extra factor in the error to one exponentially fast. By the time $q \sim \log(\min\{m,n\})$,

$$\mathbb{E}\,\|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Z}})\boldsymbol{A}\| \sim \sigma_{k+1},$$

which is the baseline for the spectral norm.

To obtain large deviation bounds for the performance of the power scheme, simply combine Theorem 9.2 or Corollary 9.3 with Theorem 10.8. We omit a detailed statement.

Remark 10.1. We lack an analogous theory for the Frobenius norm because Theorem 9.2 depends on Proposition 8.6, which is not true for the Frobenius norm.

11. SRFT test matrices. Another way to implement the proto-algorithm from §1.3 is to use a structured random matrix so that the matrix product in Step 2 can be performed quickly. One type of structured random matrix that has been proposed in the literature is the *subsampled random Fourier transform*, or SRFT, which we discussed in §4.6. In this section, we present bounds on the performance of the proto-algorithm when it is implemented with an SRFT test matrix. In contrast with the results for Gaussian test matrices, the results in this section hold for both real and complex input matrices. Unfortunately, our results for the SRFT are also substantially less refined.

11.1. Construction and Properties. Recall from §4.6 that an SRFT is a tall $n \times \ell$ matrix of the form $\boldsymbol{\Omega} = \sqrt{n/\ell} \cdot \boldsymbol{D}\boldsymbol{F}\boldsymbol{R}^*$ where

- $\boldsymbol{D}$ is a random $n \times n$ diagonal matrix whose entries are independent and uniformly distributed on the complex unit circle;
- $\boldsymbol{F}$ is the $n \times n$ unitary discrete Fourier transform; and
- $\boldsymbol{R}$ is a random $\ell \times n$ matrix that restricts an n-dimensional vector to ℓ coordinates, chosen uniformly at random.

Up to scaling, an SRFT is just a section of a unitary matrix, so it satisfies the norm identity $\|\boldsymbol{\Omega}\| = \sqrt{n/\ell}$.

This design may seem mysterious, but there are clear intuitions to support it. Suppose that we want to estimate the energy (i.e., squared ℓ_2 norm) of a fixed vector $\boldsymbol{x}$ by sampling ℓ of its n entries at random. On average, these random entries carry

ℓ/n of the total energy. (The factor $\sqrt{n/\ell}$ reverses this scaling.) When $\boldsymbol{x}$ has a few large components, the variance of this estimator is very high. On the other hand, when the components of $\boldsymbol{x}$ have comparable magnitude, the estimator has much lower variance, so it is precise with very high probability.

The purpose of the matrix product $\boldsymbol{DF}$ is to flatten out input vectors before we sample. To see why it achieves this goal, fix a unit vector $\boldsymbol{x}$, and examine the first component of $\boldsymbol{x}^*\boldsymbol{DF}$.

$$(\boldsymbol{x}^*\boldsymbol{DF})_1 = \sum\nolimits_{i=1}^{n} x_i \varepsilon_i f_{i1},$$

where f_{ij} are the components of the DFT matrix $\boldsymbol{F}$. This random sum clearly has zero mean. Since the entries of the DFT have magnitude $n^{-1/2}$, the variance of the sum is n^{-1}. Hoeffding's inequality [71] shows that the magnitude of the first component of $\boldsymbol{x}^*\boldsymbol{DF}$ is on the order of $n^{-1/2}$ with extremely high probability. The remaining components have exactly the same property.

In fact, an appropriately designed SRFT approximately preserves the geometry of an *entire subspace of vectors*. We present a weaker result that is adequate for our purposes.

THEOREM 11.1 (The SRFT preserves rank). *Fix a $n \times k$ orthonormal matrix $\boldsymbol{W}$ with $k \geq 246$, and draw an $n \times \ell$ SRFT matrix $\boldsymbol{\Omega}$ where the parameter ℓ satisfies*

$$24\left[\sqrt{k} + \sqrt{8\log(25n)}\right]^2 \log(4k) \leq \ell \leq n.$$

Then

$$\sigma_k(\boldsymbol{W}\boldsymbol{\Omega}) \geq \frac{1}{\sqrt{3}}$$

with probability at least 1/2.

In words, the kernel of an SRFT of dimension $\ell \sim k \log k$ is unlikely to intersect a fixed k-dimensional subspace. In contrast with the Gaussian case, the logarithmic factor $\log k$ in the lower bound on ℓ cannot generally be removed (Remark B.2). Although this theorem only provides a constant failure probability, the failure rate in practice is polynomial in k^{-1}.

The proof of Theorem 11.1 appears in Appendix B. The argument closely follows [116, Thm. 3.1] and [132, Sec. 9]. Some related results appear in [103].

REMARK 11.1. For large problems, we have been able to reduce the numerical constants further. If $1 \ll \log(n) \ll k$, then sampling

$$\ell \geq 9k\log(4k)$$

coordinates is sufficient to ensure that $\sigma_k(\boldsymbol{W}\boldsymbol{\Omega}) \geq 1/\sqrt{18}$ with probability $\mathrm{O}(k^{-1/36})$. Although we have taken some pains to calculate explicit and reasonable constants, we must emphasize that no significance attaches to the precise values stated here. This prejudice toward constants is the reason for the weak bounds on the failure probability.

REMARK 11.2. This discussion suggests that the precise form of the SRFT is not particularly important. Indeed, we can replace $\boldsymbol{F}$ by any unitary matrix whose

entries are uniformly small and which is equipped with a fast matrix–vector multiply. Likewise, we can choose other distributions for the diagonal entries of $\boldsymbol{D}$, such as random signs.

11.2. Performance guarantees. We are now prepared to present detailed information on the performance of the proto-algorithm when the test matrix $\boldsymbol{\Omega}$ is an SRFT.

THEOREM 11.2 (Error bounds for SRFT). *Fix an $m \times n$ matrix $\boldsymbol{A}$ with singular values $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \dots$. Fix a number $k \geq 246$, and draw an $n \times \ell$ SRFT matrix $\boldsymbol{\Omega}$, where*

$$24 \left[\sqrt{k} + \sqrt{8 \log(25n)} \right]^2 \log(4k) \leq \ell \leq n.$$

Construct the sample matrix $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{\Omega}$. Then

$$\|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Y}})\boldsymbol{A}\| \leq \sqrt{1 + 3n/\ell} \cdot \sigma_{k+1} \quad \text{and}$$
$$\|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Y}})\boldsymbol{A}\|_{\mathrm{F}} \leq \sqrt{1 + 3n/\ell} \cdot \left(\sum\nolimits_{j>k} \sigma_j^2 \right)^{1/2}$$

with probability at least 1/2.

As we saw in §10.2, the quantity σ_{k+1} is the minimal spectral-norm error possible when approximating $\boldsymbol{A}$ with a rank-k matrix. Similarly, the series in the second bound is the minimal Frobenius-norm error when approximating $\boldsymbol{A}$ with a rank-k matrix. We see that both error bounds lie within a polynomial factor of the baseline, and this factor decreases with the number ℓ of samples we retain.

In practice, $\ell = k \log k$ is a reasonable choice for the sampling parameter. Although the factor $\log k$ cannot generally be removed, experiments suggest that the selection $\ell = k + 20$ is often adequate.

The likelihood of error with an SRFT test matrix is much higher than in the Gaussian case. In practice, the failure probability here is polynomial in k^{-1}, while in the Gaussian failure probability is $\mathrm{e}^{-(\ell - k)}$ or better. This difference is not an artifact of the analysis. Discrete sampling techniques inherently fail with higher probability because of coupon collection issues (Remark B.2).

We finish the section with the proof of Theorem 11.2.

Proof. [Theorem 11.2] Let $\boldsymbol{V}$ be the right unitary factor of matrix $\boldsymbol{A}$, and partition $\boldsymbol{V} = [\boldsymbol{V}_1 \mid \boldsymbol{V}_2]$ into blocks containing, respectively, k and $n - k$ columns. Recall that

$$\boldsymbol{\Omega}_1 = \boldsymbol{V}_1^* \boldsymbol{\Omega} \quad \text{and} \quad \boldsymbol{\Omega}_2 = \boldsymbol{V}_2^* \boldsymbol{\Omega}.$$

where $\boldsymbol{\Omega}$ is the conjugate transpose of an SRFT. Theorem 11.1 ensures that the submatrix $\boldsymbol{\Omega}_1$ has full row rank with probability at least 1/2. Therefore, Theorem 9.1 implies that

$$|||(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{X}})\boldsymbol{A}||| \leq |||\boldsymbol{\Sigma}_2||| \left[1 + \left\| \boldsymbol{\Omega}_1^\dagger \right\|^2 \cdot \|\boldsymbol{\Omega}_2\|^2 \right]^{1/2},$$

where $|||\cdot|||$ denotes either the spectral norm or the Frobenius norm. Our application of Theorem 11.1 also ensures that the spectral norm of $\boldsymbol{\Omega}_1^\dagger$ is under control:

$$\left\| \boldsymbol{\Omega}_1^\dagger \right\| \leq \sqrt{3}.$$

We may bound the spectral norm of $\boldsymbol{\Omega}_2$ deterministically.

$$\|\boldsymbol{\Omega}_2\| = \|\boldsymbol{V}_2^*\boldsymbol{\Omega}\| \le \|\boldsymbol{V}_2^*\| \, \|\boldsymbol{\Omega}\| = \sqrt{n/\ell}$$

since $\boldsymbol{V}_2$ and $\sqrt{\ell/n} \cdot \boldsymbol{\Omega}$ are both orthonormal matrices. Combine these estimates to complete the proof. □

Acknowledgments. The authors have benefited from valuable discussions with many researchers, among them Inderjit Dhillon, Petros Drineas, Ming Gu, Edo Liberty, Michael Mahoney, Vladimir Rokhlin, Yoel Shkolnisky, and Arthur Szlam. In particular, we would like to thank Mark Tygert for his insightful remarks on early drafts of this paper. The example in Section 7.3 was provided by François Meyer of the University of Colorado at Boulder. The example in Section 7.4 comes from the FERET database of facial images collected under the FERET program, sponsored by the DoD Counterdrug Technology Development Program Office. The work reported was initiated during the program *Mathematics of Knowledge and Search Engines* held at IPAM in the fall of 2007.

Appendix A. On Gaussian matrices. This section collects some of the properties of Gaussian matrices that we use in our analysis. Most of the results follow quickly from material that is already available in the literature. In one case, however, we require a surprisingly difficult new argument. We focus on the real case here; the complex case is similar but actually yields better results.

A.1. Expectation of norms. We begin with the expected Frobenius norm of a scaled Gaussian matrix, which follows from an easy calculation.

PROPOSITION A.1. *Fix matrices $\boldsymbol{S}, \boldsymbol{T}$, and draw a standard Gaussian matrix $\boldsymbol{G}$. Then*

$$\left(\mathbb{E}\,\|\boldsymbol{S}\boldsymbol{G}\boldsymbol{T}^*\|_{\mathrm{F}}^2\right)^{1/2} = \|\boldsymbol{S}\|_{\mathrm{F}} \, \|\boldsymbol{T}\|_{\mathrm{F}} \, .$$

Proof. The distribution of a Gaussian matrix is invariant under orthogonal transformations, and the Frobenius norm is also unitarily invariant. As a result, it represents no loss of generality to assume that $\boldsymbol{S}$ and $\boldsymbol{T}$ are diagonal. Therefore,

$$\mathbb{E}\,\|\boldsymbol{S}\boldsymbol{G}\boldsymbol{T}^*\|_{\mathrm{F}}^2 = \mathbb{E}\left[\sum\nolimits_{jk} |s_{jj} g_{jk} t_{kk}|^2\right] = \sum\nolimits_{jk} |s_{jj}|^2 |t_{kk}|^2 = \|\boldsymbol{S}\|_{\mathrm{F}}^2 \, \|\boldsymbol{T}\|_{\mathrm{F}}^2 \, .$$

Since the right-hand side is unitarily invariant, we have also identified the value of the expectation for general matrices $\boldsymbol{S}$ and $\boldsymbol{T}$. □

The expected spectral norm of a scaled Gaussian matrix is also known. The result is due to Gordon [62, 63], who established the bound using a sharp version of Slepian's lemma. See [85, §3.3] and [37, §2.3] for additional discussion.

PROPOSITION A.2. *Fix matrices $\boldsymbol{S}, \boldsymbol{T}$, and draw a standard Gaussian matrix $\boldsymbol{G}$. Then*

$$\mathbb{E}\,\|\boldsymbol{S}\boldsymbol{G}\boldsymbol{T}^*\| \le \|\boldsymbol{S}\| \, \|\boldsymbol{T}\|_{\mathrm{F}} + \|\boldsymbol{S}\|_{\mathrm{F}} \, \|\boldsymbol{T}\| \, .$$

REMARK A.1. *Proposition A.2 provides a very accurate bound. When $\boldsymbol{S}$ and $\boldsymbol{T}$ are diagonal, the first term $\|\boldsymbol{S}\| \, \|\boldsymbol{T}\|_{\mathrm{F}}$ is roughly the maximum expected ℓ_2 norm of a row of $\boldsymbol{SGT}^*$; the second term is the maximum expected ℓ_2 norm of a column. Each of these quantities is a lower bound for the spectral norm. Furthermore, when $\boldsymbol{S}, \boldsymbol{T}$ are identity matrices, the result is asymptotically sharp as the size of the matrix $\boldsymbol{G}$ tends to infinity with the ratio of its dimensions fixed.*

A.2. Spectral norm of pseudoinverse. Now, we turn to the pseudoinverse of a Gaussian matrix. Recently, Chen and Dongarra developed a good bound on the probability that its spectral norm is large. The statement here follows from [28, Lem. 4.1] after an application of Stirling's approximation. See also [94, Lem. 2.14]

PROPOSITION A.3. *Let $\boldsymbol{G}$ be an $m \times n$ standard Gaussian matrix with $n \geq m$. For each $t \geq 1$,*

$$\mathbb{P}\left\{ \left\| \boldsymbol{G}^\dagger \right\| > t \right\} \leq \frac{1}{\sqrt{2\pi(n-m+1)}} \left[\frac{\mathrm{e}\sqrt{n}}{n-m+1} \right]^{n-m+1} t^{-(n-m+1)}.$$

We can use Proposition A.3 to bound the expected spectral norm of a pseudo-inverted Gaussian matrix.

PROPOSITION A.4. *Let $\boldsymbol{G}$ be a $m \times n$ standard Gaussian matrix with $n - m \geq 1$. Then*

$$\mathbb{E} \left\| \boldsymbol{G}^\dagger \right\| < \frac{\mathrm{e}\sqrt{n}}{n-m}$$

Proof. Let us make the abbreviations $p = n - m$ and

$$C = \frac{1}{\sqrt{2\pi(p+1)}} \left[\frac{\mathrm{e}\sqrt{n}}{p+1} \right]^{p+1}.$$

We compute the expectation by way of a standard argument. The integral formula for the mean of a nonnegative random variable implies that, for all $E > 0$,

$$\begin{aligned} \mathbb{E} \left\| \boldsymbol{G}^\dagger \right\| = \int_0^\infty \mathbb{P}\left\{ \left\| \boldsymbol{G}^\dagger \right\| > t \right\} \, \mathrm{d}t &\leq E + \int_E^\infty \mathbb{P}\left\{ \left\| \boldsymbol{G}^\dagger \right\| > t \right\} \, \mathrm{d}t \\ &\leq E + C \int_E^\infty t^{-(p+1)} \, \mathrm{d}t = E + \frac{1}{p} C E^{-p}, \end{aligned}$$

where the second inequality follows from Proposition A.3. The right-hand side is minimized when $E = C^{1/(p+1)}$. Substitute and simplify. ∎

A.3. Frobenius norm of pseudoinverse. The squared Frobenius norm of a pseudo-inverted Gaussian matrix is closely connected with the trace of an inverted Wishart matrix. This observation leads to an exact expression for the expectation.

PROPOSITION A.5. *Let $\boldsymbol{G}$ be an $m \times n$ standard Gaussian matrix with $n - m \geq 2$. Then*

$$\mathbb{E} \left\| \boldsymbol{G}^\dagger \right\|_{\mathrm{F}}^2 = \frac{m}{n-m-1}.$$

Proof. Observe that, almost surely,

$$\left\|\boldsymbol{G}^{\dagger}\right\|_{\mathrm{F}}^{2} = \operatorname{trace}\left[(\boldsymbol{G}^{\dagger})^{*}\boldsymbol{G}^{\dagger}\right] = \operatorname{trace}\left[(\boldsymbol{G}\boldsymbol{G}^{*})^{-1}\right]$$

The random matrix $(\boldsymbol{G}\boldsymbol{G}^{*})^{-1}$ follows the inverted Wishart distribution. As a result, its expected trace can be computed directly [100, p. 97]. □

On the other hand, very little sees to be known about the tail behavior of the Frobenius norm of a pseudo-inverted Gaussian matrix. The following theorem, which is new, provides an adequate bound on the probability of a large deviation.

THEOREM A.6. *Let $\boldsymbol{G}$ be an $m \times n$ standard Gaussian matrix with $n - m \geq 4$. For each $t \geq 1$,*

$$\mathbb{P}\left\{\left\|\boldsymbol{G}^{\dagger}\right\|_{\mathrm{F}}^{2} > \frac{12m}{n-m} \cdot t\right\} \leq 4t^{-(n-m)/2}.$$

Neither the precise form of the estimate nor the constants are ideal, as we explain in Remark A.2. Our goal has been to develop a useful bound with minimal fuss.

The rest of the section is devoted to the rather lengthy proof. Unfortunately, most of the standard methods for producing tail bounds fail for random variables that do not exhibit normal or exponential concentration. Our argument relies on special properties of Gaussian matrices and a dose of brute force.

A.3.1. Technical background. We begin with a piece of notation. For any number $q \geq 1$, we define the L_q norm of a random variable Z by

$$\mathbb{E}^{q}(Z) = \left(\mathbb{E}\,|Z|^{q}\right)^{1/q}.$$

In particular, the L_q norm satisfies the triangle inequality.

We continue with a collection of technical results. First, we present a striking fact about the structure of Gaussian matrices [54, §3.5].

PROPOSITION A.7. *An $m \times n$ standard Gaussian matrix is orthogonally equivalent with the bidiagonal matrix*

$$\boldsymbol{L} = \begin{bmatrix} X_n & & & & & \\ Y_{m-1} & X_{n-1} & & & & \\ & Y_{m-2} & X_{n-2} & & & \\ & & \ddots & \ddots & & \\ & & & Y_1 & X_{n-(m-1)} & \end{bmatrix}_{m \times n}, \tag{A.1}$$

where, for each j, the random variables X_j^2 and Y_j^2 follow the χ^2 distribution with j degrees of freedom. Furthermore, these variates are mutually independent.

We also require the moments of a chi-square variate, which are expressed in terms of special functions.

PROPOSITION A.8. *Let Ξ be a χ^2 variate with k degrees of freedom. When $0 \leq q < k/2$,*

$$\mathbb{E}\left(\Xi^{q}\right) = \frac{2^{q}\Gamma(k/2+q)}{\Gamma(k/2)} \quad \text{and} \quad \mathbb{E}\left(\Xi^{-q}\right) = \frac{\Gamma(k/2-q)}{2^{q}\Gamma(k/2)}.$$

Proof. Recall that a χ^2 variate with k degrees of freedom has the probability density function

$$f(t) = \frac{1}{2^{k/2}\Gamma(k/2)} t^{k/2-1} \mathrm{e}^{-t/2}, \quad \text{for } t \geq 0.$$

By the integral formula for expectation,

$$\mathbb{E}(\Xi^q) = \int_0^\infty t^q f(t)\,\mathrm{d}t = \frac{2^q \Gamma(k/2+q)}{\Gamma(k/2)},$$

where the second equality follows from Euler's integral expression for the gamma function. The other calculation is similar. ∎

To simplify the proof, we need to eliminate the gamma functions. The next result bounds the positive moments of a chi-square variate.

LEMMA A.9. *Let Ξ be a χ^2 variate with k degrees of freedom. For $q \geq 1$,*

$$\mathbb{E}^q(\Xi) \leq k + q.$$

Proof. Write $q = r + \theta$, where $r = \lfloor q \rfloor$. Repeated application of the functional equation $z\Gamma(z) = \Gamma(z+1)$ yields

$$\mathbb{E}^q(\Xi) = \left[\frac{2^\theta \Gamma(k/2+\theta)}{\Gamma(k/2)} \cdot \prod_{j=1}^{r} (k + 2(q-j)) \right]^{1/q}.$$

The gamma function is logarithmically convex, so

$$\frac{2^\theta \Gamma(k/2+\theta)}{\Gamma(k/2)} \leq \frac{2^\theta \cdot \Gamma(k/2)^{1-\theta} \cdot \Gamma(k/2+1)^\theta}{\Gamma(k/2)} = k^\theta \leq \left[\prod_{j=1}^{r} (k + 2(q-j)) \right]^{\theta/r}.$$

The second inequality holds because k is smaller than each term in the product, hence is smaller than their geometric mean. As a consequence,

$$\mathbb{E}^q(\Xi) \leq \left[\prod_{j=1}^{r} (k + 2(q-j)) \right]^{1/r} \leq \frac{1}{r} \sum_{j=1}^{r} (k + 2(q-j)) \leq k + q.$$

The second relation is the inequality between the geometric and arithmetic mean. ∎

Finally, we develop a bound for the negative moments of a chi-square variate.

LEMMA A.10. *Let Ξ be a χ^2 variate with k degrees of freedom, where $k \geq 5$. When $2 \leq q \leq (k-1)/2$,*

$$\mathbb{E}^q\left(\Xi^{-1}\right) < \frac{3}{k}.$$

Proof. We establish the bound for $q = (k-1)/2$. For smaller values of q, the result follows from Hölder's inequality. Proposition A.8 shows that

$$\mathbb{E}^q\left(\Xi^{-1}\right) = \left[\frac{\Gamma(k/2-q)}{2^q\Gamma(k/2)}\right]^{1/q} = \left[\frac{\Gamma(1/2)}{2^q\Gamma(k/2)}\right]^{1/q}.$$

Stirling's approximation ensures that $\Gamma(k/2) \geq \sqrt{2\pi}\cdot(k/2)^{(k-1)/2}\cdot \mathrm{e}^{-k/2}$. Since the value $\Gamma(1/2) = \sqrt{\pi}$,

$$\mathbb{E}^q\left(\Xi^{-1}\right) \leq \left[\frac{\sqrt{\pi}}{2^q\sqrt{2\pi}\cdot(k/2)^q\cdot \mathrm{e}^{-q-1/2}}\right]^{1/q} = \frac{\mathrm{e}}{k}\left[\frac{\mathrm{e}}{2}\right]^{1/2q} < \frac{3}{k},$$

where we used the assumption $q \geq 2$ to complete the numerical estimate. ∎

A.3.2. Proof of Theorem A.6. Let $\boldsymbol{G}$ be an $m \times n$ Gaussian matrix, where we assume that $n - m \geq 4$. Define the random variable

$$Z = \left\|\boldsymbol{G}^\dagger\right\|_{\mathrm{F}}^2.$$

Our goal is to develop a tail bound for Z. The argument is inspired by work of Szarek [129, §6] for square Gaussian matrices.

The first step is to find an explicit, tractable representation for the random variable. According to Proposition A.7, a Gaussian matrix $\boldsymbol{G}$ is orthogonally equivalent with a bidiagonal matrix $\boldsymbol{L}$ of the form (A.1). Making an analogy with the inversion formula for a triangular matrix, we realize that the pseudoinverse of $\boldsymbol{L}$ is given by

$$\boldsymbol{L}^\dagger = \begin{bmatrix} X_n^{-1} & & & & \\ \frac{-Y_{m-1}}{X_nX_{n-1}} & X_{n-1}^{-1} & & & \\ & \frac{-Y_{m-2}}{X_{n-1}X_{n-2}} & X_{n-2}^{-1} & & \\ & & \ddots & \ddots & \\ & & & \frac{-Y_1}{X_{n-(m-2)}X_{n-(m-1)}} & X_{n-(m-1)}^{-1} \end{bmatrix}_{n\times m}.$$

Since $\boldsymbol{L}^\dagger$ is orthogonally equivalent with $\boldsymbol{G}^\dagger$,

$$Z = \left\|\boldsymbol{G}^\dagger\right\|_{\mathrm{F}}^2 = \left\|\boldsymbol{L}^\dagger\right\|_{\mathrm{F}}^2 \leq \sum_{j=0}^{m-1}\frac{1}{X_{n-j}^2}\left(1 + \frac{Y_{m-j}^2}{X_{n-j+1}^2}\right),$$

where we have added an extra subdiagonal term (corresponding with $j = 0$) so that we can avoid exceptional cases later. We abbreviate the summands as

$$W_j = \frac{1}{X_{n-j}^2}\left(1 + \frac{Y_{m-j}^2}{X_{n-j+1}^2}\right), \quad j = 0, 1, 2, \ldots, m-1.$$

Next, we develop a large deviation bound for each summand by computing a moment and invoking Markov's inequality. For the exponent $q = (n-m)/2$, Lemmas A.9

and A.10 yield

$$\begin{aligned}\mathbb{E}^q(W_j) &= \mathbb{E}^q(X_{n-j}^{-2}) \cdot \mathbb{E}^q\left[1 + \frac{Y_{m-j}^2}{X_{n-j+1}^2}\right] \\ &\le \mathbb{E}^q(X_{n-j}^{-2})\left[1 + \mathbb{E}^q(Y_{m-j}^2) \cdot \mathbb{E}^q(X_{n-j+1}^{-2})\right] \\ &\le \frac{3}{n-j}\left[1 + \frac{3(m-j+q)}{n-j+1}\right] \\ &= \frac{3}{n-j}\left[1 + 3 - \frac{3(n-m+1-q)}{n-j+1}\right]\end{aligned}$$

Note that the first two relations require the independence of the variates and the triangle inequality for the L_q norm. The maximum value of the bracket evidently occurs when $j = 0$, so

$$\mathbb{E}^q(W_j) < \frac{12}{n-j}, \quad j = 0, 1, 2, \dots, m-1.$$

Markov's inequality results in

$$\mathbb{P}\left\{W_j \ge \frac{12}{n-j} \cdot u\right\} \le u^{-q}.$$

Select $u = t \cdot (n-j)/(n-m)$ to reach

$$\mathbb{P}\left\{W_j \ge \frac{12}{n-m} \cdot t\right\} \le \left[\frac{n-m}{n-j}\right]^q t^{-q}.$$

To complete the argument, we combine these estimates by means of the union bound and clean up the resulting mess. Since $Z \le \sum_{j=0}^{m-1} W_j$,

$$\mathbb{P}\left\{Z \ge \frac{12m}{n-m} \cdot t\right\} \le t^{-q} \sum_{j=0}^{m-1} \left[\frac{n-m}{n-j}\right]^q.$$

To control the sum on the right-hand side, observe that

$$\begin{aligned}\sum_{j=0}^{m-1} \left[\frac{n-m}{n-j}\right]^q &< (n-m)^q \int_0^m (n-x)^{-q}\,\mathrm{d}x \\ &< \frac{(n-m)^q}{q-1}(n-m)^{-q+1} = \frac{2(n-m)}{n-m-2} \le 4,\end{aligned}$$

where the last inequality follows from the hypothesis $n - m \ge 4$. Together, the estimates in this paragraph produce the advertised bound.

REMARK A.2. *We have presented the simplest argument we know that produces a reasonable result. With a sufficient investment of care, it is possible to obtain sharper estimates. For example, one can establish that the tail bound actually decays like* $t^{-(n-m+1)/2}$ *by controlling the summands* W_j *more delicately.*

One can also show that the correct scale for large deviations is $(n-m)^{-1}$*, rather than* $m(n-m)^{-1}$*. This argument depends on the observation that the even-indexed*

summands $W_0, W_2, W_4, \dots$ are mutually independent. To analyze the sum of the even terms, we can use truncation and Bernstein's inequality. This approach also requires a detailed estimate of the variance of the truncated sum. The odd terms receive a parallel treatment. Unfortunately, this proof requires some fortitude.

It would be very interesting to find a direct argument that yields concentration results for inverse spectral functions of a Gaussian matrix.

Appendix B. Subsampled random Fourier transforms. This section demonstrates that a structured dimension reduction map, called the SRFT, preserves the geometry of an entire subspace of vectors. These results have strong precedents in [116, Thm. 3.1] and [132, §9]. See also [103].

B.1. Technical background. We begin with some notation. For any $q \geq 1$, we write

$$\mathbb{E}^q(Z) = (\mathbb{E}\,|Z|^q)^{1/q}$$

for the L_q norm of the random variable Z. We also use a special notation for the expectation with respect to one random variable, holding others fixed.

$$\mathbb{E}_X\, f(X, Y) = \mathbb{E}[f(X, Y) \mid Y].$$

A *Rademacher random variable* takes the values ± 1 with equal probability. We reserve the letter ξ for a Rademacher variable, and we often write $\boldsymbol{\xi}$ for a vector whose entries are independent Rademacher variables. A *Steinhaus random variable* is uniformly distributed on the complex unit circle. We reserve ε and $\boldsymbol{\varepsilon}$ for Steinhaus random variables and vectors.

We need to distinguish two different models for a random subset T of the indices $\{1, 2, \dots, n\}$. Let ℓ be an integer in the range $[0, n]$.

Dependent Model Let $\delta_1, \dots, \delta_n$ be 0–1 random variables that satisfy $\sum \delta_j = \ell$, with all such choices equally likely. Observe that the random variables have common mean $\delta = \ell/n$, but they are not independent.

Independent Model Let $\delta_1, \dots, \delta_n$ be independent 0–1 random variables with common mean $\delta = \ell/n$.

In each case, we define a random set $T = \{j : \delta_j = 1\}$. In the dependent model, we interpret T as a uniformly random subset of $\{1, 2, \dots, n\}$ with cardinality ℓ. In the independent model, T is a random subset with *expected* cardinality ℓ.

Given a subset T of indices in $\{1, 2, \dots, n\}$, we define the restriction operator $\boldsymbol{R}_T : \mathbb{C}^n \to \mathbb{C}^T$ via the rule

$$(\boldsymbol{R}_T \boldsymbol{x})(j) = x_j, \quad j \in T.$$

When T is a random subset, $\boldsymbol{R}_T$ is a restriction to a random set of coordinates.

In this appendix, we find it more convenient to work with the *conjugate transpose* of the SRFT matrix that we work with in the body of the paper. We use a different letter to preserve the distinction, so it should present no confusion that we also call this transform an SRFT. For $1 \leq \ell \leq n$, we define the $\ell \times n$ matrix

$$\boldsymbol{\Phi} = \sqrt{\frac{n}{\ell}} \cdot \boldsymbol{R}_T \boldsymbol{F} \boldsymbol{D},$$

where

- $\boldsymbol{D} = \operatorname{diag}(\boldsymbol{\varepsilon})$ is an $n \times n$ diagonal matrix with Steinhaus entries;
- $\boldsymbol{F}$ is the $n \times n$ unitary DFT matrix; and
- T is a random set of ℓ coordinates, drawn from the dependent model.

Alternatively, we can construct an SRFT by changing the distribution of the subset T to the independent model. An independent SRFT has ℓ rows *only in expectation*. For this conceptual reason, we have adopted the dependent model in the body of the paper. On the other hand, the technical arguments are better adapted to the independent model.

The following basic result allows us to transfer conclusions from the independent model back to the dependent model. We learned this argument from [25, Sec. 3]. In the statement, we say that a real-valued function f on sets is *monotonic* if $S \subset S'$ implies $f(S) \le f(S')$.

PROPOSITION B.1 (Decoupling). *Suppose f is a monotonic function on subsets of $\{1, 2, \dots, n\}$. Let S be a random set of expected cardinality ℓ drawn from the independent model, and let T be a random set of cardinality ℓ drawn from the dependent model. Then*

$$\mathbb{P}\{f(T) \le u\} \le 2\mathbb{P}\{f(S) \le u\} \quad \text{for all } u \in \mathbb{R}.$$

Proof. Using the law of total probability, we compute that

$$\begin{aligned}
\mathbb{P}\{f(S) \le u\} &= \sum\nolimits_{j=0}^{n} \mathbb{P}\{f(S) \le u \mid |S| = j\} \cdot \mathbb{P}\{|S| = j\} \\
&\ge \sum\nolimits_{j=0}^{\ell} \mathbb{P}\{f(S) \le u \mid |S| = j\} \cdot \mathbb{P}\{|S| = j\} \\
&\ge \mathbb{P}\{f(S) \le u \mid |S| = \ell\} \cdot \sum\nolimits_{j=0}^{\ell} \mathbb{P}\{|S| = j\} \\
&\ge \mathbb{P}\{f(T) \le u\} \cdot \frac{1}{2}.
\end{aligned}$$

The second inequality requires the monotonicity of f, and the third depends on the fact [74, Thm. 3.2] that the medians of the BINOMIAL(ℓ, n) distribution lie between $\ell - 1$ and ℓ. ∎

We also need a tail bound for a function of Steinhaus variates. This result follows from an argument that Ledoux developed for bounded, real random variables [83, Cor. 1.3 et seq.]. See also [84, §5.2] for a discussion of concentration in product spaces.

PROPOSITION B.2 (Steinhaus tail bound). *Suppose h is a convex function on (complex) vectors that satisfies the Lipschitz bound*

$$|h(\boldsymbol{x}) - h(\boldsymbol{y})| \le L \|\boldsymbol{x} - \boldsymbol{y}\| \quad \text{for all } \boldsymbol{x}, \boldsymbol{y}.$$

Let $\boldsymbol{\varepsilon}$ be a Steinhaus vector. For all $t > 0$,

$$\mathbb{P}\{h(\boldsymbol{\varepsilon}) \ge \mathbb{E}\, h(\boldsymbol{\varepsilon}) + Lt\} \le \mathrm{e}^{-t^2/8}.$$

The heavy lifting is accomplished by means of the following result, due to Rudelson [114], which depends on the noncommutative Khintchine inequality [19, 91].

PROPOSITION B.3 (Rudelson's Lemma). *Let $\{\boldsymbol{a}_j\} \subset \mathbb{C}^k$ be a sequence of vectors, and let $\{c_j\}$ be a sequence of complex scalars. For each integer $q \geq 1$,*

$$\mathbb{E}^{2q} \left\| \sum\nolimits_j \xi_j c_j \boldsymbol{a}_j \boldsymbol{a}_j^* \right\| \leq (k\sqrt{2})^{1/2q} \mathrm{e}^{-0.5} \sqrt{2q} \cdot \max\nolimits_j \|\boldsymbol{a}_j\| \cdot \left\| \sum\nolimits_j |c_j|^2 \boldsymbol{a}_j \boldsymbol{a}_j^* \right\|^{1/2},$$

where $\boldsymbol{\xi}$ is a Rademacher vector.

The version here is extracted from the discussion in [132, Sec. 9], along with the fact that the optimal constant D_{2q} in the noncommutative Khintchine inequality for Rademacher series [20] verifies

$$\mathrm{D}_{2q} = [(2q-1)!!]^{1/2q} = \left[\frac{(2q)!}{2^q q!} \right]^{1/2q} \leq 2^{1/4q} \mathrm{e}^{-0.5} \sqrt{2q},$$

where the inequality follows from Stirling's approximation.

B.2. The SRFT preserves geometry. The SRFT preserves the geometry of an entire subspace. For simplicity, we focus on showing that the operator never maps a vector in the subspace to zero. The same argument also shows that the SRFT does not inflate any vector in the subspace. We establish the following result.

THEOREM B.4 (The SRFT preserves rank). *Let $\boldsymbol{V}$ be an $n \times k$ orthonormal matrix, where $k \geq 246$. Select a parameter ℓ that satisfies*

$$24 \left[\sqrt{k} + \sqrt{8 \log(25n)} \right]^2 \log(4k) \leq \ell \leq n.$$

Draw a dependent or independent $\ell \times n$ SRFT matrix $\boldsymbol{\Phi}$. Then

$$\sigma_k(\boldsymbol{\Phi V}) \geq \frac{1}{\sqrt{3}}$$

with probability exceeding $1/2$.

The constants in Theorem B.4 are somewhat larger than we might like because we wanted to ensure that the statement is effective for moderate values of k and n. We also have the following result of a more asymptotic flavor that may be more attractive.

THEOREM B.5 (SRFT: Large sample bounds). *Let $\boldsymbol{V}$ be an $n \times k$ orthonormal matrix, where $k \gg 1$ and $k \gg \log(n)$. If the number ℓ of samples satisfies*

$$9k \log(4k) \leq \ell \leq n,$$

then a dependent or independent $\ell \times n$ SRFT $\boldsymbol{\Phi}$ satisfies

$$\sigma_k(\boldsymbol{\Phi V}) \geq \frac{1}{\sqrt{18}}$$

except with probability $\mathrm{O}(k^{-1/36})$.

B.3. Overview of Theorem B.4. We present the main line of reasoning here, postponing the proofs of the lemmata to the sequel. The first step is to show that the matrix $\boldsymbol{FD}$ equilibrates row norms.

LEMMA B.6 (Row norms). *Let $\boldsymbol{V}$ be an $n \times k$ orthonormal matrix. Then $\boldsymbol{FDV}$ is an $n \times k$ orthonormal matrix, and*

$$\mathbb{P}\left\{ \max_{j=1,\dots,n} \left\| \mathbf{e}_j^*(\boldsymbol{FDV}) \right\| \leq \sqrt{\frac{k}{n}} + \sqrt{\frac{8\log(\beta n)}{n}} \right\} \leq \frac{1}{\beta}$$

for each $\beta > 1$.

When $k \gg \log n$, the second term in the bound is negligible, in which case the row norms are essentially as small as possible. On the other hand, when $k < \log n$, small sample effects can make some row norms large.

The next result states that randomly sampling rows from an orthonormal matrix results in a full-rank matrix. The minimum size of the sample depends primarily on the row norms, and the sampling procedure is most efficient when the orthonormal matrix has uniformly small rows.

LEMMA B.7 (Row sampling: Independent model). *Let $\boldsymbol{W}$ be an $n \times k$ orthonormal matrix, and define $r = n \cdot \max_{j=1,\dots,n} \left\| \mathbf{e}_j^* \boldsymbol{W} \right\|^2$. Fix $\eta \in (0, 0.5)$, and select*

$$\ell \geq \frac{8r\log(4k)}{1-2\eta}.$$

Draw a random set T of expected cardinality ℓ from the independent model. Then

$$\sigma_k(\boldsymbol{R}_T \boldsymbol{W}) \geq \sqrt{\frac{\eta\ell}{n}}$$

except with probability $(1 - \eta/(4 - 4\eta))^{2\log(k)} \leq k^{-\eta/2}$.

Select $\boldsymbol{W} = \boldsymbol{FDV}$. Lemma B.6 with $\beta = 25$ establishes that $\boldsymbol{W}$ is an orthonormal matrix whose largest row norm is essentially as small as possible, so that

$$r = n \cdot \max_{j=1,\dots,n} \left\| \mathbf{e}_j^* \boldsymbol{W} \right\|^2 \leq \left[\sqrt{k} + \sqrt{8\log(25n)} \right]^2,$$

except with probability 0.04. Next, we apply Lemma B.7 with $\eta = 1/3$. For

$$\ell \geq 24r\log(4k),$$

we have $\sigma_k(\boldsymbol{R}_T \boldsymbol{W}) \geq \sqrt{\ell/3n}$ except with probability

$$\left[1 - \frac{\eta}{4(1-\eta)} \right]^{2\log(k)} = \left(\frac{7}{8} \right)^{2\log(k)} \leq 0.46,$$

provided that $k \geq 19$. Since $\boldsymbol{\Phi V} = \sqrt{n/\ell} \cdot \boldsymbol{R}_T \boldsymbol{W}$, we have established Theorem B.4 for the independent model.

For the dependent model, we replace Lemma B.7 with the following corollary.

COROLLARY B.8 (Row sampling: Dependent model). *Let T be a random subset of exact cardinality ℓ drawn from the dependent model. Then the conclusion of Lemma B.7 holds except that the failure probability may double.*

Proof. The set function $S \mapsto \sigma_k(\boldsymbol{R}_S \boldsymbol{W})$ is monotonic because of the interlacing theorem for singular values [72, Thm. 7.3.9]. The claim follows by Proposition B.1 on decoupling. □

In the dependent case, we need to assume that $k \geq 246$ to obtain a total failure probability of at most 0.5. This point completes the proof of Theorem B.4.

The same type of argument implies Theorem B.5. In this case, we take $\beta = k$ and $\eta = 1/18$ to obtain the result. We omit the details.

B.4. Proofs of supporting lemmas. It remains to check that the underlying results are true. We begin with the claim that $\boldsymbol{FE}$ balances row norms.

Proof. [Lemma B.6] Observe that the orthonormality condition on $\boldsymbol{V}$ is equivalent with $\boldsymbol{V}^*\boldsymbol{V} = \mathbf{I}_k$. Therefore, $\|\boldsymbol{V}\| = 1$ and $\|\boldsymbol{V}\|_{\mathrm{F}} = \sqrt{k}$. To check that $\boldsymbol{FDV}$ is also an orthonormal matrix, simply compute that

$$(\boldsymbol{FDV})^*(\boldsymbol{FDV}) = \boldsymbol{V}^*\boldsymbol{V} = \mathbf{I}$$

because $\boldsymbol{D}, \boldsymbol{F}$ are unitary.

Recall that $\boldsymbol{D} = \operatorname{diag}(\boldsymbol{\varepsilon})$ for a Steinhaus vector $\boldsymbol{\varepsilon}$, so we may use the Steinhaus tail bound to study the deviation of the row norms of $\boldsymbol{FDV}$. Fix a row index $j \in \{1, 2, \dots, n\}$, and define the random variable

$$h(\boldsymbol{\varepsilon}) = \left\|\mathbf{e}_j^* \boldsymbol{FDV}\right\| = \left\|\mathbf{e}_j^* \boldsymbol{F} \operatorname{diag}(\boldsymbol{\varepsilon}) \boldsymbol{V}\right\| = \|\boldsymbol{\varepsilon}^* \boldsymbol{EV}\|,$$

where $\boldsymbol{E} = \operatorname{diag}(\mathbf{e}_j^* \boldsymbol{F})$, the diagonal matrix constructed from the jth row of $\boldsymbol{F}$. It is easy to see that h is a convex function, and we quickly determine its Lipschitz constant:

$$|h(\boldsymbol{x}) - h(\boldsymbol{y})| \leq \|(\boldsymbol{x} - \boldsymbol{y})^* \boldsymbol{EV}\| \leq \|\boldsymbol{x} - \boldsymbol{y}\| \, \|\boldsymbol{E}\| \, \|\boldsymbol{V}\| = \frac{1}{\sqrt{n}} \|\boldsymbol{x} - \boldsymbol{y}\|.$$

The equality holds because each entry of the diagonal matrix $\boldsymbol{E}$ has magnitude $n^{-1/2}$ and $\boldsymbol{V}$ has unit spectral norm. It is also straightforward to bound the expectation of the random variable:

$$\mathbb{E}\, h(\boldsymbol{\varepsilon}) \leq \left[\mathbb{E}\, h(\boldsymbol{\varepsilon})^2\right]^{1/2} = \|\boldsymbol{EV}\|_{\mathrm{F}} \leq \|\boldsymbol{E}\| \, \|\boldsymbol{V}\|_{\mathrm{F}} = \sqrt{\frac{k}{n}},$$

since $\boldsymbol{V}$ has Frobenius norm $\sqrt{k}$. Apply the Steinhaus tail bound, Proposition B.2, with $t = \sqrt{8 \log(\beta n)}$ to reach

$$\mathbb{P}\left\{ \left\|\mathbf{e}_j^*(\boldsymbol{FDV})\right\| \geq \sqrt{\frac{k}{n}} + \sqrt{\frac{8 \log(\beta n)}{n}} \right\} \leq \mathrm{e}^{-8 \log(\beta n)/8} = \frac{1}{\beta n}.$$

This estimate holds for each row index $j = 1, 2, \dots, n$. Finally, take a union bound over these n events to reach the advertised conclusion. □

Proof. [Lemma B.7] We can control the kth singular value of the matrix $\boldsymbol{R}_T \boldsymbol{W}$ by bounding the smallest eigenvalue of the $k \times k$ Gram matrix

$$\boldsymbol{X} = (\boldsymbol{R}_T \boldsymbol{W})^* (\boldsymbol{R}_T \boldsymbol{W}).$$

The first task is to rewrite this Gram matrix. Let $\boldsymbol{w}_j$ denote the jth row of $\boldsymbol{W}$, and abbreviate $M = \max_{j=1,\dots,n} \|\boldsymbol{w}_j\|$. Since $\boldsymbol{W}$ is an orthonormal matrix, $\sum_j \boldsymbol{w}_j \boldsymbol{w}_j^* = \mathbf{I}$. Recall that the random set $T = \{j : \delta_j = 1\}$, where $\delta_1, \dots, \delta_n$ are independent 0–1 random variables with common mean $\delta = \ell/n$. It follows that

$$\boldsymbol{X} = \sum\nolimits_{j=1}^{n} \delta_j \boldsymbol{w}_j \boldsymbol{w}_j^*.$$

This expression also makes clear that $\mathbb{E}\,\boldsymbol{X} = \delta\,\mathbf{I}$.

Let us define a random variable that measures the spectral-norm deviation of $\boldsymbol{X}$ from its mean:

$$Z = \|\boldsymbol{X} - \delta\,\mathbf{I}\|.$$

We will develop a condition on the parameter ℓ under which $Z \le (1-\eta)\delta$ with high probability. In that event, the least eigenvalue λ_k of $\boldsymbol{X}$ satisfies $|\lambda_k - \delta| \le (1-\eta)\delta$, which implies that $\lambda_k \ge \eta\delta$. It follows that

$$\sigma_k(\boldsymbol{R}_T \boldsymbol{W}) = \sqrt{\lambda_k} \ge \sqrt{\eta\delta} = \sqrt{\frac{\eta\ell}{n}},$$

which is the conclusion of the lemma.

The major challenge is to bound an appropriate moment of the random variable, so we set

$$E = \mathbb{E}^{2q}(Z) = \mathbb{E}^{2q} \left\| \sum\nolimits_j \delta_j \boldsymbol{w}_j \boldsymbol{w}_j^* - \delta\,\mathbf{I} \right\| = \mathbb{E}^{2q} \left\| \sum\nolimits_j (\delta_j - \delta) \boldsymbol{w}_j \boldsymbol{w}_j^* \right\|,$$

where $q \ge 1$, the precise value to be decided later. To begin the estimate, we inject additional randomness via the symmetrization procedure. Let $\{\delta_j'\}$ be an independent copy of the selector variables. By Jensen's inequality,

$$E = \mathbb{E}^{2q} \left\| \sum\nolimits_j (\delta_j - \mathbb{E}\,\delta_j') \boldsymbol{w}_j \boldsymbol{w}_j^* \right\| \le \mathbb{E}^{2q} \left\| \sum\nolimits_j (\delta_j - \delta_j') \boldsymbol{w}_j \boldsymbol{w}_j^* \right\|.$$

Since $\{\delta_j - \delta_j'\}$ is an independent family of symmetric variables, the expectation does not change if, for each j, we multiply the jth summand by an independent Rademacher variable ξ_j and average.

$$\begin{aligned} E &\le \left[\mathbb{E}_{\boldsymbol{\xi}}\, \mathbb{E}_{\boldsymbol{\delta},\boldsymbol{\delta}'} \left\| \sum\nolimits_j \xi_j (\delta_j - \delta_j') \boldsymbol{w}_j \boldsymbol{w}_j^* \right\|^{2q} \right]^{1/2q} \\ &= \left[\mathbb{E}_{\boldsymbol{\delta},\boldsymbol{\delta}'}\, \mathbb{E}_{\boldsymbol{\xi}} \left\| \sum\nolimits_j \xi_j (\delta_j - \delta_j') \boldsymbol{w}_j \boldsymbol{w}_j^* \right\|^{2q} \right]^{1/2q}, \end{aligned}$$

where we use independence of $\boldsymbol{\xi}$ from the other variables to justify the interchange of the expectations. See [85, Lem. 6.3] for more discussion on symmetrization.

Rudelson's lemma, conditional on the choice of the selectors $\boldsymbol{\delta}$ and $\boldsymbol{\delta}'$, yields

$$E \le (k\sqrt{2})^{1/2q} \mathrm{e}^{-0.5} \sqrt{2q} \cdot M \cdot \mathbb{E}^{2q} \left(\left\| \sum\nolimits_j \left|\delta_j - \delta_j'\right|^2 \boldsymbol{w}_j \boldsymbol{w}_j^* \right\|^{1/2} \right),$$

Jensen's inequality permits us to move the moment inside the square root:

$$E \le (k\sqrt{2})^{1/2q} \mathrm{e}^{-0.5} \sqrt{2q} \cdot M \cdot \left(\mathbb{E}^{2q} \left\| \sum\nolimits_j \left|\delta_j - \delta_j'\right|^2 \boldsymbol{w}_j \boldsymbol{w}_j^* \right\| \right)^{1/2}.$$

Let us make some bounds on the large parenthesis. Each matrix $\boldsymbol{w}_j\boldsymbol{w}_j^*$ is psd, so the norm increases monotonically with each scalar coefficient. Thus, we can introduce the inequality $|\delta_j - \delta_j'|^2 \le \delta_j + \delta_j'$ and invoke the triangle inequality (twice) to obtain

$$\begin{aligned}\mathbb{E}^{2q}\left\|\sum\nolimits_j |\delta_j - \delta_j'|^2 \boldsymbol{w}_j\boldsymbol{w}_j^*\right\| &\le \mathbb{E}^{2q}\left\|\sum\nolimits_j (\delta_j + \delta_j')\boldsymbol{w}_j\boldsymbol{w}_j^*\right\| \\ &\le \mathbb{E}^{2q}\left\|\sum\nolimits_j \delta_j\boldsymbol{w}_j\boldsymbol{w}_j^*\right\| + \mathbb{E}^{2q}\left\|\sum\nolimits_j \delta_j'\boldsymbol{w}_j\boldsymbol{w}_j^*\right\| \\ &= 2\,\mathbb{E}^{2q}\left\|\sum\nolimits_j \delta_j\boldsymbol{w}_j\boldsymbol{w}_j^*\right\|,\end{aligned}$$

where the last identity follows from the identical distribution of the sequences. Next, we add and subtract the scalar matrix $\delta\,\mathbf{I}$ inside the norm, apply the triangle inequality, and then identify a copy of E:

$$\mathbb{E}^{2q}\left\|\sum\nolimits_j \delta_j\boldsymbol{w}_j\boldsymbol{w}_j^*\right\| \le \mathbb{E}^{2q}\left\|\sum\nolimits_j \delta_j\boldsymbol{w}_j\boldsymbol{w}_j^* - \delta\,\mathbf{I}\right\| + \|\delta\,\mathbf{I}\| = E + \delta.$$

Combining these observations, we discover

$$E \le (k\sqrt{2})^{1/2q}\mathrm{e}^{-0.5}\sqrt{4q}\cdot M\cdot (E+\delta)^{1/2}.$$

To minimize the constant, select $q = \lceil \log(k\sqrt{2})\rceil$, which is roughly optimal. In summary,

$$E \le \sqrt{4qM^2}\cdot (E+\delta)^{1/2}.$$

To study this relation, we make the change of variables $F = \delta^{-1}E$ so that

$$F^2 \le \frac{4qM^2}{\delta}\cdot(F+1).$$

A rather tedious (but elementary) computation shows that solutions to $F^2 \le \alpha(F+1)$ satisfy the rule

$$\alpha \le \frac{1}{2}(1-2\eta) \quad\Longrightarrow\quad F \le 1 - \frac{5}{4}\eta.$$

Therefore,

$$\frac{4qM^2}{\delta} \le \frac{1}{2}(1-2\eta) \quad\Longrightarrow\quad F \le 1-\frac{5}{4}\eta \quad\Longrightarrow\quad E \le \left(1-\frac{5}{4}\eta\right)\delta.$$

Recalling the definition of E along with the facts $\delta = \ell/n$ and $q < \log(4k)$, we see that

$$\ell \ge \frac{8(M^2 n)\log(4k)}{1-2\eta} \quad\Longrightarrow\quad \mathbb{E}^{2q}(Z) \le \left(1-\frac{5}{4}\eta\right)\delta.$$

In words, a lower bound on ℓ delivers an upper bound for a carefully chosen moment of the random variable Z.

Given this information, we can produce a tail bound for Z with minimal effort by invoking Markov's inequality.

$$\mathbb{P}\{Z \ge (1-\eta)\delta\} \le \left[\frac{\mathbb{E}^{2q}(Z)}{(1-\eta)\delta}\right]^{2q}.$$

Introduce the estimate for the moment, and use the lower bound $q > \log k$ to reach

$$\mathbb{P}\{Z \geq (1-\eta)\delta\} \leq \left(\frac{1-5\eta/4}{1-\eta}\right)^{2q} \leq \left[1 - \frac{\eta}{4(1-\eta)}\right]^{2\log(k)}.$$

We can further bound the right-hand side as

$$\left[1 - \frac{\eta}{4(1-\eta)}\right]^{2\log(k)} \leq (1-\eta/4)^{2\log(k)} = k^{-2\log(1-\eta/4)} \leq k^{-\eta/2}.$$

This point completes the argument. □

REMARK B.1. In the proof of Lemma B.7, we used Markov's inequality to bound the probability that Z exhibits a large deviation above its mean. In the present setting, this approach produces an imprecise conclusion. A more careful argument, reminiscent of [115, Thm. 3.9], shows that large deviations are even less likely.

REMARK B.2. The logarithmic factor in these results is *necessary*, as we show by example. Fix an integer k, and let $n = k^2$. Form an $n \times k$ orthonormal matrix $\boldsymbol{W}$ by vertically stacking k copies of the $k \times k$ unitary DFT, scaled so that $\|\boldsymbol{W}\| = 1$. Suppose that we sample random rows from $\boldsymbol{W}$ to produce a matrix $\boldsymbol{X}$. To ensure that $\sigma_k(\boldsymbol{X}) > 0$, we must pick at least one copy each of the k distinct rows. This is a coupon collection problem [56]. It is well known that, to obtain a complete set of k coupons (i.e., rows) with nonnegligible probability, about $k \log k$ draws are required. We accrue essentially no advantage by sampling without replacement because the matrix has too many rows.

It is indeed possible for a matrix of the form $\boldsymbol{W} = \boldsymbol{FDV}$ to present this undesirable property. Consider the matrix $\boldsymbol{V}$ formed by regular decimation of the $n \times n$ identity matrix. More precisely, let $\boldsymbol{V}$ be the $n \times k$ matrix whose jth row is $k^{-1/2}\mathbf{e}_{j/k}^*$ when $j \equiv 0 \pmod{k}$ and zero otherwise.

REFERENCES

[1] D. ACHLIOPTAS, *Database-friendly random projections: Johnson–Lindenstrauss with binary coins*, J. Comput. System Sci., 66 (2003), pp. 671–687.

[2] D. ACHLIOPTAS AND F. MCSHERRY, *Fast computation of low-rank matrix approximations*, J. ACM, 54 (2007), pp. Art. 9, 19 pp. (electronic).

[3] N. AILON AND B. CHAZELLE, *Approximate nearest neighbors and the fast Johnson–Lindenstrauss transform*, in STOC '06: Proc. 38th Ann. ACM Symp. Theory of Computing, 2006, pp. 557–563.

[4] N. AILON AND E. LIBERTY, *Fast dimension reduction using Rademacher series on dual BCH codes*, in STOC '08: Proc. 40th Ann. ACM Symp. Theory of Computing, 2008.

[5] N. ALON, P. GIBBONS, Y. MATIAS, AND M. SZEGEDY, *Tracking join and self-join sizes in limited storage*, in Proc. 18th ACM Symp. Principles of Database Systems (PODS), 1999, pp. 10–20.

[6] N. ALON, Y. MATIAS, AND M. SZEGEDY, *The space complexity of approximating frequency moments*, in STOC '96: Proc. 28th Ann. ACM Symp. Theory of Algorithms, 1996, pp. 20–29.

[7] S. ARORA, E. HAZAN, AND S. KALE, *A fast random sampling algorithm for sparsifying matrices*, in Approximation, randomization and combinatorial optimization: Algorithms and Techniques, Springer, Berlin, 2006, pp. 272–279.

[8] A. R. BARRON, *Universal approximation bounds for superpositions of a sigmoidal function*, IEEE Trans. Inform. Theory, 39 (1993), pp. 930–945.

[9] I. BEICHL, *The Metropolis algorithm*, Comput. Sci. Eng., (2000), pp. 65–69.

[10] M.-A. Bellabas and P. J. Wolfe, *On sparse representations of linear operators and the approximation of matrix products*, in Proc. 42nd Ann. Conf. Information Sciences and Systems (CISS), 2008, pp. 258–263.
[11] R. Bhatia, *Matrix Analysis*, no. 169 in GTM, Springer, Berlin, 1997.
[12] Å. Björck, *Numerics of Gram-Schmidt orthogonalization*, Linear Algebra Appl., 197–198 (1994), pp. 297–316.
[13] ———, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, PA, 1996.
[14] V. Bogdanov, *Gaussian Measures*, AMS, Providence, RI, 1998.
[15] J. Bourgain, *On Lipschitz embedding of finite metric spaces in Hilbert space*, Israel J. Math., 52 (1985), pp. 46–52.
[16] C. Boutsidis and P. Drineas, *Random projections for nonnegative least squares*, Linear Algebra Appl., 431 (2009), pp. 760–771.
[17] C. Boutsidis, P. Drineas, and M. W. Mahoney, *An improved approximation algorithm for the column subset selection problem*, in STOC '09: Proc. 41st Ann. ACM Symp. Theory of Computing, 2009.
[18] C. Boutsidis, M. W. Mahoney, and P. Drineas, *Unsupervised feature selection for principal components analysis*, in Proc. ACM SIGKDD Intl. Conf. Knowledge Discovery and Data Mining (KDD), Aug. 2008.
[19] A. Buchholz, *Operator Khintchine inequality in non-commutative probability*, Math. Ann., 319 (2001), pp. 1–16.
[20] ———, *Optimal constants in Khintchine-type inequalities for Fermions, Rademachers and q-Gaussian operators*, Bull. Pol. Acad. Sci. Math., 53 (2005), pp. 315–321.
[21] E. Candès and J. K. Romberg, *Sparsity and incoherence in compressive sampling*, Inverse Problems, 23 (2007), pp. 969–985.
[22] E. Candès, J. K. Romberg, and T. Tao, *Robust uncertainty principles: Exact signal reconstruction from highly incomplete Fourier information*, IEEE Trans. Inform. Theory, 52 (2006), pp. 489–509.
[23] E. J. Candès, *Compressive sampling*, in Proc. 2006 Intl. Congress of Mathematicians, Madrid, 2006.
[24] E. J. Candès and B. Recht, *Exact matrix completion via convex optimization*, Found. Comput. Math., (2009). To appear. Available at `arXiv:0805.4471`.
[25] E. J. Candès and J. K. Romberg, *Quantitative robust uncertainty principles and optimally sparse decompositions*, Found. Comput. Math, 6 (2006), pp. 227–254.
[26] E. J. Candès and T. Tao, *The power of convex relaxation: Near-optimal matrix completion.* Submitted. Available at `arXiv:0903.1476`, Mar. 2009.
[27] B. Carl, *Inequalities of Bernstein–Jackson-type and the degree of compactness in Banach spaces*, Ann. Inst. Fourier (Grenoble), 35 (1985), pp. 79–118.
[28] Z. Chen and J. J. Dongarra, *Condition numbers of Gaussian random matrices*, SIAM J. Matrix Anal. Appl., 27 (2005), pp. 603–620.
[29] H. Cheng, Z. Gimbutas, P. G. Martinsson, and V. Rokhlin, *On the compression of low rank matrices*, SIAM J. Sci. Comput., 26 (2005), pp. 1389–1404 (electronic).
[30] A. Çivril and M. Magdon-Ismail, *On selecting a maximum volume sub-matrix of a matrix and related problems*, Theoret. Comput. Sci., (2009). To appear. Available at `http://www.cs.rpi.edu/~civria/`.
[31] K. L. Clarkson, *Subgradient and sampling algorithms for ℓ_1 regression*, in Proc. 16th Ann. ACM–SIAM Symp. Discrete Algorithms (SODA), 2005, pp. 257–266.
[32] K. L. Clarkson and D. P. Woodruff, *Numerical linear algebra in the streaming model*, in STOC '09: Proc. 41st Ann. ACM Symp. Theory of Computing, 2009.
[33] R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker, *Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps*, Proc. Natl. Acad. Sci. USA, 102 (2005), pp. 7426–7431.
[34] A. Dasgupta, P. Drineas, B. Harb, R. Kumar, and M. W. Mahoney, *Sampling algorithms and coresets for ℓ_p regression*, SIAM J. Comput., 38 (2009), pp. 2060–2078.
[35] S. Dasgupta and A. Gupta, *An elementary proof of the Johnson–Lindenstrauss lemma*, Computer Science Dept. Tech. Report 99-006, Univ. California at Berkeley, Mar. 1999.
[36] A. d'Aspremont, *Subsampling algorithms for semidefinite programming.* Submitted. Available at `arXiv:0803.1990`, Apr. 2009.
[37] K. R. Davidson and S. J. Szarek, *Local operator theory, random matrices, and Banach spaces*, in Handbook of Banach Space Geometry, W. B. Johnson and J. Lindenstrauss, eds., Elsevier, 2002, pp. 317–366.
[38] J. Demmel, I. Dumitriu, and O. Holtz, *Fast linear algebra is stable*, Numer. Math., 108 (2007), pp. 59–91.

[39] A. Deshpande, L. Rademacher, S. Vempala, and G. Wang, *Matrix approximation and projective clustering via volume sampling*, in Proc. 17th Ann. ACM–SIAM Symp. Discrete Algorithms (SODA), 2006, pp. 1117–1126.

[40] A. Deshpande and S. Vempala, *Adaptive sampling and fast low-rank matrix approximation*, in Approximation, randomization and combinatorial optimization, vol. 4110 of LNCS, Springer, Berlin, 2006, pp. 292–303.

[41] J. D. Dixon, *Estimating extremal eigenvalues and condition numbers of matrices*, SIAM J. Numer. Anal., 20 (1983), pp. 812–814.

[42] J. Dongarra and F.Sullivan, *The top 10 algorithms*, Comput. Sci. Eng., (2000), pp. 22–23.

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

[44] D. L. Donoho, M. Vetterli, R. A. DeVore, and I. Daubechies, *Data compression and harmonic analysis*, IEEE Trans. Inform. Theory, 44 (1998), pp. 2433–2452.

[45] P. Drineas, R. Kannan, and M. W. Mahoney, *Fast Monte Carlo algorithms for matrices. I. Approximating matrix multiplication*, SIAM J. Comput., 36 (2006), pp. 132–157.

[46] ———, *Fast Monte Carlo algorithms for matrices. II. Computing a low-rank approximation to a matrix*, SIAM J. Comput., 36 (2006), pp. 158–183 (electronic).

[47] ———, *Fast Monte Carlo algorithms for matrices. III. Computing a compressed approximate matrix decomposition*, SIAM J. Comput., 36 (2006), pp. 184–206.

[48] P. Drineas and M. W. Mahoney, *A randomized algorithm for a tensor-based generalization of the singular value decomposition*, Linear Algebra Appl., 420 (2007), pp. 553–571.

[49] P. Drineas, M. W. Mahoney, and S. Muthukrishnan, *Subspace sampling and relative-error matrix approximation: Column-based methods*, in APPROX and RANDOM 2006, J. Diaz et al., ed., no. 4110 in LNCS, Springer, Berlin, 2006, pp. 321–326.

[50] ———, *Relative-error CUR matrix decompositions*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 844–881.

[51] P. Drineas, M. W. Mahoney, S. Muthukrishnan, and T. Sarlós, *Faster least squares approximation*. Submitted. Available at `arXiv:0710.1435`.

[52] A. Dvoretsky, *Some results on convex bodies and Banach spaces*, in Proc. Intl. Symp. Linear Spaces, Jerusalem, 1961, pp. 123–160.

[53] C. Eckart and G. Young, *The approximation of one matrix by another of lower rank*, Psychometrika, 1 (1936), pp. 211–218.

[54] A. Edelman, *Eigenvalues and condition numbers of random matrices*, Ph.D. thesis, Mathematics Dept., Massachusetts Inst. Tech., Boston, MA, May 1989.

[55] B. Engquist and O. Runborg, *Wavelet-based numerical homogenization with applications*, in Multiscale and Multiresolution Methods: Theory and Applications, T. J. Barth et al., ed., vol. 20 of LNCSE, Springer, Berlin, 2001, pp. 97–148.

[56] W. Feller, *An introduction to probability theory and its applications*, vol. 1, Wiley, New York, NY, 3rd ed., 1968.

[57] A. Frieze, R. Kannan, and S. Vempala, *Fast Monte Carlo algorithms for finding low-rank approximations*, in Proc. 39th Ann. IEEE Symp. Foundations of Computer Science (FOCS), 1998, pp. 370–378.

[58] ———, *Fast Monte Carlo algorithms for finding low-rank approximations*, J. ACM, 51 (2004), pp. 1025–1041. (electronic).

[59] A. Yu. Garnaev and E. D. Gluskin, *The widths of a Euclidean ball*, Dokl. Akad. Nauk. SSSR, 277 (1984), pp. 1048–1052. In Russian.

[60] A. Gittens and J. A. Tropp, *Error bounds for random matrix approximation schemes*. In preparation.

[61] G. H. Golub and C. F. van Loan, *Matrix Computations*, Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins Univ. Press, Baltimore, MD, 3rd ed., 1996.

[62] Y. Gordon, *Some inequalities for Gaussian processes and applications*, Israel J. Math., 50 (1985), pp. 265–289.

[63] ———, *Gaussian processes and almost spherical sections of convex bodies*, Ann. Probab., 16 (1988), pp. 180–188.

[64] S. A. Goreinov, E. E. Tyrtyshnikov, and N. L. Zamarashkin, *Theory of pseudo-skeleton matrix approximations*, Linear Algebra Appl., 261 (1997), pp. 1–21.

[65] L. Grasedyck and W. Hackbusch, *Construction and arithmetics of $\mathcal{H}$-matrices*, Computing, 70 (2003), pp. 295–334.

[66] L. Greengard and V. Rokhlin, *A new version of the fast multipole method for the Laplace equation in three dimensions*, Acta Numer., 17 (1997), pp. 229–269.

[67] M. Gu, 2007. Personal communication.

[68] M. Gu and S. C. Eisenstat, *Efficient algorithms for computing a strong rank-revealing QR factorization*, SIAM J. Sci. Comput., 17 (1996), pp. 848–869.

[69] S. Har-Peled, *Matrix approximation in linear time*. Manuscript. Available at `http://valis.cs.uiuc.edu/~sariel/research/papers/05/lrank/`, 2006.
[70] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, Berlin, 2nd ed., 2008.
[71] W. Hoeffding, *Probability inequalities for sums of bounded random variables*, J. Amer. Statist. Assoc., 58 (1963), pp. 13–30.
[72] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge Univ. Press, Cambridge, 1985.
[73] P. Indyk and R. Motwani, *Approximate nearest neighbors: Toward removing the curse of dimensionality*, in STOC '98: Proc. 30th Ann. ACM Symp. Theory of Computing, 1998, pp. 604–613.
[74] K. Jogdeo and S. M. Samuels, *Monotone convergence of binomial probabilities and a generalization of Ramanujan's equation*, Ann. Math. Statist., 39 (1968), pp. 1191–1195.
[75] W. B. Johnson and J. Lindenstrauss, *Extensions of Lipschitz mappings into a Hilbert space*, Contemp. Math., 26 (1984), pp. 189–206.
[76] D. R. Karger, *Random sampling in cut, flow, and network design problems*, Math. Oper. Res., 24 (1999), pp. 383–413.
[77] ———, *Minimum cuts in near-linear time*, J. ACM, 47 (2000), pp. 46–76.
[78] B. S. Kašin, *On the widths of certain finite-dimensional sets and classes of smooth functions*, Izv. Akad. Nauk. SSSR Ser. Mat., 41 (1977), pp. 334–351, 478. In Russian.
[79] J. Kleinberg, *Two algorithms for nearest neighbor search in high dimensions*, in STOC '97: Proc. 29th ACM Symp. Theory of Computing, 1997, pp. 599–608.
[80] J. Kuczyński and H. Woźniakowski, *Estimating the largest eigenvalue by the power and Lanczos algorithms with a random start*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 1094–1122.
[81] E. Kushilevitz, R. Ostrovski, and Y. Rabani, *Efficient search for approximate nearest neighbor in high-dimensional spaces*, SIAM J. Comput., 30 (2000), pp. 457–474.
[82] D. Le and D.S. Parker, *Using randomization to make recursive matrix algorithms practical*, 1999.
[83] M. Ledoux, *On Talagrand's deviation inequalities for product measures*, ESAIM Probab. Stat., 1 (1996), pp. 63–87.
[84] ———, *The Concentration of Measure Phenomenon*, no. 89 in MSM, AMS, Providence, RI, 2001.
[85] M. Ledoux and M. Talagrand, *Probability in Banach Spaces: Isoperimetry and Processes*, Springer, Berlin, 1991.
[86] W. S. Lee, P. L. Bartlett, and R. C. Williamson, *Efficient agnostic learning of neural networks with bounded fan-in*, IEEE Trans. Inform. Theory, 42 (1996), pp. 2118–2132.
[87] Z. Leyk and H. Woźniakowski, *Estimating the largest eigenvector by Lanczos and polynomial algorithms with a random start*, Num. Linear Algebra Appl., 5 (1998), pp. 147–164.
[88] E. Liberty, *Accelerated dense random projections*, Ph.D. thesis, Computer Science Dept., Yale University, New Haven, CT, 2009.
[89] E. Liberty, N. Ailon, and A. Singer, *Dense fast random projections and lean Walsh transforms*, in APPROX and RANDOM 2008, A. Goel et al., ed., no. 5171 in LNCS, Springer, Berlin, 2008, pp. 512–522.
[90] E. Liberty, F. F. Woolfe, P. G. Martinsson, V. Rokhlin, and M. Tygert, *Randomized algorithms for the low-rank approximation of matrices*, Proc. Natl. Acad. Sci. USA, 104 (2007), pp. 20167–20172.
[91] F. Lust-Picquard, *Inégalités de Khintchine dans C_p $(1 < p < \infty)$*, C. R. Math. Acad. Sci. Paris, 303 (1986), pp. 289–292.
[92] M. W. Mahoney and P. Drineas, *CUR matrix decompositions for improved data analysis*, Proc. Natl. Acad. Sci. USA, 106 (2009), pp. 697–702.
[93] P.G. Martinsson, V. Rokhlin, Y. Shkolnisky, and M. Tygert, *ID: A software package for low-rank approximation of matrices via interpolative decompositions, version 0.2*, 2008.
[94] P. G. Martinsson, V. Rokhlin, and M. Tygert, *A randomized algorithm for the approximation of matrices*, Computer Science Dept. Tech. Report 1361, Yale Univ., New Haven, CT, 2006.
[95] J. Matoušek, *Lectures on Discrete Geometry*, Springer, Berlin, 2002.
[96] F. McSherry, *Spectral Methods in Data Analysis*, Ph.D. thesis, Computer Science Dept., Univ. Washington, Seattle, WA, 2004.
[97] N. Metropolis and S. Ulam, *The Monte Carlo method*, J. Amer. Statist. Assoc., 44 (1949), pp. 335–341.
[98] V. D. Milman, *A new proof of A. Dvoretsky's theorem on cross-sections of convex bodies*, Funkcional. Anal. i Priložen, 5 (1971), pp. 28–37.

[99] L. Mirsky, *Symmetric gauge functions and unitarily invariant norms*, Quart. J. Math. Oxford Ser. (2), 11 (1960), pp. 50–59.
[100] R. J. Muirhead, *Aspects of Multivariate Statistical Theory*, Wiley, New York, NY, 1982.
[101] S. Muthukrishnan, *Data Streams: Algorithms and Applications*, Now Publ., Boston, MA, 2005.
[102] D. Needell, *Randomized Kaczmarz solver for noisy linear systems*. Submitted. Available at `arXiv:0902.0958`, Jan. 2009.
[103] N. H. Nguyen, T. T. Do, and T. D. Tran, *A fast and efficient algorithm for low-rank approximation of a matrix*, in STOC '09: Proc. 41st Ann. ACM Symp. Theory of Computing, 2009.
[104] C. H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala, *Latent semantic indexing: A probabilistic analysis*, in Proc. 17th ACM Symp. Principles of Database Systems (PODS), 1998.
[105] ———, *Latent semantic indexing: A probabilistic analysis*, J. Comput. System Sci., 61 (2000), pp. 217–235.
[106] D. S. Parker and B. Pierce, *The randomizing FFT: An alternative to pivoting in Gaussian elimination*, Computer Science Dept. Tech. Report CSD 950037, Univ. California at Los Angeles, 1995.
[107] P. J. Phillips, H. Moon, S.A. Rizvi, and P.J. Rauss, *The FERET evaluation methodology for face recognition algorithms*, IEEE Trans. Pattern Anal. Mach. Intelligence, 22 (2000), pp. 1090–1104.
[108] P. J. Phillips, H. Wechsler, J. Huang, and P. Rauss, *The FERET database and evaluation procedure for face recognition algorithms*, Image Vision Comput., 16 (1998), pp. 295–306.
[109] A. Rahimi and B. Recht, *Random features for large-scale kernel machines*, in Proc. 21st Ann. Conf. Advances in Neural Information Processing Systems (NIPS), 2007.
[110] B. Recht, M. Fazel, and P. Parillo, *Guaranteed minimum-rank solutions ot matrix equations via nuclear-norm minimization*, SIAM Rev., (2009). To appear. Available at `arXiv:0706.4138`.
[111] V. Rokhlin, A. Szlam, and M. Tygert, *A randomized algorithm for principal component analysis*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 1100–1124.
[112] 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.
[113] S. Roweis, *EM algorithms for PCA and SPCA*, in Proc. 10th Ann. Conf. Advances in Neural Information Processing Systems (NIPS), MIT Press, 1997, pp. 626–632.
[114] M. Rudelson, *Random vectors in the isotropic position*, J. Funct. Anal., 164 (1999), pp. 60–72.
[115] M. Rudelson and R. Vershynin, *Sparse reconstruction by convex relaxation: Fourier and Gaussian measurements*, in Proc. 40th Ann. Conf. Information Sciences and Systems (CISS), Mar. 2006.
[116] ———, *Sampling from large matrices: An approach through geometric functional analysis*, J. ACM, 54 (2007), pp. Art. 21, 19 pp. (electronic).
[117] A. F. Ruston, *Auerbach's theorem*, Math. Proc. Cambridge Philos. Soc., 56 (1964), pp. 476–480.
[118] T. Sarlós, *Improved approximation algorithms for large matrices via random projections*, in Proc. 47th Ann. IEEE Symp. Foundations of Computer Science (FOCS), 2006, pp. 143–152.
[119] S. Shalev-Shwartz and N. Srebro, *Low ℓ_1-norm and guarantees on sparsifiability*, in ICML/COLT/UAI Sparse Optimization and Variable Selection Workshop, July 2008.
[120] X. Shen and F.G. Meyer, *Low-dimensional embedding of fMRI datasets*, Neuroimage, 41 (2008), pp. 886–902.
[121] N. D. Shyamalkumar and K. Varadarajaran, *Efficient subspace approximation algorithms*, in Proc. 18th Ann. ACM–SIAM Symp. Discrete Algorithms (SODA), 2007, pp. 532–540.
[122] L. Sirovich and M. Kirby, *Low-dimensional procedure for the characterization of human faces.*, Journal of the Optical Society of America. A, Optics and image science, 4 (1987), pp. 519–524.
[123] D. Spielman and N. Srivastasa, *Graph sparsification by effective resistances*, in STOC '08: Proc. 40th Ann. ACM Symp. Theory of Computing, 2008.
[124] G. W. Stewart, *Perturbation of pseudo-inverses, projections, and linear least squares problems*, SIAM Rev., 19 (1977), pp. 634–662.
[125] ———, *Four algorithms for the efficient computation of truncated pivoted QR approximations to a sparse matrix*, Numer. Math., 83 (1999), pp. 313–323.
[126] ———, *The decompositional approach to matrix computation*, Comput. Sci. Eng., (2000),

pp. 50–59.
[127] T. Strohmer and R. Vershynin, *A randomized Kaczmarz algorithm with exponential convergence*, J. Fourier Anal. Appl., 15 (2009), pp. 262–278.
[128] J. Sun, Y. Xie, H. Zhang, and C. Faloutsos, *Less is more: Compact matrix decomposition for large sparse graphs*, Stat. Anal. Data Min., 1 (2008), pp. 6–22.
[129] S. J. Szarek, *Spaces with large distance from ℓ_∞^n and random matrices*, Amer. J. Math., 112 (1990), pp. 899–942.
[130] A. Szlam, M. Maggioni, and R.R. Coifman, *Regularization on graphs with function-adapted diffsion processes*, Journal of Machine Learning Research, 9 (2008), pp. 1711–1739.
[131] L. N. Trefethen and D. Bau III, *Numerical Linear Algebra*, SIAM, Philadelphia, PA, 1997.
[132] J. A. Tropp, *On the conditioning of random subdictionaries*, Appl. Comp. Harmon. Anal., 25 (2008), pp. 1–24.
[133] J. von Neumann and H. H. Goldstine, *Numerical inverting of matrices of high order*, Bull. Amer. Math. Soc., 53 (1947), pp. 1021–1099.
[134] ———, *Numerical inverting of matrices of high order. II*, Proc. AMS, 2 (1952), pp. 188–202.
[135] F. Woolfe, E. Liberty, V. Rokhlin, and M. Tygert, *A fast randomized algorithm for the approximation of matrices*, Appl. Comp. Harmon. Anal., 25 (2008), pp. 335–366.