DOI:10.1145/2842602

Randomization offers new benefits for large-scale linear algebra computations.

BY PETROS DRINEAS AND MICHAEL W. MAHONEY

RandNLA: Randomized Numerical Linear Algebra

MATRICES ARE UBIQUITOUS in computer science, statistics, and applied mathematics. An $m \times n$ matrix can encode information about m objects (each described by n features), or the behavior of a discretized differential operator on a finite element mesh; an $n \times n$ positive-definite matrix can encode the correlations between all pairs of n objects, or the edge-connectivity between all pairs of nodes in a social network; and so on. Motivated largely by technological developments that generate extremely large scientific and Internet datasets, recent years have witnessed exciting developments in the theory and practice of matrix algorithms. Particularly remarkable is the use of *randomization*—typically assumed to be a property of the input data due to, for example, noise in the data generation mechanisms—as an algorithmic or computational resource for the development of improved algorithms for fundamental matrix problems such as matrix multiplication, least-squares (LS) approximation, low-rank matrix approximation, and Laplacian-based linear equation solvers.

Randomized Numerical Linear Algebra (RandNLA) is an interdisciplinary research area that exploits randomization as a computational resource to develop improved algorithms for large-scale linear algebra problems.[32] From a foundational perspective, RandNLA has its roots in theoretical computer science (TCS), with deep connections to mathematics (convex analysis, probability theory, metric embedding theory) and applied mathematics (scientific computing, signal processing, numerical linear algebra). From an applied perspective, RandNLA is a vital new tool for machine learning, statistics, and data analysis. Well-engineered implementations have already outperformed highly optimized software libraries for ubiquitous problems such as least-squares,[4,35] with good scalability in parallel and distributed environments.[52] Moreover, RandNLA promises a sound algorithmic and statistical foundation for modern large-scale data analysis.

» key insights

- **Randomization isn't just used to model noise in data; it can be a powerful computational resource to develop algorithms with improved running times and stability properties as well as algorithms that are more interpretable in downstream data science applications.**
- **To achieve best results, random sampling of elements or columns/rows must be done carefully; but random projections can be used to transform or rotate the input data to a random basis where simple uniform random sampling of elements or rows/columns can be successfully applied.**
- **Random sketches can be used directly to get low-precision solutions to data science applications; or they can be used indirectly to construct preconditioners for traditional iterative numerical algorithms to get high-precision solutions in scientific computing applications.**

IMAGE BY FORANCE

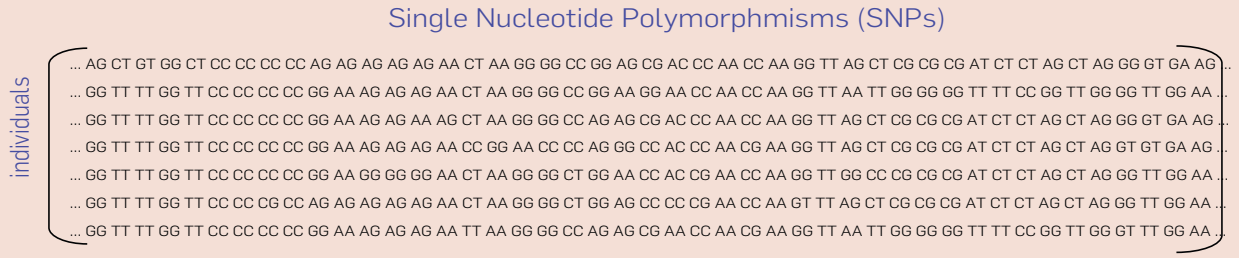
An Historical Perspective

To get a broader sense of RandNLA, recall that linear algebra—the mathematics of vector spaces and linear mappings between vector spaces—has had a long history in large-scale (by the standards of the day) statistical data analysis.[46] For example, the least-squares method is due to Gauss, Legendre, and others, and was used in the early 1800s for fitting linear equations to data to determine planet orbits. Low-rank approximations based on Principal Component Analysis (PCA) are due to Pearson, Hotelling, and others, and were used in the early 1900s for exploratory data analysis and for making predictive models. Such methods are of interest for many reasons, but especially if there is noise or randomness *in the data*, because the leading principal components then tend to capture the signal and remove the noise.

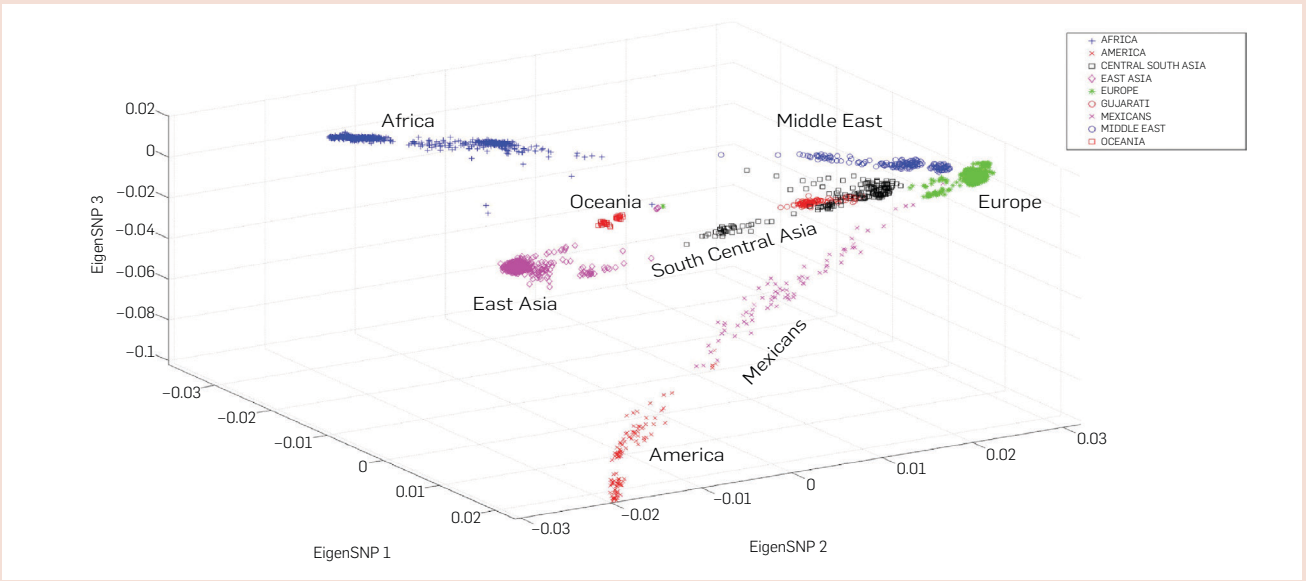
With the advent of the digital computer in the 1950s, it became apparent that, even when applied to well-posed problems, many algorithms performed poorly in the presence of the finite precision that was used to represent real numbers. Thus, much of the early work in computer science focused on solving discrete approximations to continuous numerical problems. Work by Turing and von Neumann (then Householder, Wilkinson, and others) laid much of the foundations for scientific computing and NLA.[48,49] Among other things, this led to the introduction of problem-specific complexity measures (for example, the condition number) that characterize the behavior of an input for a specific class of algorithms (for example, iterative algorithms).

A split then occurred in the nascent field of computer science. Continuous linear algebra became the domain of applied mathematics, and much of computer science theory and practice became discrete and combinatorial.[44] Nearly all subsequent work in scientific computing and NLA has been deterministic (a notable exception being the work on integral evaluation using the Markov Chain Monte Carlo method). This led to high-quality codes in the 1980s and 1990s (LINPACK, EISPACK, LAPACK, ScaLAPACK) that remain widely used today. Meanwhile, Turing, Church, and others began the study of computation per se. It became clear that several seemingly different approaches (recursion theory, the λ-calculus, and Turing machines) defined the same class of functions; and this led to the belief in TCS that the concept of computability is formally captured in a

Figure 1. (a) Matrices are a common way to model data. In genetics, for example, matrices can describe data from tens of thousands of individuals typed at millions of Single Nucleotide Polymorphisms or SNPs (loci in the human genome). Here, the (*i*, *j*)th entry is the genotype of the *i*th individual at the *j*th SNP. (b) PCA/SVD can be used to project every individual on the top left singular vectors (or "eigenSNPs"), thereby providing a convenient visualization of the "out of Africa hypothesis" well known in population genetics.



(a)



(b)

qualitative and robust way by these three equivalent approaches, *independent of the input data*. Many of these developments were deterministic; but, motivated by early work on the Monte Carlo method, randomization—where the randomness is *inside the algorithm* and the algorithm is applied to arbitrary or worst-case data—was introduced and exploited as a powerful computational resource.

Recent years have seen these two very different perspectives start to converge. Motivated by modern massive dataset problems, there has been a great deal of interest in developing algorithms with improved running times and/or improved statistical properties that are more appropriate for obtaining insight from the enormous quantities of noisy data that is now being generated. At the center of these developments is work on novel algorithms for linear algebra problems, and central to this is work on RandNLA algorithms.[a] In this article, we will describe the basic ideas that underlie recent developments in this interdisciplinary area.

For a prototypical data analysis example where RandNLA methods have been applied, consider Figure 1, which illustrates an application in genetics[38] (although the same RandNLA methods have been applied in astronomy, mass spectrometry imaging, and related areas[33,38,53,54]). While the low-dimensional PCA plot illustrates the famous correlation between geography and genetics, there are several weaknesses of PCA/SVD-based methods. One is running time: computing PCA/SVD approximations of even moderately large data matrices is expensive, especially if it needs to be done many times as part of cross validation or exploratory data analysis. Another is interpretability: in general, eigenSNPs (that is, eigenvectors of individual-by-SNP matrices) as well as other eigenfeatures don't "mean" anything in terms of the processes generating the data. Both issues have served as motivation to design RandNLA algorithms to compute PCA/SVD approximations faster than conventional numerical methods as well as to identify actual features (instead of eigenfeatures) that might be easier to interpret for domain scientists.

a Avron et al., in the first sentence of their Blendenpik paper, observe that RandNLA is "arguably the most exciting and innovative idea to have hit linear algebra in a long time."[4]

Basic RandNLA Principles

RandNLA algorithms involve taking an input matrix; constructing a "sketch" of that input matrix—where a sketch is a smaller or sparser matrix that represents the essential information in the original matrix—by random sampling; and then using that sketch as a surrogate for the full matrix to help compute quantities of interest. To be useful, the sketch should be similar to the original matrix in some way, for example, small residual error on the difference between the two matrices, or the two matrices should have similar action on sets of vectors or in downstream classification tasks. While these ideas have been developed in many ways, several basic design principles underlie much of RandNLA: (i) randomly sample, in a careful data-dependent manner, a small number of elements from an input matrix to create a much sparser sketch of the original matrix; (ii) randomly sample, in a careful data-dependent manner, a small number of columns and/or rows from an input matrix to create a much smaller sketch of the original matrix; and (iii) preprocess an input matrix with a random-projection-type matrix, in order to "spread out" or uniformize the information in the original matrix, and then use naïve data-independent uniform sampling of rows/columns/elements in order to create a sketch.

Element-wise sampling. A naïve way to view an $m \times n$ matrix A is an array of numbers: these are the mn elements of the matrix, and they are denoted by A_{ij} (for all $i = 1, \ldots, m$ and all $j = 1, \ldots, n$). It is therefore natural to consider the following approach in order to create a small sketch of a matrix A: instead of keeping all its elements, randomly sample and keep a small number of them. Algorithm 1 is a meta-algorithm that samples s elements from a matrix A in independent, identically distributed trials, where in each trial a single element of A is sampled with respect to the importance sampling probability distribution p_{ij}. The algorithm outputs a matrix $\tilde{A}$ that contains precisely the selected elements of A, after appropriate rescaling. This rescaling is fundamental from a statistical perspective: the sketch $\tilde{A}$ is an estimator for A. This rescaling makes it an unbiased estimator since, element-wise, the expectation of the estimator matrix $\tilde{A}$ is equal to the original matrix A.

Algorithm 1 A meta-algorithm for element-wise sampling

Input: $m \times n$ matrix A; integer $s > 0$ denoting the number of elements to be sampled; probability distribution p_{ij} ($i = 1, \ldots, m$ and $j = 1, \ldots, n$) with $\sum_{i,j} p_{ij} = 1$.

1. Let $\tilde{A}$ be an all-zeros $m \times n$ matrix.
2. **For** $t = 1$ **to** s,
   - **Randomly sample** one element of A using the probability distribution p_{ij}.
   - Let $A_{i_t j_t}$ denote the sampled element and set

$$\tilde{A}_{i_t j_t} := \tilde{A}_{i_t j_t} + \frac{1}{s p_{i_t j_t}} A_{i_t j_t}. \qquad (1)$$

Output: Return the $m \times n$ matrix $\tilde{A}$.

How to sample is, of course, very important. A simple choice is to perform uniform sampling, that is, set $p_{ij} = 1/mn$, for all i, j, and sample each element with equal probability. While simple, this suffers from obvious problems: for example, if all but one of the entries of the original matrix equal zero, and only a single non-zero entry exists, then the probability of sampling the single non-zero entry of A using uniform sampling is negligible. Thus, the estimator would have very large variance, in which case the sketch would, with high probability, fail to capture the relevant structure of the original matrix. Qualitatively improved results can be obtained by using nonuniform data-dependent importance sampling distributions. For example, sampling larger elements (in absolute value) with higher probability is advantageous in terms of variance reduction and can be used to obtain worst-case additive-error bounds for low-rank matrix approximation.[1,2,18,28] More elaborate probability distributions (the so-called element-wise leverage scores that use information in the singular subspaces of A[10]) have been shown to provide still finer results.

The first results for Algorithm 1[2] showed that if one chooses entries with probability proportional to their squared-magnitudes (that is, if $p_{ij} = A_{ij}^2 / \sum_{i,j} A_{ij}^2$, in which case larger magnitude entries are more likely to be chosen), then the sketch $\tilde{A}$ is similar to the original matrix A, in the sense that the error matrix, $A - \tilde{A}$, has, with high probability, a small spectral norm. A more refined analysis[18] showed that

$$\left\| A - \tilde{A} \right\|_2 \leq O\left(\sqrt{\frac{(m+n)\ln(m+n)}{s}} \right) \|A\|_F, \qquad (2)$$

where $\|\cdot\|_2$ and $\|\cdot\|_F$ are the spectral and Frobenius norms, respectively, of the matrix.[b] If the spectral norm of the difference $A - \tilde{A}$ is small, then $\tilde{A}$ can be used as proxy for A in applications. For example, one can use $\tilde{A}$ to approximate the spectrum (that is, the singular values and singular vectors) of the original matrix.[2] If s is set to be a constant multiple of $(m + n) \ln (m + n)$, then the error scales with the Frobenius norm of the matrix. This leads to an additive-error low-rank matrix approximation algorithm, in which $\|A\|_F$ is the scale of the additional additive error.[2] This is a large scaling factor, but improving upon this with element-wise sampling, even in special cases, is a challenging open problem.

The mathematical techniques used in the proof of these element-wise sampling results exploit the fact that the residual matrix $A - \tilde{A}$ is a random matrix whose entries have zero mean and bounded variance. Bounding the spectral norm of such matrices has a long history in random matrix theory.[50] Early RandNLA element-wise sampling bounds[2] used a result of Füredi and Komlós on the spectral norm of symmetric, zero mean matrices of bounded variance.[20] Subsequently, Drineas and Zouzias[18] introduced the idea of using matrix measure concentration inequalities[37,40,47] to simplify the proofs, and follow-up work[18] has improved these bounds.

Row/column sampling. A more sophisticated way to view a matrix A is as a linear operator, in which case the role of rows and columns becomes more central. Much RandNLA research has focused on sketching a matrix by keeping only a few of its rows and/or

b In words, the spectral norm of a matrix measures how much the matrix elongates or deforms the unit ball in the worst case, and the Frobenius norm measures how much the matrix elongates or deforms the unit ball on average. Sometimes the spectral norm may have better properties especially when dealing with noisy data, as discussed by Achlioptas and McSherry.[2]

columns. This method of sampling predates element-wise sampling algorithms,[19] and it leads to much stronger worst-case bounds.[15,16]

Algorithm 2 A meta-algorithm for row sampling

Input: $m \times n$ matrix A; integer $s > 0$ denoting the number of rows to be sampled; probabilities p_i ($i = 1, \ldots, m$) with $\sum_i p_i = 1$.

1. Let $\tilde{A}$ be the empty matrix.
2. **For** $t = 1$ **to** s,
   - **Randomly sample** one row of A using the probability distribution p_i.
   - Let A_{i_t*} denote the sampled row and set
$$\tilde{A}_{t*} := \frac{1}{\sqrt{sp_{i_t}}} A_{i_t*}. \quad (3)$$

Output: Return the $s \times n$ matrix $\tilde{A}$.

Consider the meta-algorithm for row sampling (column sampling is analogous) presented in Algorithm 2. Much of the discussion of Algorithm 1 is relevant to Algorithm 2. In particular, Algorithm 2 samples s rows of A in independent, identically distributed trials according to the input probabilities p_is; and the output matrix $\tilde{A}$ contains precisely the selected rows of A, after a rescaling that ensures un-biasedness of appropriate estimators (for example, the expectation of $\tilde{A}^T \tilde{A}$ is equal to $A^T A$, element-wise).[13,19] In addition, uniform sampling can easily lead to very poor results, but qualitatively improved results can be obtained by using nonuniform, data-dependent, importance sampling distributions. Some things, however, are different: the dimension of the sketch $\tilde{A}$ is different than that of the original matrix A. The solution is to measure the quality of the sketch by comparing the difference between the matrices $A^T A$ and $\tilde{A}^T \tilde{A}$. The simplest nonuniform distribution is known as ℓ_2 *sampling* or *norm-squared sampling*, in which p_i is proportional to square of the Euclidean norm of the ith row[c]:

$$p_i = \frac{\|A_{i*}\|_2^2}{\sum_i \|A_{i*}\|_2^2} = \frac{\|A_{i*}\|_2^2}{\|A\|_F^2}. \quad (4)$$

c We will use the notation A_{i*} to denote the ith row of A as a row vector.

Motivated by modern massive dataset problems, there has been a great deal of interest in developing algorithms with improved running times and/or improved statistical properties that are more appropriate for obtaining insight from the enormous quantities of noisy data now being generated.

When using norm-squared sampling, one can prove that

$$\|A^T A - \tilde{A}^T \tilde{A}\|_F \leq \frac{1}{\sqrt{s}} \|A\|_F^2 \quad (5)$$

holds in expectation (and thus, by standard arguments, with high probability) for arbitrary A.[13,19,d] The proof of Equation (5) is a simple exercise using basic properties of expectation and variance. This result can be generalized to approximate the product of two arbitrary matrices A and B.[13] Proving such bounds with respect to other matrix norms is more challenging but very important for RandNLA. While Equation (5) trivially implies a bound for $\|A^T A - \tilde{A}^T \tilde{A}\|_2$, proving a better spectral norm error bound necessitates the use of more sophisticated methods such as the Khintchine inequality or matrix-Bernstein inequalities.[42,47]

Bounds of the form of Equation (5) immediately imply that $\tilde{A}$ can be used as a proxy for A, for example, in order to approximate its (top few) singular values and singular vectors. Since $\tilde{A}$ is an $s \times n$ matrix, with $s \ll n$, computing its singular values and singular vectors is a very fast task that scales linearly with n. Due to the form of Equation (5), this leads to additive-error low-rank matrix approximation algorithms, in which $\|A\|_F$ is the scale of the additional additive error.[19] That is, while norm-squared sampling avoids pitfalls of uniform sampling, it results in additive-error bounds that are only comparable to what element-wise sampling achieves.[2,19]

To obtain stronger and more useful bounds, one needs information about the geometry or subspace structure of the high-dimensional Euclidean space spanned by the columns of A (if $m \gg n$) or the space spanned by the best rank-k approximation to A (if $m \sim n$). This can be achieved with *leverage score sampling*, in which p_i is proportional to

d That is, a provably good approximation to the product $A^T A$ can be computed using just a few rows of A; and these rows can be found by sampling randomly according to a simple data-dependent importance sampling distribution. This matrix multiplication algorithm can be implemented in one pass over the data from external storage, using only $O(sn)$ additional space and $O(s^2 n)$ additional time.

the ith leverage score of A. To define these scores, for simplicity assume that $m \gg n$ and that U is any $m \times n$ orthogonal matrix spanning the column space of A.[e] In this case, $U^T U$ is equal to the identity and $UU^T = P_A$ is an m-dimensional projection matrix onto the span of A. Then, the importance sampling probablities of Equation (4), applied to U, equal

$$p_i = \frac{\|U_{i*}\|_2^2}{\sum_i \|U_{i*}\|_2^2} = \frac{1}{n}(P_A)_{ii}. \quad (6)$$

Due to their historical importance in regression diagnostics and outlier detection, the p_i's in Equation (6) are known as *statistical leverage scores*.[9,14] In some applications of RandNLA, the largest leverage score is called the *coherence* of the matrix.[8,14]

Importantly, while one can naïvely compute these scores via Equation (6) by spending $O(mn^2)$ time to compute U exactly, this is *not* necessary.[14] Let Π be the fast Hadamard Transform as used in Drineas et al.[14] or the input-sparsity-time random projection of Refs.[12,34,36] Then, in $o(mn^2)$ time, one can compute the R matrix from a QR decomposition of ΠA and from that compute $1 \pm \epsilon$ relative-error approximations to all the leverage scores.[14]

In RandNLA, one is typically interested in proving that

$$\left\|U^T U - \tilde{U}^T \tilde{U}\right\|_2 = \left\|I - \tilde{U}^T \tilde{U}\right\|_2 \leq \epsilon, \quad (7)$$

either for arbitrary $\epsilon \in (0, 1)$ or for some fixed $\epsilon \in (0, 1)$. Approximate matrix multiplication bounds of the form of Equation (7) are very important in RandNLA algorithm design since the resulting sketch $\tilde{A}$ preserves rank properties of the original data matrix A and provides a *subspace embedding*: from the NLA perspective, this is simply an acute perturbation from the original high-dimensional space to a much lower dimensional space.[22] From the TCS perspective, this provides bounds analogous to the usual Johnson–Lindenstrauss bounds, except that it preserves the geometry of the entire subspace.[43]

e A generalization holds if $m \sim n$: in this case, U is any $m \times k$ orthogonal matrix spanning the best rank-k approximation to the column space of A, and one uses the *leverage scores relative to the best rank-k approximation to A*.[14,16,33]

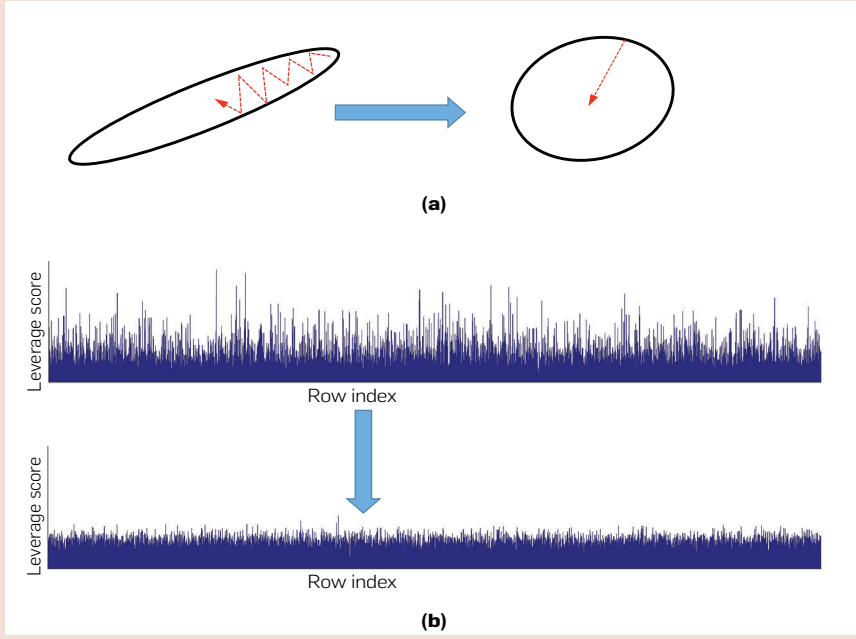
Subspace embeddings were first used in RandNLA in a data-aware manner (meaning, by looking at the input data to compute exact or approximate leverage scores[14]) to obtain sampling-based relative-error approximation to the LS regression and related low-rank CX/CUR approximation problems.[15,16] They were then used in a data-oblivious manner (meaning, in conjuction with a random projection as a preconditioner) to obtain projection-based relative-error approximation to several RandNLA problems.[43] A review of data-oblivious subspace embeddings for RandNLA, including its relationship with the early work on least absolute deviations regression,[11] has been provided.[51] Due to the connection with data-aware and data-oblivious subspace embeddings, approximating matrix multiplication is one of most powerful primitives in RandNLA. Many error formulae for other problems ultimately boil down to matrix inequalities, where the randomness of the algorithm only appears as a (randomized) approximate matrix multiplication.

Random projections as preconditioners. Preconditioning refers to the application of a transformation, called the preconditioner, to a given problem instance such that the transformed instance is more easily solved by a given class of algorithms.[f] The main challenge for sampling-based RandNLA algorithms is the construction of the nonuniform sampling probabilities. A natural question arises: is there a way to precondition an input instance such that *uniform random sampling* of rows, columns, or elements yields an insignificant loss in approximation accuracy?

The obvious obstacle to sampling uniformly at random from a matrix is that the relevant information in the matrix could be concentrated on a small number of rows, columns, or elements of the matrix. The solution is to *spread out* or *uniformize* this information, so that it is distributed almost uniformly over all rows, columns, or elements of the matrix. (This is illustrated in Figure 2.) At the same time, the preprocessed

f For example, if one is interested in iterative algorithms for solving the linear system $Ax = b$, one typically transforms a given problem instance to a related instance in which the so-called condition number is not too large.

Figure 2. In RandNLA, random projections can be used to "precondition" the input data so that uniform sampling algorithms perform well, in a manner analogous to how traditional pre-conditioners transform the input to decrease the usual condition number so that iterative algorithms perform well (see (a)). In RandNLA, the random projection-based preconditioning involves uniformizing information in the eigenvectors, rather than flattening the eigenvalues (see (b)).



matrix should have similar properties (for example, singular values and singular vectors) as the original matrix, and the preprocessing should be computationally efficient (for example, it should be faster than solving the original problem exactly) to perform.

Consider Algorithm 3, our meta-algorithm for preprocessing an input matrix A in order to uniformize information in its rows or columns or elements. Depending on the choice of preprocessing (only from the left, only from the right, or from both sides) the information in A is uniformized in different ways (across its rows, columns, or elements, respectively). For pedagogical simplicity, Algorithm 3 is described such that the output matrix has the same dimensions as the original matrix (in which case Π is approximately a *random rotation*). Clearly, however, if this algorithm is coupled with Algorithm 1 or Algorithm 2, then with trivial to implement uniform sampling, only the rows/columns that are sampled actually need to be generated. In this case the sampled version of Π is known as a *random projection*.

Algorithm 3 A meta-algorithm for preconditioning a matrix for random sampling algorithms

1: **Input:** $m \times n$ matrix A, randomized preprocessing matrices Π_L and/or Π_R.
2: **Output:**
- To uniformize information across the rows of A, return $\Pi_L A$.
- To uniformize information across the columns of A, return $A\Pi_R$.
- To uniformize information across the elements of A, return $\Pi_L A \Pi_R$.

There is wide latitude in the choice of the random matrix Π. For example, although Π can be chosen to be a random orthogonal matrix, other constructions can have much better algorithmic properties: Π can consist of appropriately-scaled independent identically distributed (i.i.d.) Gaussian random variables, i.i.d. Rademacher random variable (+1 or –1, up to scaling, each with probability 50%), or i.i.d. random variables drawn from any sub-Gaussian distribution. Implementing these variants depends on the time to generate the random bits plus the time to perform the matrix-matrix multiplication that actually performs the random projection.

More interestingly, Π could be a so-called Fast Johnson Lindenstrauss Transform (FJLT). This is the product of two matrices, a random diagonal matrix with +1 or –1 on each diagonal entry, each with probability 1/2, and the Hadamard-Walsh (or related Fourier-based) matrix.[3] Implementing FJLT-based random projections can take advantage of well-studied fast Fourier techniques and can be extremely fast for arbitrary dense input matrices.[4,41] Recently, there has even been introduced an extremely sparse random projection construction that for arbitrary input matrices can be implemented in "input-sparsity time," that is, time depending on the number of nonzeros of A, plus lower-order terms, as opposed to the dimensions of A.[12,34,36]

With appropriate settings of problem parameters (for example, the number of uniform samples that are subsequently drawn, which equals the dimension onto which the data is projected), all of these methods precondition arbitrary input matrices so that uniform sampling in the randomly rotated basis performs as well as nonuniform sampling in the original basis. For example, if $m \gg n$, in which case the leverage scores of A are given by Equation (6), then by keeping only roughly $O(n \log n)$ randomly-rotated dimensions, uniformly at random, one can prove that the leverage scores of the preconditioned system are, up to logarithmic fluctuations, uniform.[g] Which construction for Π should be used in any particular application of RandNLA depends on the details of the problem, for example, the aspect ratio of the matrix, whether the RAM model is appropriate for the particular computational infrastructure, how expensive it is to generate random bits, and so on. For example, while slower in the RAM model, Gaussian-based random projections can have stronger conditioning properties than other constructions. Thus, given their ease of use, they are often more appropriate for certain parallel and cloud-computing architectures.[25,35]

Summary. Of the three basic RandNLA principles described in this section, the first two have to do with identifying nonuniformity structure in the input data; and the third has to do with preconditioning the input (that is, uniformizing the nonuniformity structure) so uniform random sampling performs well. Depending on the area in which RandNLA algorithms have been developed and/or implemented and/or applied, these principles can manifest themselves in very different ways. Relatedly, in applications where elements are of primary importance (for example, recommender systems[26]), element-wise methods might be most appropriate, while in applications where subspaces are of primary importance (for example, scientific computing[25]), column/row-based methods might be most appropriate.

Extensions and Applications of Basic RandNLA Principles

We now turn to several examples of problems in various domains where the basic RandNLA principles have been used in the design and analysis, implementation, and application of novel algorithms.

Low-precision approximations and high-precision numerical implementations: least-squares and low-rank approximation. One of the most fundamental problems in linear algebra is the least-squares (LS) regression problem: given an $m \times n$ matrix A and an m-dimensional vector b, solve

$$x_{opt} = \arg\min_{x \in \mathbb{R}^n} \|Ax - b\|_2, \qquad (8)$$

where $\|\cdot\|_2$ denotes the ℓ_2 norm of a vector. That is, compute the n-dimensional vector x that minimizes the Euclidean norm of the residual $Ax - b$.[h] If $m \gg n$, then we have the overdetermined (or overconstrained) LS problem, and its solution can be obtained in $O(mn^2)$ time in the RAM model with one of several methods, for example, solving the normal equations, QR decompositions, or the SVD. Two major successes of RandNLA concern faster (in terms of low-precision asymptotic worst-case theory, or in terms of high-precision wall-clock time) algorithms for this ubiquitous problem.

g This is equivalent to the statement that the coherence of the preconditioned system is small.

h Observe this formulation includes as a special case the problem of solving systems of linear equations (if $m = n$ and A has full rank, then the resulting system of linear equations has a unique solution).

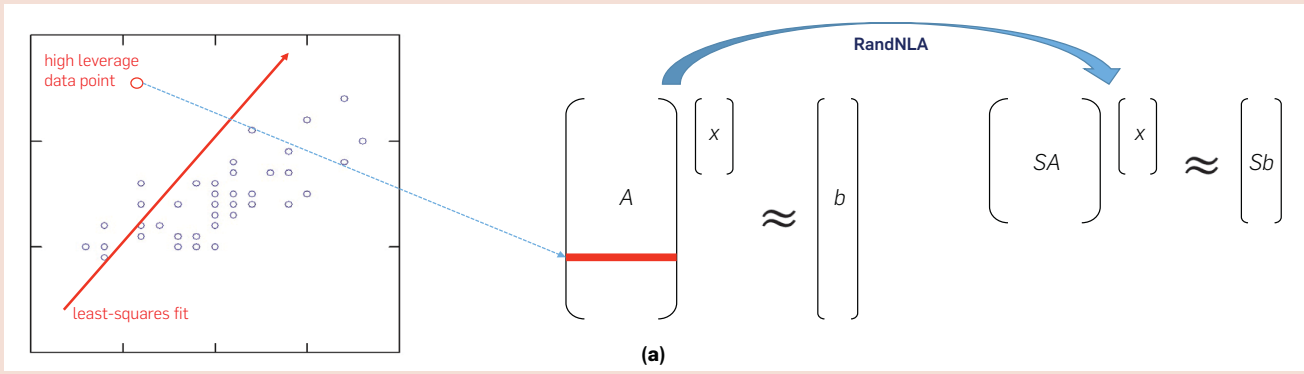
One major success of RandNLA was the following random sampling algorithm for the LS problem: quickly compute $1 \pm \epsilon$ approximations to the leverage scores;[14] form a subproblem by sampling with Algorithm 2 roughly $\Theta(n \log(m)/\epsilon)$ rows from A and the corresponding elements from b using those approximations as importance sampling probabilities; and return the LS solution of the subproblem.[14,15] Alternatively, one can run the following random projection algorithm: precondition the input with a Hadamard-based random projection; form a subproblem by sampling with Algorithm 2 roughly $\Theta(n \log(m)/\epsilon)$ rows from A and the corresponding elements from b uniformly at random; and return the LS solution of the subproblem.[17,43]

Both of these algorithms return $1\pm\epsilon$ relative-error approximate solutions for arbitrary or worst-case input; and both run in roughly $\Theta(mn \log(n)/\epsilon) = o(mn^2)$ time, that is, qualitatively faster than traditional algorithms for the overdetermined LS problem. (Although this random projection algorithm is not faster in terms of asymptotic FLOPS than the corresponding random sampling algorithm, preconditioning with random projections is a powerful primitive more generally for RandNLA algorithms.) Moreover, both of these algorithms have been improved to run in time that is proportional to the number of nonzeros in the matrix, plus lower-order terms that depend on the lower dimension of the input.[12]

Another major success of RandNLA was the demonstration that the sketches constructed by RandNLA could be used to construct preconditioners for high-quality traditional NLA iterative software libraries.[4] To see the need for this, observe that because of its dependence on ϵ, the previous RandNLA algorithmic strategy (construct a sketch and solve a LS problem on that sketch) can yield low-precision solutions, for example, $\epsilon = 0.1$, but cannot practically yield high-precision solutions, for example, $\epsilon = 10^{-16}$. Blendenpik[4] and LSRN[35] are LS solvers that are appropriate for RAM and parallel environments, respectively, that adopt the following RandNLA algorithmic strategy: construct a sketch, using an appropriate random projection; use that sketch to construct a preconditioner for a traditional iterative NLA algorithm; and use that to solve the preconditioned version of the original full problem. This improves the ϵ dependence from poly$(1/\epsilon)$ to log$(1/\epsilon)$. Carefully-engineered implementations of this approach are competitive with or beat high-quality numerical implementations of LS solvers such as those implemented in LAPACK.[4]

The difference between these two algorithmic strategies (see Figure 3 for an illustration) highlights important differences between TCS and NLA approaches to RandNLA, as well as between computer science and scientific computing more generally: subtle but important differences in problem parameterization, between what counts as a "good" solution, and between error norms of interest. Moreover, similar approaches have been used to extend TCS-style RandNLA algorithms for providing $1 \pm \epsilon$ relative-error low-rank matrix approximation[16,43] to NLA-style RandNLA algorithms for high-quality numerical low-rank matrix approximation.[24,25,41]

Figure 3. (a) RandNLA algorithms for least-squares problems first compute sketches, *SA* and *Sb*, of the input data, *A* and *b*. Then, either they solve a least-squares problem on the sketch to obtain a low-precision approximation, or they use the sketch to construct a traditional preconditioner for an iterative algorithm on the original input data to get high-precision approximations. Subspace-preserving embedding: if *S* is a random sampling matrix, then the high leverage point will be sampled and included in *SA*; and if *S* is a random-projection-type matrix, then the information in the high leverage point will be homogenized or uniformized in *SA*. (b) The "heart" of RandNLA proofs is subspace-preserving embedding for orthogonal matrices: if U_A is an orthogonal matrix (say the matrix of the left singular vectors of *A*), then SU_A is approximately orthogonal.



For example, a fundamental structural condition for a sketching matrix to satisfy to obtain good low-rank matrix approximation is the following. Let $V_k \in \mathbb{R}^{n\times k}$ (resp., $V_{k,\perp} \in \mathbb{R}^{n\times(n-k)}$) be any matrix spanning the top-k (resp., bottom-$(n-k)$) right singular subspace of $A \in \mathbb{R}^{m\times n}$, and let Σ_k (resp., $\Sigma_{k,\perp}$) be the diagonal matrix containing the top-k (resp., all but the top-k) singular values. In addition, let $Z \in \mathbb{R}^{n\times r}$ ($r \geq k$) be *any* matrix (for example, a random sampling matrix S, a random projection matrix Π, or a matrix Z constructed deterministically) such that $V_k^T Z$ has full rank. Then,

$$\left\|A - P_{AZ}A\right\| \leq \left\|\Sigma_k\right\| + \left\|\Sigma_{k,\perp}\left(V_{k,\perp}^T Z\right)\left(V_k^T Z\right)^+\right\|, \quad (9)$$

where $\|\cdot\|$ is any unitarily invariant matrix norm.

How this structural condition is used depends on the particular low-rank problem of interest, but it is widely used (either explicitly or implicitly) by low-rank RandNLA algorithms. For example, Equation (9) was introduced in the context of the Column Subset Selection Problem[7] and was reproven and used to reparameterize low-rank random projection algorithms in ways that could be more easily implemented.[25] It has also been used in ways ranging from developing improved bounds for kernel methods in machine learning[21] to coupling with a version of the power method to obtain improved numerical implementations[41] to improving subspace iteration methods.[24]

The structural condition in Equation (9) immediately suggests a proof strategy for bounding the error of RandNLA algorithms for low-rank matrix approximation: identify a sketching matrix Z such that $V_k^T Z$ has full rank; and, at the same time, bound the relevant norms of $\left(V_k^T Z\right)^+$ and $V_{k,\perp}^T Z$. Importantly, in many of the motivating scientific computing applications, the matrices of interest are linear operators that are only implicitly represented but that are structured such that they can be applied to an arbitrary vector quickly. In these cases, FJLT-based or input-sparsity-based projections applied to arbitrary matrices can be replaced with Gaussian-based projections applied to these structured operators with similar computational costs and quality guarantees.

Matrix completion. Consider the following problem, which is an idealization of the important recommender systems problem.[26] Given an arbitrary $m \times n$ matrix A, reconstruct A by sampling a set of $O((m+n)\text{poly}(1/\epsilon^\alpha))$, as opposed to all mn, entries of the matrix such that the resulting approximation $\tilde{A}$ satisfies, either deterministically or up to some failure probability,

$$\left\|A - \tilde{A}\right\|_F \leq (1+\epsilon)\left\|A - A_k\right\|_F. \quad (10)$$

Here, α should be small (for example, 2); and the sample size could be increased by (less important) logarithmic factors of m, n, and ϵ. In addition, one would like to construct the sample and compute $\tilde{A}$ after making a small number of passes over A or without even touching all of the entries of A.

A first line of research (already mentioned) on this problem from TCS focuses on element-wise sampling:[2] sample entries from a matrix with probabilities that (roughly) depend on their magnitude squared. This can be done in one pass over the matrix, but the resulting additive-error bound is much larger than the requirements of Equation (10), as it scales with the Frobenius norm of A instead of the Frobenius norm of $A - A_k$.

A second line of research from signal processing and applied mathematics has referred to this as the *matrix completion problem*.[8] In this case, one is interested in computing $\tilde{A}$ without even observing all of the entries of A. Clearly, this is not possible without assumptions on A.[i] Typical assumptions are on the eigenvalues and eigenvectors of A: for example, the input matrix A has rank exactly k, with $k \ll \min\{m, n\}$, and also that A satisfies some sort of *eigenvector delocalization* or *incoherence conditions*.[8] The simplest form of the latter is the leverage scores of Equation (6) are approximately uniform. Under these assumptions, one can prove that given a uniform sample of $O((m+n)\,k\ln(m+n))$ entries of A, the solution to the following nuclear norm minimization problem recovers A exactly, with high probability:

$$\min_{\tilde{A}\in\mathbb{R}^{m\times n}} \quad \left\|\tilde{A}\right\|_* \quad (11)$$

$$\text{s.t.} \quad \tilde{A}_{ij} = A_{ij}, \quad \text{for all sampled entries } A_{ij},$$

where $\|\cdot\|_*$ denotes the nuclear (or trace) norm of a matrix (basically, the sum of the singular values of the matrix). That is, if A is exactly low-rank (that is, $A = A_k$ and thus $A - A_k$ is zero) and satisfies an incoherence assumption, then Equation (10) is satisfied, since $A = A_k = \tilde{A}$. Recently, the incoherence assumption has been relaxed, under the assumption that one is given oracle access to A according to a non-uniform sampling distribution that essentially corresponds to *element-wise leverage scores*.[10] However, removing the assumption that A has exact low-rank k, with $k \ll \min\{m, n\}$, is still an open problem.[j]

Informally, keeping only a few rows/columns of a matrix seems more powerful than keeping a comparable number of elements of a matrix. For example, consider an $m \times n$ matrix A whose rank is exactly equal to k, with $k \ll \min\{m, n\}$: selecting any set of k linearly independent rows allows every row of A to be expressed as a linear combination of the selected rows. The analogous procedure for element-wise sampling seems harder. This is reflected in that state-of-the-art element-wise sampling algorithms use convex optimization and other heavier-duty algorithmic machinery.

Solving systems of Laplacian-based linear equations. Consider the special case of the LS regression problem of Equation (8) when $m = n$, that is, the well-known problem of solving the system of linear equations $Ax = b$. For worst-case dense input matrices A this problem can be solved in exactly $O(n^3)$ time, for example, using the partial LU decomposition and other methods. However, especially when A is positive semidefinite (PSD), iterative techniques such as the conjugate gradients method are typically preferable, mainly because of their linear dependency on the number of non-zero entries in the matrix A (times a factor depending on the condition number of A).

An important special case is when the PSD matrix A is the Laplacian matrix

i This highlights an important difference in problem parameterization: TCS-style approaches assume worst-case input and must identify nonuniformity strucutre, while applied mathematics approaches typically assume well-posed problems where the worst nonuniformity structure is not present.

j It should be noted that there exists prior work on matrix completion for low-rank matrices with the addition of well-behaved noise; however, removing the low-rank assumption and achieving error that is relative to some norm of the residual $A - A_k$ is still open.

of an underlying undirected graph $G = (V, E)$, with $n = |V|$ vertices and $|E|$ weighted, undirected edges.[5] Variants of this special case are common in unsupervised and semi-supervised machine learning.[6] Recall the Laplacian matrix of an undirected graph G is an $n \times n$ matrix that is equal to the $n \times n$ diagonal matrix D of node weights minus the $n \times n$ adjacency matrix of the graph. In this special case, there exist randomized, relative-error algorithms for the problem of Equation (8).[5] The running time of these algorithms is

$$O(\mathbf{nnz}(A)\text{polylog}(n)),$$

where $\mathbf{nnz}(A)$ represents the number of non-zero elements of the matrix A, that is, the number of edges in the graph G. The first step of these algorithms corresponds to randomized graph sparsification and keeps a small number of edges from G, thus creating a much sparser Laplacian matrix $\tilde{L}$. This sparse matrix $\tilde{L}$ is subsequently used (in a recursive manner) as an efficient preconditioner to approximate the solution of the problem of Equation (8).

While the original algorithms in this line of work were major theoretical breakthroughs, they were not immediately applicable to numerical implementations and data applications. In an effort to bridge the theory-practice gap, subsequent work proposed a much simpler algorithm for the graph sparsification step.[45] This subsequent work showed that randomly sampling edges from the graph G (equivalently, rows from the edge-incidence matrix) with probabilities proportional to the *effective resistances* of the edges provides a sparse Laplacian matrix $\tilde{L}$ satisfying the desired properties. (On the negative side, in order to approximate the effective resistances of the edges of G, a call to the original solver was necessary, clearly hindering the applicability of the simpler sparsification algorithm.[45]) The effective resistances are equivalent to the statistical leverage scores of the weighted edge-incidence matrix of G. Subsequent work has exploited graph theoretic ideas to provide efficient algorithms to approximate them in time proportional to the number of edges in the graph (up to polylogarithmic factors).[27] Recent improvements have essentially removed these polylogarithmic factors, leading to useful implementations of Laplacian-based solvers.[27] Extending such techniques to handle general PSD input matrices A that are not Laplacian is an open problem.

RandNLA has proven to be a model for truly interdisciplinary research in this era of large-scale data.

Statistics and machine learning. RandNLA has been used in statistics and machine learning in several ways, the most common of which is in the so-called kernel-based machine learning.[21] This involves using a PSD matrix to encode nonlinear relationships between data points; and one obtains different results depending on whether one is interested in approximating a given kernel matrix,[21] constructing new kernel matrices of particular forms,[39] or obtaining a low-rank basis with which to perform downstream classification, clustering, and other related tasks.[29] Alternatively, the analysis used to provide relative-error low-rank matrix approximation for worst-case input can also be used to provide bounds for kernel-based divide-and-conquer algorithms.[31] More generally, CX/CUR decompositions provide scalable and interpretable solutions to downstream data analysis problems in genetics, astronomy, and related areas.[33,38,53,54] Recent work has focused on statistical aspects of the "algorithmic leveraging" approach that is central to RandNLA algorithms.[30]

Looking Forward

RandNLA has proven to be a model for truly interdisciplinary research in this era of large-scale data. For example, while TCS, NLA, scientific computing, mathematics, machine learning, statistics, and downstream scientific domains are all interested in these results, each of these areas is interested for very different reasons. Relatedly, while technical results underlying the development of RandNLA have been nontrivial, some of the largest obstacles to progress in RandNLA have been cultural: TCS being cavalier about polynomial factors, ϵ factors, and working in overly idealized computational models; NLA being extremely slow to embrace randomization as an algorithmic resource; scientific computing researchers formulating and implementing algorithms that make strong domain-specific assumptions; and machine learning and statistics researchers being more interested in results on hypoth-

esized unseen data rather than the data being input to the algorithm.

In spite of this, RandNLA has already led to improved algorithms for several fundamental matrix problems, but it is important to emphasize that "improved" means different things to different people. For example, TCS is interested in these methods due to the deep connections with Laplacian-based linear equation solvers[5,27] and since fast random sampling and random projection algorithms[12,14,17,43] represent an improvement in the asymptotic running time of the 200-year-old Gaussian elimination algorithms for least-squares problems on worst-case input. NLA is interested in these methods since they can be used to engineer variants of traditional NLA algorithms that are more robust and/or faster in wall clock time than high-quality software that has been developed over recent decades. (For example, Blendenpik "beats LAPACK's direct dense least-squares solver by a large margin on essentially any dense tall matrix;"[4] the randomized approach for low-rank matrix approximation in scientific computing "beats its classical competitors in terms of accuracy, speed, and robustness;"[25] and least-squares and least absolute deviations regression problems "can be solved to low, medium, or high precision in existing distributed systems on up to terabyte-sized data."[52]) Mathematicians are interested in these methods since they have led to new and fruitful fundamental mathematical questions.[23,40,42,47] Statisticians and machine learners are interested in these methods due to their connections with kernel-based learning and since the randomness inside the algorithm often implicitly implements a form of regularization on realistic noisy input data.[21,29,30] Finally, data analysts are interested in these methods since they provide scalable and interpretable solutions to downstream scientific data analysis problems.[33, 38,54] Given the central role that matrix problems have historically played in large-scale data analysis, we expect RandNLA methods will continue to make important contributions not only to each of those research areas but also to bridging the gaps between them. C

References

1. Achlioptas, D., Karnin, Z., Liberty, E. Near-optimal entrywise sampling for data matrices. In *Annual Advances in Neural Information Processing Systems 26: Proceedings of the 2013 Conference*, 2013.
2. Achlioptas, D., McSherry, F. Fast computation of low-rank matrix approximations. *J. ACM 54*, 2 (2007), Article 9.
3. Ailon, N., Chazelle, B. Faster dimension reduction. *Commun. ACM 53*, 2 (2010), 97–104.
4. Avron, H., Maymounkov, P., Toledo, S. Blendenpik: Supercharging LAPACK's least-squares solver. *SIAM J. Sci. Comput. 32* (2010), 1217–1236.
5. Batson, J., Spielman, D.A., Srivastava, N., Teng, S.-H. Spectral sparsification of graphs: Theory and algorithms. *Commun. ACM 56*, 8 (2013), 87–94.
6. Belkin, M., Niyogi, P. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput. 15*, 6 (2003), 1373–1396.
7. Boutsidis, C., Mahoney, M.W., Drineas, P. An improved approximation algorithm for the column subset selection problem. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms* (2009), 968–977.
8. Candes, E.J., Recht, B. Exact matrix completion via convex optimization. *Commun. ACM 55*, 6 (2012), 111–119.
9. Chatterjee, S., Hadi, A.S. Influential observations, high leverage points, and outliers in linear regression. *Stat. Sci. 1*, 3 (1986), 379–393.
10. Chen, Y., Bhojanapalli, S., Sanghavi, S., Ward, R. Coherent matrix completion. In *Proceedings of the 31st International Conference on Machine Learning* (2014), 674–682.
11. Clarkson, K. Subgradient and sampling algorithms for ℓ_1 regression. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms* (2005), 257–266.
12. Clarkson, K.L., Woodruff, D.P. Low rank approximation and regression in input sparsity time. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing* (2013), 81–90.
13. Drineas, P., Kannan, R., Mahoney, M.W. Fast Monte Carlo algorithms for matrices I: approximating matrix multiplication. *SIAM J. Comput. 36* (2006), 132–157.
14. Drineas, P., Magdon-Ismail, M., Mahoney, M.W., Woodruff, D.P. Fast approximation of matrix coherence and statistical leverage. *J. Mach. Learn. Res. 13* (2012), 3475–3506.
15. Drineas, P., Mahoney, M.W., Muthukrishnan, S. Sampling algorithms for ℓ_2 regression and applications. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms* (2006), 1127–1136.
16. Drineas, P., Mahoney, M.W., Muthukrishnan, S. Relative-error CUR matrix decompositions. *SIAM J. Matrix Anal. Appl. 30* (2008), 844–881.
17. Drineas, P., Mahoney, M.W., Muthukrishnan, S., Sarlós, T. Faster least squares approximation. *Numer. Math. 117*, 2 (2010), 219–249.
18. Drineas, P., Zouzias, A. A note on element-wise matrix sparsification via a matrix-valued Bernstein inequality. *Inform. Process. Lett. 111* (2011), 385–389.
19. Frieze, A., Kannan, R., Vempala, S. Fast Monte-Carlo algorithms for finding low-rank approximations. *J. ACM 51*, 6 (2004), 1025–1041.
20. Füredi, Z., Komlós, J. The eigenvalues of random symmetric matrices. *Combinatorica 1*, 3 (1981), 233–241.
21. Gittens, A. Mahoney, M.W. Revisiting the Nyström method for improved large-scale machine learning. *J. Mach. Learn Res*. In press.
22. Golub, G.H., Van Loan, C.F. *Matrix Computations*. Johns Hopkins University Press, Baltimore, 1996.
23. Gross, D. Recovering low-rank matrices from few coefficients in any basis. *IEEE Trans. Inform. Theory 57*, 3 (2011), 1548–1566.
24. Gu, M. Subspace iteration randomization and singular value problems. Technical report, 2014. Preprint: arXiv:1408.2208.
25. Halko, N., Martinsson, P.-G., Tropp, J.A. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev. 53*, 2 (2011), 217–288.
26. Koren, Y., Bell, R., Volinsky, C. Matrix factorization techniques for recommender systems. *IEEE Comp. 42*, 8 (2009), 30–37.
27. Koutis, I., Miller, G.L., Peng, R. A fast solver for a class of linear systems. *Commun. ACM 55*, 10 (2012), 99–107.
28. Kundu, A., Drineas, P. A note on randomized element-wise matrix sparsification. Technical report, 2014. Preprint: arXiv:1404.0320.
29. Le, Q.V., Sarlós, T., Smola, A.J. Fastfood—approximating kernel expansions in loglinear time. In *Proceedings of the 30th International Conference on Machine Learning*, 2013.
30. Ma, P., Mahoney, M.W., Yu, B. A statistical perspective on algorithmic leveraging. *J. Mach. Learn. Res. 16* (2015), 861–911.
31. Mackey, L., Talwalkar, A., Jordan, M.I. Distributed matrix completion and robust factorization. *J. Mach. Learn. Res. 16* (2015), 913–960.
32. Mahoney, M.W. *Randomized Algorithms for Matrices and Data*. Foundations and Trends in Machine Learning. NOW Publishers, Boston, 2011.
33. Mahoney, M.W., Drineas, P. CUR matrix decompositions for improved data analysis. *Proc. Natl. Acad. Sci. USA 106* (2009), 697–702.
34. Meng, X., Mahoney, M.W. Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing* (2013), 91–100.
35. Meng, X., Saunders, M.A., Mahoney, M.W. LSRN: A parallel iterative solver for strongly over- or under-determined systems. *SIAM J. Sci. Comput. 36*, 2 (2014), C95–C118.
36. Nelson, J., Huy, N.L. OSNAP: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science* (2013), 117–126.
37. Oliveira, R.I. Sums of random Hermitian matrices and an inequality by Rudelson. *Electron. Commun. Prob. 15* (2010) 203–212.
38. Paschou, P., Ziv, E., Burchard, E.G., Choudhry, S., Rodriguez-Cintron, W., Mahoney, M.W., Drineas, P. PCA-correlated SNPs for structure identification in worldwide human populations. *PLoS Genet. 3* (2007), 1672–1686.
39. Rahimi, A., Recht, B. Random features for large-scale kernel machines. In *Annual Advances in Neural Information Processing Systems 20: Proceedings of the 2007 Conference*, 2008.
40. Recht, B. A simpler approach to matrix completion. *J. Mach. Learn. Res. 12* (2011), 3413–3430.
41. Rokhlin, V., Szlam, A., Tygert, M. A randomized algorithm for principal component analysis. *SIAM J. Matrix Anal. Appl. 31*, 3 (2009), 1100–1124.
42. Rudelson, M., Vershynin, R. Sampling from large matrices: an approach through geometric functional analysis. *J. ACM 54*, 4 (2007), Article 21.
43. Sarlós, T.. Improved approximation algorithms for large matrices via random projections. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science* (2006), 143–152.
44. Smale, S. Some remarks on the foundations of numerical analysis. *SIAM Rev. 32*, 2 (1990), 211–220.
45. Spielman, D.A., Srivastava, N. Graph sparsification by effective resistances. *SIAM J. Comput. 40*, 6 (2011), 1913–1926.
46. Stigler, S.M. *The History of Statistics: The Measurement of Uncertainty before 1900*. Harvard University Press, Cambridge, 1986.
47. Tropp, J.A. User-friendly tail bounds for sums of random matrices. *Found. Comput. Math. 12*, 4 (2012), 389–434.
48. Turing, A.M. Rounding-off errors in matrix processes. *Quart. J. Mech. Appl. Math. 1* (1948), 287–308.
49. von Neumann, J., Goldstine, H.H. Numerical inverting of matrices of high order. *Bull. Am. Math. Soc. 53* (1947), 1021–1099.
50. Wigner, E.P. Random matrices in physics. *SIAM Rev. 9*, 1 (1967), 1–23.
51. Woodruff, D.P. *Sketching as a Tool for Numerical Linear Algebra*. Foundations and Trends in Theoretical Computer Science. NOW Publishers, Boston, 2014.
52. Yang, J., Meng, X., Mahoney, M.W. Implementing randomized matrix algorithms in parallel and distributed environments. *Proc. IEEE 104*, 1 (2016), 58–92.
53. Yang, J., Rübel, O., Prabhat, Mahoney, M.W., Bowen, B.P. Identifying important ions and positions in mass spectrometry imaging data using CUR matrix decompositions. *Anal. Chem. 87*, 9 (2015), 4658–4666.
54. Yip, C.-W., Mahoney, M.W., Szalay, A.S., Csabai, I., Budavari, T., Wyse, R.F.G., Dobos, L. Objective identification of informative wavelength regions in galaxy spectra. *Astron. J. 147*, 110 (2014), 15.

Petros Drineas (drineas@gmail.com) is an associate professor in the Department of Computer Science at Rensselaer Polytechnic Institute, Troy, NY.

Michael W. Mahoney (mmahoney@stat.berkeley.edu) is an associate professor in ICSI and in the Department of Statistics at the University of California at Berkeley.