SIAM J. SCI. COMPUT.
Vol. 41, No. 4, pp. A2430–A2463

STREAMING LOW-RANK MATRIX APPROXIMATION WITH AN APPLICATION TO SCIENTIFIC SIMULATION*

JOEL A. TROPP†, ALP YURTSEVER‡, MADELEINE UDELL§, AND VOLKAN CEVHER‡

Abstract. This paper argues that randomized linear sketching is a natural tool for on-the-fly compression of data matrices that arise from large-scale scientific simulations and data collection. The technical contribution consists in a new algorithm for constructing an accurate low-rank approximation of a matrix from streaming data. This method is accompanied by an a priori analysis that allows the user to set algorithm parameters with confidence and an a posteriori error estimator that allows the user to validate the quality of the reconstructed matrix. In comparison to previous techniques, the new method achieves smaller relative approximation errors and is less sensitive to parameter choices. As concrete applications, the paper outlines how the algorithm can be used to compress a Navier–Stokes simulation and a sea surface temperature dataset.

Key words. dimension reduction, matrix approximation, numerical linear algebra, sketching, streaming, singular value decomposition

AMS subject classifications. Primary, 65F30; Secondary, 68W20

DOI. 10.1137/18M1201068

1. Motivation.

Computer simulations of scientific models often generate data matrices that are too large to store, process, or transmit in full. This challenge arises in a huge number of fields, including weather and climate forecasting [72, 25, 8], heat transfer and fluid flow [57, 10], computational fluid dynamics [9, 28], and aircraft design [51, 62]. Similar exigencies can arise with automated methods for acquiring large volumes of scientific data [19].

In these settings, the data matrix often has a decaying singular value spectrum, so it admits an accurate low-rank approximation. For some downstream applications, the approximation serves as well as—or even better than—the full matrix [64, 17]. Indeed, the approximation is easier to manipulate, and it can expose latent structure. This observation raises the question of how best to compute a low-rank approximation of a matrix of scientific data with limited storage, arithmetic, and communication.

The main purpose of this paper is to argue that sketching methods from the field of randomized linear algebra [74, 20, 35, 46, 73, 14, 29, 69, 68] have tremendous potential in this context. As we will explain, these algorithms can inexpensively maintain a summary, or *sketch*, of the data as it is being generated. After the data collection process terminates, we can extract a near-optimal low-rank approximation

*Submitted to the journal's Methods and Algorithms for Scientific Computing section July 17, 2018; accepted for publication (in revised form) May 1, 2019; published electronically July 30, 2019. https://doi.org/10.1137/18M1201068

Funding: The work of the first author was partially supported by ONR awards N00014-11-1002, N00014-17-1-214, N00014-17-1-2146, and the Gordon & Betty Moore Foundation. The work of the third author was partially supported by DARPA award FA8750-17-2-0101. The work of the fourth author was supported by the European Research Council (ERC) under the European Union's Horizon 2020 Research and Innovation Programme under grant agreement 725594 (time-data) and the Swiss National Science Foundation (SNSF) under grant 200021_178865.

†Computing + Mathematical Sciences, California Institute of Technology, Pasadena, CA 91125-5000 (jtropp@cms.caltech.edu).

‡Ecole Polytechnique Federal de Lausanne, 1015 Lausanne, Switzerland (alp.yurtsever@epfl.ch, volkan.cevher@epfl.ch).

§Operations Research and Information Engineering, Cornell University, Ithaca, NY 14850 (udell@cornell.edu).

from the sketch. This approximation is accompanied by an a posteriori error estimate.

The second purpose of this paper is to design, analyze, and test a new sketching algorithm that is suitable for handling scientific data. We will build out the theoretical infrastructure needed for practitioners to deploy this algorithm with confidence. We will also demonstrate that the method is effective for some small- and medium-scale examples, including a computer simulation of the Navier–Stokes equations and a high-resolution sea surface temperature dataset [1].

1.1. Streaming, sketching, and matrix approximation. Let us begin with a brief introduction to streaming data and sketching, as they apply to the problem of low-rank matrix approximation. This abstract presentation will solidify into a concrete algorithm in sections 2 and 6. The explanation borrows heavily from our previous paper [69], which contains more details and context.

1.1.1. Streaming. We are interested in acquiring a compressed representation of an enormous matrix $\boldsymbol{A} \in \mathbb{F}^{m \times n}$ where $\mathbb{F} = \mathbb{R}$ or $\mathbb{F} = \mathbb{C}$. This work focuses on a setting where the matrix is presented as a long sequence of "simple" linear updates:

$$\boldsymbol{A} = \boldsymbol{H}_1 + \boldsymbol{H}_2 + \boldsymbol{H}_3 + \cdots . \tag{1.1}$$

In applications, each innovation $\boldsymbol{H}_i$ is sparse, is low-rank, or enjoys another favorable structure. The challenge arises because we do not wish to store the full matrix $\boldsymbol{A}$, and we cannot revisit the innovation $\boldsymbol{H}_i$ after processing it. The formula (1.1) describes a particular type of *streaming data model* [53, 20, 73].

1.1.2. Sketching. To manage the data stream (1.1), we can use a *randomized linear sketch* [5, 4]. Before any data arrives, we draw and fix a random linear map $\mathscr{S} : \mathbb{F}^{m \times n} \to \mathbb{F}^d$, called a *sketching operator*. Instead of keeping $\boldsymbol{A}$ in working memory, we retain only the image $\mathscr{S}(\boldsymbol{A})$. This image is called a *sketch* of the matrix. The dimension d of the sketch is much smaller than the dimension mn of the matrix space, so the sketching operator compresses the data matrix. Nonetheless, because of the randomness, a well-designed sketching operator is likely to yield useful information about any matrix $\boldsymbol{A}$ that is statistically independent from $\mathscr{S}$.

Sketches and data streams enjoy a natural synergy. If the matrix $\boldsymbol{A}$ is presented via the data stream (1.1), the linearity of the sketching operator ensures that

$$\mathscr{S}(\boldsymbol{A}) = \mathscr{S}(\boldsymbol{H}_1) + \mathscr{S}(\boldsymbol{H}_2) + \mathscr{S}(\boldsymbol{H}_3) + \cdots .$$

In other words, we can process an innovation $\boldsymbol{H}_i$ by forming $\mathscr{S}(\boldsymbol{H}_i)$ and adding it to the current value of the sketch. This update can be performed efficiently when $\boldsymbol{H}_i$ is structured. It is a striking fact [43] that randomized linear sketches are essentially the only mechanism for tracking a general data stream of the form (1.1).

1.1.3. Matrix approximation. After the updating process terminates, we need to extract a low-rank approximation of the data matrix $\boldsymbol{A}$ from the sketch $\mathscr{S}(\boldsymbol{A})$. More precisely, we report a rank-r matrix $\hat{\boldsymbol{A}}_r$, in factored form, that satisfies

$$\|\boldsymbol{A} - \hat{\boldsymbol{A}}_r\|_2 \approx \min_{\operatorname{rank} \boldsymbol{B} \le r} \|\boldsymbol{A} - \boldsymbol{B}\|_2, \tag{1.2}$$

where $\|\cdot\|_2$ is the Schatten 2-norm (also known as the Frobenius norm). We can also exploit the sketch to compute an a posteriori estimate of the error:

$$\operatorname{err}_2(\hat{\boldsymbol{A}}_r) \approx \|\boldsymbol{A} - \hat{\boldsymbol{A}}_r\|_2.$$

This estimator helps us to select the precise rank r of the approximation.

1.1.4. Goals. Our objective is to design a sketch rich enough to support these operations. Since a rank-r matrix has about $2r(m+n)$ degrees of freedom, the sketch ideally should have size $d = \Theta(r(m+n))$. We want to compute the approximation using $\mathcal{O}(r^2(m+n))$ floating-point operations, the cost of orthogonalizing r vectors.

Existing one-pass SVD algorithms (e.g., [74, 20, 35, 73, 15, 71, 69]) already meet these desiderata. Nevertheless, there remains room for improvement [69, sect. 7.6].

1.1.5. Contributions. The main technical contribution of this paper is a new sketch-based algorithm for computing a low-rank approximation of a matrix from streaming data. The new algorithm is a hybrid of the methods from [71, Thm. 12] and [69, Alg. 7] that improves on the performance of its predecessors. Here are the key features of our work:

- The new method can achieve a near-optimal relative approximation (1.2) when the input matrix has a decaying singular value spectrum. In particular, our approach is more accurate than existing methods, especially when the storage budget is small. As a consequence, the new method delivers higher-quality estimates of leading singular vectors when the associated singular values are sufficiently separated (section 7).
- The algorithm is accompanied by a priori error bounds that help us set the parameters of the sketch reliably. The new method is less sensitive to the choice of sketch parameters and to the truncation rank, as compared with existing methods (sections 5 and 7).
- Our toolkit includes an a posteriori error estimator for validating the quality of the approximation. This estimator also provides a principled mechanism for selecting the precise rank of the final approximation (section 6).
- The method treats the two matrix dimensions symmetrically. As a consequence, we can extend it to obtain an algorithm for low-rank Tucker approximation of a tensor from streaming data. See our follow-up paper [65].

For scientific simulation and data analysis, these advances are significant because they allow us to approximate the truncated singular value decomposition (SVD) of a huge matrix accurately and with minimal resource usage.

Remark 1.1 (multiple passes). The algorithms in this paper may not be effective for matrices that lack spectral decay. If it is possible to make several passes over the data, we strongly recommend the randomized SVD algorithms from [35, 34].

1.2. Application to scientific simulation. As we have mentioned, it is often desirable to reduce scientific data before we submit it for further processing. This section outlines some of the techniques that are commonly used for this purpose, and it argues that randomized linear sketching may offer a better solution.

1.2.1. Dynamical model for a simulation. In many cases, we can model a simulation as a process that computes the state $\boldsymbol{a}_{t+1} \in \mathbb{F}^m$ of a system at time $t+1$ from the state $\boldsymbol{a}_t \in \mathbb{F}^m$ of the system at time t. We may collect the data generated by the simulation into a matrix $\boldsymbol{A} = [\boldsymbol{a}_1, \dots, \boldsymbol{a}_n] \in \mathbb{F}^{m \times n}$. In scientific applications, it is common that this matrix has a decaying singular value spectrum.

The dimension m of the state typically increases with the resolution of the simulation, and it can be very big. The time horizon n can also be large, especially for problems involving multiple time scales and for "stiff" equations that have high sen-

sitivity to numerical errors. In some settings, we may not even know the time horizon n or the dimension m of the state variable in advance.

1.2.2. On-the-fly compression via sketching. Let us explain how sketching interacts with the dynamical model from subsection 1.2.1. For simplicity, assume that the dimensions m and n of the data matrix $\boldsymbol{A} \in \mathbb{F}^{m \times n}$ are known. Draw and fix a randomized linear sketching operator $\mathscr{S} : \mathbb{F}^{m \times n} \to \mathbb{F}^d$.

We can view the dynamical model for the simulation as an instance of the data stream (1.1):

$$\boldsymbol{A} = \boldsymbol{a}_1 \mathbf{e}_1^* + \boldsymbol{a}_2 \mathbf{e}_2^* + \boldsymbol{a}_3 \mathbf{e}_3^* + \cdots .$$

Here, $\mathbf{e}_i$ is the ith standard basis vector in $\mathbb{F}^n$. The sketch $\boldsymbol{x} = \mathscr{S}(\boldsymbol{A}) \in \mathbb{F}^d$ evolves as

$$\boldsymbol{x} = \mathscr{S}(\boldsymbol{a}_1 \mathbf{e}_1^*) + \mathscr{S}(\boldsymbol{a}_2 \mathbf{e}_2^*) + \mathscr{S}(\boldsymbol{a}_3 \mathbf{e}_3^*) + \cdots .$$

Each time the simulation generates a new state $\boldsymbol{a}_t$, we update the sketch $\boldsymbol{x}$ to reflect the innovation $\boldsymbol{a}_t \mathbf{e}_t^*$ to the data matrix $\boldsymbol{A}$. We can exploit the fact that the innovation is a rank-one matrix to ensure that this computation has negligible incremental cost. After sketching the new state, we write it to external memory or simply discard it. Once the simulation is complete, we can extract a provably good low-rank approximation from the sketch, along with an error estimate.

1.2.3. Compression of scientific data: Current art. At present, computational scientists rely on several other strategies for data reduction. One standard practice is to collect the full data matrix and then to compress it. Methods include direct computation of a low-rank matrix or tensor approximation [76, 6] or fitting a statistical model [18, 33, 47]. These approaches have high storage costs, and they entail communication of large volumes of data.

There are also some techniques for compressing simulation output as it is generated. One approach is to store only a subset of the columns of the data matrix ("snapshots" or "checkpointing") instead of keeping the full trajectory [32, 37]. Another approach is to maintain a truncated SVD using a rank-one updating method [16, 77]. Both techniques have the disadvantage that they do not preserve a complete, consistent view of the data matrix. The rank-one updating method also incurs a substantial computational cost at each step.

1.2.4. Contributions. We believe that randomized linear sketching resolves many of the shortcomings of earlier data reduction methods for scientific applications. We will show by example (section 7) that our new sketching algorithm can be used to compress scientific data drawn from several applications:

- We apply the method to a 430 megabyte (MB) data matrix from a direct numerical simulation, via the Navier–Stokes equations, of vortex shedding from a cylinder in a two-dimensional channel flow.
- The method is used to approximate a 1.1 gigabyte (GB) temperature dataset collected at a network of weather stations in the northeastern United States.
- We can invoke the sketching algorithm as a module in an optimization algorithm for solving a large-scale phase retrieval problem that arises in microscopic imaging via Fourier ptychography. The full matrix would require over 5 GB of storage.
- As a larger-scale example, we show that our methodology allows us to compute an accurate truncated SVD of a sea surface temperature dataset, which

requires over 75 GB in double precision. This experiment is performed without any adjustment of parameters or other retrospection.

These demonstrations support our assertion that sketching is a powerful tool for managing large-scale data from scientific simulations and measurement processes. We have written this paper to motivate computational scientists to consider sketching in their own applications.

1.3. Roadmap. In section 2, we give a detailed presentation of the proposed method and its relationship to earlier work. We provide an informative mathematical analysis that explains the behavior of our algorithm (section 5), and we describe how to construct a posteriori error estimates (section 6). We also discuss implementation issues (section 4), and we present extensive numerical experiments on real and simulated data (section 7).

1.4. Notation. We use $\mathbb{F}$ for the scalar field, which is real $\mathbb{R}$ or complex $\mathbb{C}$. The symbol * refers to the (conjugate) transpose of a matrix or vector. The dagger † denotes the Moore–Penrose pseudoinverse. We write $\|\cdot\|_p$ for the Schatten p-norm for $p \in [1, \infty]$. The map $[\![\cdot]\!]_r$ returns any (simultaneous) best rank-r approximation of its argument with respect to the Schatten p-norms [36, sect. 6].

2. Sketching and low-rank approximation of a matrix. Let us describe the basic procedure for sketching a matrix and for computing a low-rank approximation from the sketch. We discuss prior work in subsection 2.8. See section 4 for implementation, section 5 for parameter selection, and section 6 for error estimation.

2.1. Dimension reduction maps. We will use dimension reduction to collect information about an input matrix. Assume that $d \leq N$. A *randomized linear dimension reduction map* is a random matrix $\boldsymbol{\Xi} \in \mathbb{F}^{d \times N}$ with the property that

$$\mathbb{E}\,\|\boldsymbol{\Xi}\boldsymbol{u}\|_2^2 = \text{const} \cdot \|\boldsymbol{u}\|_2^2 \quad \text{for all } \boldsymbol{u} \in \mathbb{F}^N. \tag{2.1}$$

In other words, the map reduces a vector of dimension N to dimension d, but it still preserves Euclidean distances on average. It is also desirable that we can store the map $\boldsymbol{\Xi}$ and apply it to vectors efficiently. See section 3 for concrete examples.

2.2. The input matrix. Let $\boldsymbol{A} \in \mathbb{F}^{m \times n}$ be an arbitrary matrix that we wish to approximate. In many applications where sketching is appropriate, the matrix is presented implicitly as a sequence of linear updates; see subsection 2.4.

To apply sketching methods for low-rank matrix approximation, the user needs to specify a target rank r_0. The target rank r_0 is a rough estimate for the final rank of the approximation, and it influences the choice of the sketch size. We can exploit a posteriori information to select the final rank; see subsection 6.5.

Remark 2.1 (unknown dimensions). For simplicity, we assume the matrix dimensions are known in advance. The framework can be modified to handle matrices with growing dimensions, such as a simulation with an unspecified time horizon.

2.3. The sketch. Let us describe the sketching operators we use to acquire data about the input matrix. The sketching operators are parameterized by a "range" parameter k and a "core" parameter s that satisfy

$$r_0 \leq k \leq s \leq \min\{m, n\},$$

where r_0 is the target rank. The parameter k determines the maximum rank of an approximation. For now, be aware that the approximation scheme is more sensitive to

the choice of k than to the choice of s. In subsection 5.4, we offer specific parameter recommendations that are supported by theoretical analysis. In subsection 7.5, we demonstrate that these parameter choices are effective in practice.

Independently, draw and fix four randomized linear dimension reduction maps:

$$\begin{aligned} \boldsymbol{\Upsilon} &\in \mathbb{F}^{k\times m} \quad \text{and} \quad \boldsymbol{\Omega} \in \mathbb{F}^{k\times n}, \\ \boldsymbol{\Phi} &\in \mathbb{F}^{s\times m} \quad \text{and} \quad \boldsymbol{\Psi} \in \mathbb{F}^{s\times n}. \end{aligned} \tag{2.2}$$

These dimension reduction maps are often called *test matrices*. The sketch itself consists of three matrices:

$$\boldsymbol{X} := \boldsymbol{\Upsilon}\boldsymbol{A} \in \mathbb{F}^{k\times n} \quad \text{and} \quad \boldsymbol{Y} := \boldsymbol{A}\boldsymbol{\Omega}^* \in \mathbb{F}^{m\times k}, \tag{2.3}$$

$$\boldsymbol{Z} := \boldsymbol{\Phi}\boldsymbol{A}\boldsymbol{\Psi}^* \in \mathbb{F}^{s\times s}. \tag{2.4}$$

The first two matrices $(\boldsymbol{X}, \boldsymbol{Y})$ capture the corange and the range of $\boldsymbol{A}$. The core sketch $(\boldsymbol{Z})$ contains fresh information that improves our estimates of the singular values and singular vectors of $\boldsymbol{A}$; it is responsible for the superior performance of the new method.

Remark 2.2 (prior work). Upadhyay's paper [71, sect. 3] contains the insight that a sketch of the form (2.3) and (2.4) can support better low-rank matrix approximations, but it proposes a reconstruction algorithm that is less effective. Related (but distinct) sketches appear in the papers [74, 20, 35, 73, 23, 15, 70, 69].

2.4. Linear updates. In streaming data applications, the input matrix $\boldsymbol{A} \in \mathbb{F}^{m\times n}$ is presented as a sequence of linear updates of the form

$$\boldsymbol{A} \leftarrow \eta\boldsymbol{A} + \nu\boldsymbol{H}, \tag{2.5}$$

where $\eta, \nu \in \mathbb{F}$ and the matrix $\boldsymbol{H} \in \mathbb{F}^{m\times n}$.

In view of the construction (2.3) and (2.4), we can update the sketch $(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z})$ of the matrix $\boldsymbol{A}$ to reflect the innovation (2.5) by means of the formulae

$$\begin{aligned} \boldsymbol{X} &\leftarrow \eta\boldsymbol{X} + \nu\boldsymbol{\Upsilon}\boldsymbol{H}, \\ \boldsymbol{Y} &\leftarrow \eta\boldsymbol{Y} + \nu\boldsymbol{H}\boldsymbol{\Omega}^*, \\ \boldsymbol{Z} &\leftarrow \eta\boldsymbol{Z} + \nu\boldsymbol{\Phi}\boldsymbol{H}\boldsymbol{\Psi}^*. \end{aligned} \tag{2.6}$$

When implementing these updates, it is worthwhile to exploit favorable structure in the matrix $\boldsymbol{H}$, such as sparsity or low rank.

Remark 2.3 (streaming model). For the linear update model (2.5), randomized linear sketches are more or less the only way to track the input matrix [43]. There are more restrictive streaming models (e.g., when the columns of the matrix are presented in sequence) where it is possible to design other types of algorithms [26, 29].

Remark 2.4 (linearly transformed data). We can use sketching to track any matrix that depends linearly on a data stream. Suppose that the input data $\boldsymbol{a} \in \mathbb{R}^d$, and we want to maintain the matrix $\mathscr{L}(\boldsymbol{a})$ induced by a fixed linear map $\mathscr{L} : \mathbb{F}^d \to \mathbb{F}^{m\times n}$. If we receive an update $\boldsymbol{a} \leftarrow \eta\boldsymbol{a} + \nu\boldsymbol{h}$, then the linear image evolves as $\mathscr{L}(\boldsymbol{a}) \leftarrow \eta\mathscr{L}(\boldsymbol{a}) + \nu\mathscr{L}(\boldsymbol{h})$. This update has the form (2.5), so we can apply the matrix sketch (2.3) and (2.4) to track $\mathscr{L}(\boldsymbol{a})$ directly. This idea has applications to physical simulations where a known transform $\mathscr{L}$ exposes structure in the data [52].

2.5. Optional step: Centering. Many applications require us to center the data matrix to remove a trend, such as the temporal average. Principal component analysis (PCA) also involves a centering step [42]. For superior accuracy, it is wise to perform this operation *before* sketching the matrix.

As an example, let us explain how to compute and remove the mean value of each row of the data matrix in the streaming setting. We can maintain an extra vector $\boldsymbol{\mu} \in \mathbb{F}^m$ that tracks the mean value of each row. To process an update of the form (2.5), we first apply the steps

$$\boldsymbol{h} \leftarrow n^{-1}\boldsymbol{H}\mathbf{e} \quad \text{and} \quad \boldsymbol{H} \leftarrow \boldsymbol{H} - \boldsymbol{h}\mathbf{e}^* \quad \text{and} \quad \boldsymbol{\mu} \leftarrow \eta\boldsymbol{\mu} + \nu\boldsymbol{h}.$$

Here, $\mathbf{e} \in \mathbb{F}^n$ is the vector of ones. Afterward, we update the sketches using (2.6). The sketch now contains the centered data matrix, where each row has zero mean.

2.6. Computing truncated low-rank approximations. Once we have acquired a sketch $(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z})$ of the input matrix $\boldsymbol{A}$, we must produce a good low-rank approximation. Let us outline the computations we propose. The intuition appears below in subsection 2.7, and section 5 presents a theoretical analysis.

The first two components $(\boldsymbol{X}, \boldsymbol{Y})$ of the sketch are used to estimate the corange and the range of the matrix $\boldsymbol{A}$. Compute thin QR factorizations:

$$\begin{aligned} \boldsymbol{X}^* &=: \boldsymbol{P}\boldsymbol{R}_1, \quad \text{where} \quad \boldsymbol{P} \in \mathbb{F}^{n\times k}, \\ \boldsymbol{Y} &=: \boldsymbol{Q}\boldsymbol{R}_2, \quad \text{where} \quad \boldsymbol{Q} \in \mathbb{F}^{m\times k}. \end{aligned} \tag{2.7}$$

Both $\boldsymbol{P}$ and $\boldsymbol{Q}$ have orthonormal columns; discard the triangular parts $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$.

The third sketch $\boldsymbol{Z}$ is used to compute the core approximation $\boldsymbol{C}$, which describes how $\boldsymbol{A}$ acts between range$(\boldsymbol{P})$ and range$(\boldsymbol{Q})$:

$$\boldsymbol{C} := (\boldsymbol{\Phi}\boldsymbol{Q})^\dagger \boldsymbol{Z}((\boldsymbol{\Psi}\boldsymbol{P})^\dagger)^* \in \mathbb{F}^{k\times k}. \tag{2.8}$$

This step is implemented by solving a family of least-squares problems.

Next, form a rank-k approximation $\hat{\boldsymbol{A}}$ of the input matrix $\boldsymbol{A}$ via

$$\hat{\boldsymbol{A}} := \boldsymbol{Q}\boldsymbol{C}\boldsymbol{P}^*. \tag{2.9}$$

We refer to $\hat{\boldsymbol{A}}$ as the "initial" approximation. It is important to be aware that the initial approximation can contain spurious information (in the singular subspaces associated with its smaller singular values).

To produce an approximation that is fully reliable, we must truncate the rank of the initial approximation (2.9). For a truncation parameter r, we construct a rank-r approximation by replacing $\hat{\boldsymbol{A}}$ with its best rank-r approximation[1] in Frobenius norm:

$$[\![\hat{\boldsymbol{A}}]\!]_r = \boldsymbol{Q}[\![\boldsymbol{C}]\!]_r\boldsymbol{P}^*. \tag{2.10}$$

We refer to $[\![\hat{\boldsymbol{A}}]\!]_r$ as a "truncated" approximation. Subsection 6.5 outlines some ways to use a posteriori information to select the truncation rank r.

The truncation (2.10) has an appealing permanence property: $[\![[\![\hat{\boldsymbol{A}}]\!]_\varrho]\!]_r = [\![\hat{\boldsymbol{A}}]\!]_r$ for all $\varrho \geq r$. In other words, the rank-r approximation persists as part of all higher-rank approximations. In contrast, some earlier reconstruction methods are unstable in the sense that the rank-r approximation varies wildly with r; see subsection SM4.7.

[1]The formula (2.10) is an easy consequence of the Eckart–Young theorem [36, sect. 6] and the fact that $\boldsymbol{Q}, \boldsymbol{P}$ have orthonormal columns.

Remark 2.5 (extensions). We can form other structured approximations of $\boldsymbol{A}$ by projecting $\hat{\boldsymbol{A}}$ onto a set of structured matrices. See [69, sects. 5–6] for a discussion of this idea in the context of another sketching technique. For brevity, we do not develop this point further. See our paper [68] for a sketching and reconstruction method designed specifically for positive-semidefinite (psd) matrices.

Remark 2.6 (prior work). The truncated approximation (2.10) is new, but it depends on insights from our previous work [70, 69]. Upadhyay [71, Thm. 12] proposes a different reconstruction formula for the same kind of sketch. The papers [74, 20, 35, 73, 23, 15, 71] describe other methods for low-rank matrix approximation from a randomized linear sketch. The numerical work in section 7 demonstrates that (2.10) matches or improves on earlier techniques.

2.7. Intuition. The approximations (2.9) and (2.10) are based on some well-known insights from randomized linear algebra [35, sect. 1]. Since $\boldsymbol{P}$ and $\boldsymbol{Q}$ capture the corange and range of the input matrix, we expect that

$$\boldsymbol{A} \approx \boldsymbol{Q}(\boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{P})\boldsymbol{P}^*. \tag{2.11}$$

(See Lemma SM1.5 for justification.) We cannot compute the core matrix $\boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{P}$ directly from a linear sketch because $\boldsymbol{P}$ and $\boldsymbol{Q}$ are functions of $\boldsymbol{A}$. Instead, we estimate the core matrix using the core sketch $\boldsymbol{Z}$. Owing to the approximation (2.11),

$$\boldsymbol{Z} = \boldsymbol{\Phi}\boldsymbol{A}\boldsymbol{\Psi}^* \approx (\boldsymbol{\Phi}\boldsymbol{Q})(\boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{P})(\boldsymbol{P}^*\boldsymbol{\Psi}^*).$$

Transfer the outer matrices to the left-hand side to discover that the core approximation $\boldsymbol{C}$, defined in (2.8), satisfies

$$\boldsymbol{C} = (\boldsymbol{\Phi}\boldsymbol{Q})^\dagger \boldsymbol{Z}((\boldsymbol{\Psi}\boldsymbol{P})^\dagger)^* \approx \boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{P}. \tag{2.12}$$

In view of (2.11) and (2.12), we arrive at the relations

$$\boldsymbol{A} \approx \boldsymbol{Q}(\boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{P})\boldsymbol{P}^* \approx \boldsymbol{Q}\boldsymbol{C}\boldsymbol{P}^* = \hat{\boldsymbol{A}}.$$

The error in the last relation depends on the error in the best rank-k approximation of $\boldsymbol{A}$. When $\boldsymbol{A}$ has a decaying spectrum, the rank-r truncation of $\boldsymbol{A}$ for $r \ll k$ agrees closely with the rank-r truncation of the initial approximation $\hat{\boldsymbol{A}}$. That is,

$$[\![\boldsymbol{A}]\!]_r \approx [\![\hat{\boldsymbol{A}}]\!]_r = \boldsymbol{Q}[\![\boldsymbol{C}]\!]_r\boldsymbol{P}^*.$$

Theorem 5.1 and Corollary 5.5 justify these heuristics completely for Gaussian dimension reduction maps. Subsection 6.5 discusses a posteriori selection of r.

2.8. Discussion of related work. Sketching algorithms are specifically designed for the streaming model, that is, for data that is presented as a sequence of updates. The sketching paradigm is attributed to [5, 4]; see the survey [53] for an introduction and overview of early work.

Randomized algorithms for low-rank matrix approximation were proposed in the theoretical computer science (TCS) literature in the late 1990s [56, 27]. Soon after, numerical analysts developed practical versions of these algorithms [49, 74, 60, 35, 34]. For more background on the history of randomized linear algebra, see [35, 46, 73].

The paper [74] contains the first one-pass algorithm for low-rank matrix approximation; it was designed to control communication and arithmetic costs, rather than to handle streaming data. The first general treatment of numerical linear algebra in the streaming model appears in [20]. Recent papers on low-rank matrix approximation in the streaming model include [15, 71, 26, 29, 69, 68].

Algorithm 3.1 Dimension Reduction Map Class.

```
class DimRedux (𝔽)                                  ▷ Dimension reduction map over field 𝔽
    function DimRedux(d, N)                         ▷ Construct map Ξ : 𝔽^N → 𝔽^d
    function DimRedux.mtimes(DRmap, M)              ▷ Left action of map
    function DimRedux.mtimes(M, DRmap*)             ▷ Right action of adjoint
        return (DimRedux.mtimes(DRmap, M*))*        ▷ Default behavior
```

2.8.1. Approaches from NLA. The nonlinear algebra (NLA) literature contains a number of papers [74, 35, 69] on low-rank approximation from a randomized linear sketch. These methods all compute the range matrix $\boldsymbol{Q}$ and the corange matrix $\boldsymbol{P}$ using the randomized range finder [35, Alg. 4.1] encapsulated in (2.3) and (2.7).

The methods differ in how they construct a core matrix $\tilde{\boldsymbol{C}}$ so that $\boldsymbol{A} \approx \boldsymbol{Q}\tilde{\boldsymbol{C}}\boldsymbol{P}$. Earlier papers reuse the range and corange sketches $(\boldsymbol{X}, \boldsymbol{Y})$ and the associated test matrices $(\boldsymbol{\Upsilon}, \boldsymbol{\Omega})$ to form $\tilde{\boldsymbol{C}}$. Our new algorithm is based on an insight from [15, 71] that the estimate $\boldsymbol{C}$ from (2.8) is more reliable because it uses a random sketch $\boldsymbol{Z}$ that is statistically independent from $(\boldsymbol{X}, \boldsymbol{Y})$. The storage cost of the additional sketch is negligible when $s^2 \ll k(m+n)$.

The methods also truncate the rank of the approximation at different steps. The older papers [74, 35] perform the truncation *before* estimating the core matrix (cf. subsection SM4.1.1). One insight from [69] is that it is beneficial to perform the truncation *after* estimating the core matrix. Furthermore, an effective truncation mechanism is to report a best rank-r approximation of the initial estimate. We have adopted the latter approach.

2.8.2. Approaches from TCS. Most of the algorithms in the TCS literature [20, 73, 23, 15, 71] are based on a framework called "sketch-and-solve" that is attributed to Sarlós [63]. The basic idea is that the solution to a constrained least-squares problem (e.g., low-rank matrix approximation in Frobenius norm) is roughly preserved when we solve the problem after randomized linear dimension reduction.

The sketch-and-solve framework sometimes leads to the same algorithms as the NLA point of view; other times, it leads to different approaches. It would take us too far afield to detail these derivations, but we give a summary of one such method [71, Thm. 12] in subsection SM4.1.3. Unfortunately, sketch-and-solve algorithms are often unsuitable for high-accuracy computations; see section 7 and [69, sect. 7] for evidence.

A more salient criticism is that the TCS literature does not attend to the issues that arise if we want to use sketching algorithms in practice. We have expended a large amount of effort to address these challenges, which range from parameter selection to numerically sound implementation. See [69, sect. 1.7.4] for more discussion.

3. Randomized linear dimension reduction maps. In this section, we describe several randomized linear dimension reduction maps that are suitable for implementing sketching algorithms for low-rank matrix approximation. See [44, 35, 73, 69, 66] for additional discussion and examples. The class template for a dimension reduction map appears as Algorithm 3.1; the algorithms for specific dimension reduction techniques are postponed to the supplement.

3.1. Gaussian maps. The most basic dimension reduction map is simply a Gaussian matrix. That is, $\boldsymbol{\Xi} \in \mathbb{F}^{d\times N}$ is a $d \times N$ matrix with independent standard

normal entries.[2]

Algorithm SM3.1 describes an implementation of Gaussian dimension reduction. The map $\boldsymbol{\Xi}$ requires storage of dN floating-point numbers in the field $\mathbb{F}$. The cost of applying the map to a vector is $\mathcal{O}(dN)$ arithmetic operations.

Gaussian dimension reduction maps are simple, and they are effective in randomized algorithms for low-rank matrix approximation [35]. We can also analyze their behavior in full detail; see sections 5 and 6. On the other hand, it is expensive to draw a large number of Gaussian random variables, and the cost of storage and arithmetic renders these maps less appealing when the output dimension d is large.

Remark 3.1 (unknown dimension). Since the columns of a Gaussian map $\boldsymbol{\Xi}$ are statistically independent, we can instantiate more columns if we need to apply $\boldsymbol{\Xi}$ to a longer vector. Sparse maps (subsection 3.3) share this feature. This observation is valuable in the streaming setting, where a linear update might involve coordinates heretofore unseen, forcing us to enlarge the domain of the dimension reduction map.

Remark 3.2 (history). Gaussian dimension reduction has been used as an algorithmic tool since the paper [40] of Indyk and Motwani. In spirit, this approach is quite similar to the earlier theoretical work of Johnson and Lindenstrauss [41], which performs dimension reduction by projection onto a random subspace.

3.2. Scrambled SRFT maps. Next, we describe a structured dimension reduction map, called a *scrambled subsampled randomized Fourier transform* (SSRFT). We recommend this approach for practical implementations.

An SSRFT map takes the form[3]

$$\boldsymbol{\Xi} = \boldsymbol{R}\boldsymbol{F}\boldsymbol{\Pi}\boldsymbol{F}\boldsymbol{\Pi}' \in \mathbb{F}^{d\times N}.$$

The matrices $\boldsymbol{\Pi}, \boldsymbol{\Pi}' \in \mathbb{F}^{N\times N}$ are signed permutations,[4] drawn independently and uniformly at random. The matrix $\boldsymbol{F} \in \mathbb{F}^{N\times N}$ denotes a discrete cosine transform ($\mathbb{F} = \mathbb{R}$) or a discrete Fourier transform ($\mathbb{F} = \mathbb{C}$). The matrix $\boldsymbol{R} \in \mathbb{F}^{d\times N}$ is a restriction to d coordinates, chosen uniformly at random.

Algorithm SM3.2 presents an implementation of an SSRFT. The cost of storing $\boldsymbol{\Xi}$ is just $\mathcal{O}(N)$ numbers. The cost of applying $\boldsymbol{\Xi}$ to a vector is $\mathcal{O}(N \log N)$ arithmetic operations, using the fast Fourier transform (FFT) or the fast cosine transform (FCT). According to [74], this cost can be reduced to $\mathcal{O}(N \log d)$, but the improvement is rarely worth the implementation effort.

In practice, SSRFTs behave slightly better than Gaussian matrices, even though their storage cost does not scale with the output dimension d. On the other hand, the analysis [3, 67, 13] is less complete than in the Gaussian case [35]. A proper implementation requires fast trigonometric transforms. Finally, the random permutations and FFTs require data movement, which could be a challenge in the distributed setting.

Remark 3.3 (history). SSRFTs are inspired by the work of Ailon and Chazelle [3] on fast Johnson–Lindstrauss transforms. For applications in randomized linear algebra, see the papers [74, 44, 35, 67, 13].

[2] A real standard normal variable follows the Gaussian distribution with mean zero and variance one. A complex standard normal variable takes the form $g_1 + \mathrm{i}g_2$, where g_i are independent real standard normal variables.

[3] Empirical work suggests that it is not necessary to iterate the permutation and trigonometric transform twice, but this duplication can increase reliability.

[4] A signed permutation matrix has precisely one nonzero entry in each row and column, and each nonzero entry of the matrix has modulus one.

3.3. Sparse sign matrices. Finally, we describe another type of randomized dimension reduction map, called a *sparse sign matrix*. We recommend these maps for practical implementations where data movement is a concern.

To construct a sparse sign matrix $\boldsymbol{\Xi} \in \mathbb{F}^{d\times N}$, we fix a sparsity parameter ζ in the range $2 \leq \zeta \leq d$. The columns of the matrix are drawn independently at random. To construct each column, we take ζ independent and identically distributed (i.i.d.) draws from the UNIFORM$\{z \in \mathbb{F} : |z| = 1\}$ distribution, and we place these random variables in p coordinates, chosen uniformly at random. Empirically, we have found that $\zeta = \min\{d, 8\}$ is a very reliable parameter selection in the context of low-rank matrix approximation.[5]

Algorithm SM3.3 describes an implementation of sparse dimension reduction. Since the matrix $\boldsymbol{\Xi} \in \mathbb{F}^{d\times N}$ has ζ nonzeros per column, we can store the matrix with $\mathcal{O}(\zeta N \log(1 + d/\zeta))$ numbers via run-length coding. The cost of applying the map to a vector is $\mathcal{O}(\zeta N)$ arithmetic operations.

Sparse sign matrices can reduce data movement because the columns are generated independently and the matrices can be applied using (blocked) matrix multiplication. They can also adapt to input vectors whose maximum dimension N may be unknown, as discussed in Remark 3.1. One weakness is that we must use sparse data structures and arithmetic to enjoy the benefit of these maps.

Remark 3.4 (history). Sparse dimension reduction maps are inspired by the work of Achlioptas [2] on database-friendly random projections. For applications in randomized linear algebra, see [21, 50, 54, 55, 12]. See [22] for a theoretical analysis in the context of matrix approximation.

4. Implementation and costs. This section contains further details about the implementation of the sketching and reconstruction methods from section 2, including an account of storage and arithmetic costs. We combine the mathematical notation from the text with MATLAB R2018b commands. The supplemental materials include a MATLAB implementation of these methods.

4.1. Sketching and updates. Algorithms 4.1 and 4.2 contain the pseudocode for initializing the sketch and for performing the linear update (2.5). They also include optional code for maintaining an error sketch (section 6).

The sketch requires storage of four dimension reduction maps with size $k \times m$, $k \times n$, $s \times m$, $s \times n$. We recommend using SSRFTs or sparse sign matrices to minimize the storage costs associated with the dimension reduction maps.

The sketch itself consists of three matrices with dimensions $k \times n$, $m \times k$, and $s \times s$. In general, the sketch matrices are dense, so they require $k(m+n) + s^2$ floating-point numbers in the field $\mathbb{F}$.

The arithmetic cost of the linear update $\boldsymbol{A} \leftarrow \eta\boldsymbol{A} + \tau\boldsymbol{H}$ is dominated by the cost of computing $\boldsymbol{\Phi H}$ and $\boldsymbol{H\Psi}$. In practice, the innovation $\boldsymbol{H}$ is low-rank, sparse, or structured. The precise cost of the update depends on how we exploit the structure of $\boldsymbol{H}$ and the dimension reduction map.

4.2. The initial approximation. Algorithm 4.3 lists the pseudocode for computing a rank-k approximation $\hat{\boldsymbol{A}}$ of the matrix $\boldsymbol{A}$ contained in the sketch; see (2.9).

The method requires additional storage of $k(m + n)$ numbers for the orthonormal matrices $\boldsymbol{P}$ and $\boldsymbol{Q}$, as well as $\mathcal{O}(sk)$ numbers to form the core matrix $\boldsymbol{C}$. The

[5] Empirical testing supports more aggressive choices, say, $\zeta = 4$ or even $\zeta = 2$ for very large problems. On the other hand, the extreme $\zeta = 1$ is disastrous, so we have excluded it.

Algorithm 4.1 Sketch Constructor. Implements (2.2)–(2.4) and (6.2).

Input: Field $\mathbb{F}$; input matrix dimensions $m \times n$; approximation sketch size parameters $k \leq s \leq \min\{m, n\}$; error sketch size parameter q

Output: Draw test matrices for the approximation (2.2) and the error estimate (6.1); form the sketch (2.3), (2.4), and (6.2) of the zero matrix $\boldsymbol{A} = \mathbf{0}$

class SKETCH
  local variables $\boldsymbol{\Upsilon}, \boldsymbol{\Omega}, \boldsymbol{\Phi}, \boldsymbol{\Psi}$ (DIMREDUX)
  local variables $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ (matrices)
  local variables $\boldsymbol{\Theta}$ (GAUSSDR), $\boldsymbol{W}$ (matrix) ▷ [opt] For error estimation
  function SKETCH$(m, n, k, s, q;$ DR) ▷ Constructor; DR is a DIMREDUX
    $\boldsymbol{\Upsilon} \leftarrow \mathrm{DR}(k, m)$ ▷ Construct test matrix for range
    $\boldsymbol{\Omega} \leftarrow \mathrm{DR}(k, n)$ ▷ Test matrix for corange
    $\boldsymbol{\Phi} \leftarrow \mathrm{DR}(s, m)$ ▷ Test matrices for core
    $\boldsymbol{\Psi} \leftarrow \mathrm{DR}(s, n)$
    $\boldsymbol{\Theta} \leftarrow \mathrm{GAUSSDR}(q, m)$ ▷ [opt] Gaussian test matrix for error
    $\boldsymbol{X} \leftarrow \texttt{zeros}(k, n)$ ▷ Approximation sketch of zero matrix
    $\boldsymbol{Y} \leftarrow \texttt{zeros}(m, k)$
    $\boldsymbol{Z} \leftarrow \texttt{zeros}(s, s)$
    $\boldsymbol{W} \leftarrow \texttt{zeros}(q, n)$ ▷ [opt] Error sketch of zero matrix

Algorithm 4.2 Linear Update to Sketch. Implements (2.5) and (6.3).

Input: Innovation $\boldsymbol{H} \in \mathbb{F}^{m \times n}$; scalars $\eta, \nu \in \mathbb{F}$

Output: Modifies sketch to reflect linear update $\boldsymbol{A} \leftarrow \eta \boldsymbol{A} + \nu \boldsymbol{H}$

function SKETCH.LINEARUPDATE$(\boldsymbol{H}; \eta, \nu)$
  $\boldsymbol{X} \leftarrow \eta \boldsymbol{X} + \nu \boldsymbol{\Upsilon} \boldsymbol{H}$ ▷ Update range sketch
  $\boldsymbol{Y} \leftarrow \eta \boldsymbol{Y} + \nu \boldsymbol{H} \boldsymbol{\Omega}^*$ ▷ Update corange sketch
  $\boldsymbol{Z} \leftarrow \eta \boldsymbol{Z} + \nu (\boldsymbol{\Phi} \boldsymbol{H}) \boldsymbol{\Psi}^*$ ▷ Update core sketch
  $\boldsymbol{W} \leftarrow \eta \boldsymbol{W} + \nu \boldsymbol{\Theta} \boldsymbol{H}$ ▷ [opt] Update error sketch

arithmetic cost is usually dominated by the computation of the QR factorizations of $\boldsymbol{X}^*$ and $\boldsymbol{Y}$, which require $\mathcal{O}(k^2(m+n))$ operations, plus communication. When the parameters satisfy $s \gg k$, it is possible that the cost $\mathcal{O}(ks(m+n))$ of forming the core matrix $\boldsymbol{C}$ will dominate; bear this in mind when setting the parameter s.

4.3. The truncated approximation. Algorithm 4.4 presents the pseudocode for computing a rank-r approximation $[\![\hat{\boldsymbol{A}}]\!]_r$ of the matrix $\boldsymbol{A}$ contained in the sketch; see (2.10). The parameter r is an input; the output is presented as a truncated SVD.

The working storage cost $\mathcal{O}(k(m+n))$ is dominated by the call to Algorithm 4.3. Typically, the arithmetic cost is also dominated by the $\mathcal{O}(ks(m+n))$ cost of the call to Algorithm 4.3. When $s \gg k$, we need to invoke a randomized SVD algorithm [35, 34] to achieve this arithmetic cost, but an ordinary dense SVD sometimes serves.

5. A priori error bounds. It is always important to characterize the behavior of numerical algorithms, but the challenge is more acute for sketching methods. Indeed, we cannot store the stream of updates, so we cannot repeat the computation with new parameters if it is unsuccessful. As a consequence, we must perform a priori

Algorithm 4.3 Initial Approximation. Implements (2.9).

Output: Rank-k approximation of sketched matrix in the form $\hat{\boldsymbol{A}} = \boldsymbol{Q}\boldsymbol{C}\boldsymbol{P}^*$ with orthonormal $\boldsymbol{Q} \in \mathbb{F}^{m\times k}$ and $\boldsymbol{P} \in \mathbb{F}^{n\times k}$ and $\boldsymbol{C} \in \mathbb{F}^{k\times k}$

function SKETCH.INITIALAPPROX( )
  $(\boldsymbol{Q}, \sim) \leftarrow \mathtt{qr}(\boldsymbol{Y}, \mathtt{0})$ ▷ Compute orthogonal part of thin QR
  $(\boldsymbol{P}, \sim) \leftarrow \mathtt{qr}(\boldsymbol{X}^*, \mathtt{0})$
  $\boldsymbol{C} \leftarrow ((\boldsymbol{\Phi}\boldsymbol{Q})\backslash\boldsymbol{Z})/((\boldsymbol{\Psi}\boldsymbol{P})^*)$ ▷ Solve two least-squares problems
  return $(\boldsymbol{Q}, \boldsymbol{C}, \boldsymbol{P})$

Algorithm 4.4 Truncated Approximation. Implements (2.10).

Input: Final rank r of the approximation
Output: Rank-r approximation of sketched matrix in the form $\hat{\boldsymbol{A}}_r = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$ with orthonormal $\boldsymbol{U} \in \mathbb{F}^{m\times r}$ and $\boldsymbol{V} \in \mathbb{F}^{n\times r}$ and nonnegative diagonal $\boldsymbol{\Sigma} \in \mathbb{R}^{r\times r}$

function SKETCH.TRUNCATEAPPROX(r)
  $(\boldsymbol{Q}, \boldsymbol{C}, \boldsymbol{P}) \leftarrow$ SKETCH.INITIALAPPROX( )
  $(\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}) \leftarrow \mathtt{svd}(\boldsymbol{C})$ ▷ Dense or randomized SVD
  $\boldsymbol{\Sigma} \leftarrow \boldsymbol{\Sigma}(\mathtt{1{:}r}, \mathtt{1{:}r})$ ▷ Truncate SVD to rank r
  $\boldsymbol{U} \leftarrow \boldsymbol{U}(\mathtt{:}, \mathtt{1{:}r})$
  $\boldsymbol{V} \leftarrow \boldsymbol{V}(\mathtt{:}, \mathtt{1{:}r})$
  $\boldsymbol{U} \leftarrow \boldsymbol{Q}\boldsymbol{U}$ ▷ Consolidate unitary factors
  $\boldsymbol{V} \leftarrow \boldsymbol{P}\boldsymbol{V}$
  return $(\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V})$

theoretical analysis to be able to implement sketching algorithms with confidence.

In this section, we analyze the low-rank reconstruction algorithms in the ideal case where all of the dimension reduction maps are standard normal. These results allow us to make concrete recommendations for the sketch size parameters. Empirically, other types of dimension reduction exhibit almost identical performance (subsection 7.4), so our analysis also supports more practical implementations based on SSRFTs or sparse sign matrices. The numerical work in section 7 confirms the value of this analysis.

5.1. Notation. For each integer $r \geq 0$, the *tail energy* of the input matrix is

$$\tau_{r+1}^2(\boldsymbol{A}) := \min_{\mathrm{rank}(\boldsymbol{B})\leq r} \|\boldsymbol{A} - \boldsymbol{B}\|_2^2 = \|\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r\|_2^2 = \sum\nolimits_{j>r} \sigma_j^2(\boldsymbol{A}),$$

where σ_j returns the jth largest singular value of a matrix. The second identity follows from the Eckart–Young theorem [36, sect. 6].

We also introduce parameters that reflect the field over which we are working:

$$\alpha := \alpha(\mathbb{F}) := \begin{cases} 1, & \mathbb{F} = \mathbb{R}, \\ 0, & \mathbb{F} = \mathbb{C}, \end{cases} \quad \text{and} \quad \beta := \beta(\mathbb{F}) := \begin{cases} 1, & \mathbb{F} = \mathbb{R}, \\ 2, & \mathbb{F} = \mathbb{C}. \end{cases} \tag{5.1}$$

These quantities let us present real and complex results in a single formula.

5.2. Analysis of initial approximation. The first result gives a bound for the expected error in the initial rank-k approximation $\hat{\boldsymbol{A}}$ of the input matrix $\boldsymbol{A}$.

THEOREM 5.1 (initial approximation: error bound). *Let $\boldsymbol{A} \in \mathbb{F}^{m\times n}$ be an arbitrary input matrix. Assume the sketch size parameters satisfy $s \geq 2k + \alpha$. Draw independent Gaussian dimension reduction maps $(\boldsymbol{\Upsilon}, \boldsymbol{\Omega}, \boldsymbol{\Phi}, \boldsymbol{\Psi})$, as in* (2.2)*. Extract a sketch* (2.3) *and* (2.4) *of the input matrix. Then the rank-k approximation $\hat{\boldsymbol{A}}$, constructed in* (2.9)*, satisfies the error bound*

$$\mathbb{E}\,\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_2^2 \leq \frac{s-\alpha}{s-k-\alpha} \cdot \min_{\varrho<k-\alpha} \frac{k+\varrho-\alpha}{k-\varrho-\alpha} \cdot \tau_{\varrho+1}^2(\boldsymbol{A}). \tag{5.2}$$

We postpone the proof to section SM1. The analysis is similar in spirit to the proof of [69, Thm. 4.3], but it is somewhat more challenging.

Theorem 5.1 contains explicit and reasonable constants, so we can use it to design algorithms that achieve a specific error tolerance. For example, suppose that r_0 is the target rank of the approximation. Then the choice

$$k = 4r_0 + \alpha \quad \text{and} \quad s = 2k + \alpha \tag{5.3}$$

ensures that the expected error in the rank-k approximation $\hat{\boldsymbol{A}}$ is within a constant factor $10/3$ of the optimal rank-r_0 approximation:

$$\mathbb{E}\,\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_2^2 \leq \tfrac{10}{3} \cdot \tau_{r_0+1}^2(\boldsymbol{A}).$$

In practice, we have found the parameter selection (5.3) can be effective for matrices with a rapidly decaying spectrum. Note that, by taking $k/r_0 \to \infty$ and $s/k \to \infty$, we can drive the leading constants in (5.2) to one.

The true meaning of Theorem 5.1 is more subtle. The minimum over ϱ indicates that we can exploit decay in the spectrum of the input matrix by increasing the parameter k. This effect is more significant than the improvement we get from adjusting the parameter s to reduce the first constant. In subsection 5.4, we use this insight to recommend sketch size parameters for a given storage budget.

Remark 5.2 (parameter values). In Theorem 5.1, we have imposed the condition $s \geq 2k + \alpha$ because theoretical analysis and empirical work both suggest that the restriction is useful in practice. The approximation (2.9) only requires that $k \leq s$.

Remark 5.3 (failure probability). Because of measure concentration effects, there is a negligible probability that the error in the initial approximation is significantly larger than the bound (5.2) on the expected error. This claim can be established with techniques from [35, sect. 10]. See subsection 7.9 for numerical evidence.

Remark 5.4 (singular values and vectors). The error bound (5.2) indicates that we can approximate singular values of $\boldsymbol{A}$ by singular values of $\hat{\boldsymbol{A}}$. In particular, an application [11, Prob. III.6.13] of Lidskii's theorem implies that

$$\sum\nolimits_{j=1}^{\min\{m,n\}} \big[\sigma_j(\boldsymbol{A}) - \sigma_j(\hat{\boldsymbol{A}})\big]^2 \leq \|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_2^2.$$

We can also approximate the leading singular vectors of $\boldsymbol{A}$ by the leading singular vectors of $\hat{\boldsymbol{A}}$. Precise statements are slightly complicated, so we refer the reader to [11, Thm. VII.5.9] for a typical result on the perturbation theory of singular subspaces.

5.3. Analysis of truncated approximation. Our second result provides a bound on the error in the truncated approximation $[\![\hat{\boldsymbol{A}}]\!]_r$ of the input matrix $\boldsymbol{A}$.

COROLLARY 5.5 (truncated approximation: error bound). *Instate the assumptions of Theorem* 5.1. *Then the rank-r approximation* $[\![\hat{\boldsymbol{A}}]\!]_r$ *satisfies the error bound*

$$\mathbb{E}\,\|\boldsymbol{A} - [\![\hat{\boldsymbol{A}}]\!]_r\|_2 \leq \tau_{r+1}(\boldsymbol{A}) + 2\left[\frac{s-\alpha}{s-k-\alpha} \cdot \min_{\varrho<k-\alpha} \frac{k+\varrho-\alpha}{k-\varrho-\alpha} \cdot \tau_{\varrho+1}^2(\boldsymbol{A})\right]^{1/2}.$$

This statement is an immediate consequence of Theorem 5.1 and a general bound [69, Prop. 6.1] for fixed-rank approximation. We omit the details.

Let us elaborate on Corollary 5.5. If the initial approximation $\hat{\boldsymbol{A}}$ is accurate, then the truncated approximation $[\![\hat{\boldsymbol{A}}]\!]_r$ attains similar accuracy. In particular, the rank-r approximation can achieve a very small relative error when the input matrix has a decaying spectrum. The empirical work in section 7 highlights the practical importance of this phenomenon.

5.4. Theoretical guidance for sketch size parameters. If we allocate a fixed amount of storage, how can we select the sketch size parameters (k, s) to achieve superior approximations of the input matrix? Using the error bound from Theorem 5.1 and prior knowledge about the spectrum of the matrix, we can make some practical recommendations. Subsection 7.5 offers numerical support for this analysis.

5.4.1. The storage budget. We have recommended using structured dimension reduction maps $(\boldsymbol{\Upsilon}, \boldsymbol{\Omega}, \boldsymbol{\Phi}, \boldsymbol{\Psi})$ so the storage cost for the dimension reduction maps is a fixed cost that does not increase with the sketch size parameters (k, s). Therefore, we may focus on the cost of maintaining the sketch $(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z})$ itself.

Counting dimensions, via (2.3) and (2.4), we see that the three approximation sketch matrices require a total storage budget of

$$T := k(m+n) + s^2 \tag{5.4}$$

floating-point numbers in the field $\mathbb{F}$. How do we best expend this budget?

5.4.2. General spectrum. The theoretical bound Theorem 5.1 on the approximation error suggests that, lacking further information, we should make the parameter k as large as possible. Indeed, the approximation error reflects the decay in the spectrum up to the index k. Meanwhile, the condition $s \geq 2k + \alpha$ in Theorem 5.1 ensures that the first fraction in the error bound cannot exceed 2.

Therefore, for a fixed storage budget T, we pose the optimization problem

$$\text{maximize} \quad k \quad \text{subject to} \quad s \geq 2k + \alpha \quad \text{and} \quad k(m+n) + s^2 = T. \tag{5.5}$$

Up to rounding, the solution is

$$\begin{aligned} k_\natural &:= \left\lfloor \frac{1}{8}\left(\sqrt{(m+n+4\alpha)^2 + 16(T-\alpha^2)} - (m+n+4\alpha)\right)\right\rfloor; \\ s_\natural &:= \left\lfloor \sqrt{T - k_\natural(m+n)} \right\rfloor. \end{aligned} \tag{5.6}$$

The parameter choice $(k_\natural, s_\natural)$ is suitable for a wide range of examples.

5.4.3. Flat spectrum. Suppose we know that the spectrum of the input matrix does not decay past a certain point: $\sigma_j(\boldsymbol{A}) \approx \sigma_{\hat{\varrho}}(\boldsymbol{A})$ for $j > \hat{\varrho}$. In this case, the minimum value of the error (5.2) tends to occur when $\varrho = \hat{\varrho}$.

Algorithm 6.1 Randomized Error Estimator. Implements (6.4).

Input: Matrix approximation $\hat{\boldsymbol{A}}_{\text{out}}$
Output: Randomized error estimate $\text{err}_2^2(\hat{\boldsymbol{A}}_{\text{out}})$ that satisfies (6.5)–(6.7)

```
function SketchErrorEstimate(Â_out)
    β ← 1 for 𝔽 = ℝ or β ← 2 for 𝔽 = ℂ
    err₂² ← (βq)^{-1} ‖W − ΘÂ_out‖₂²
    return err₂²
```

In this case, we can obtain a theoretically supported parameter choice $(k_\flat, s_\flat)$ by numerical solution of the optimization problem

$$\text{minimize} \quad \frac{s-\alpha}{s-k-\alpha} \cdot \frac{k+\hat{\varrho}-\alpha}{k-\hat{\varrho}-\alpha} \quad \text{subject to} \quad s \geq 2k+\alpha, \quad k \geq \hat{\varrho}+\alpha+1, \quad k(m+n)+s^2 = T. \tag{5.7}$$

This problem admits a messy closed-form solution, or it can be solved numerically.

6. A posteriori error estimation. The a priori error bounds from Theorem 5.1 and Corollary 5.5 are essential for setting the sketch size parameters to make the reconstruction algorithm reliable. To evaluate whether the approximation was actually successful, we need a posteriori error estimators.

For this purpose, Martinsson [48, sect. 14] has proposed to extract a very small Gaussian sketch of the input matrix, independent from the approximation sketch. Our deep understanding of the Gaussian distribution allows for a refined analysis of error estimators computed from this sketch.

We adopt Martinsson's idea to compute a simple estimate for the Frobenius norm of the approximation error. Subsection 6.5 explains how this estimator helps us select the precise rank r for the truncated approximation (2.10).

6.1. The error sketch. For a parameter q, draw and fix a standard Gaussian dimension reduction map:

$$\boldsymbol{\Theta} \in \mathbb{F}^{q \times m}. \tag{6.1}$$

Along with the approximation sketch (2.3) and (2.4), we also maintain an error sketch:

$$\boldsymbol{W} := \boldsymbol{\Theta}\boldsymbol{A} \in \mathbb{F}^{q \times n}. \tag{6.2}$$

We can track the error sketch along a sequence (2.5) of linear updates:

$$\boldsymbol{W} \leftarrow \eta\boldsymbol{W} + \nu\boldsymbol{\Theta}\boldsymbol{H}. \tag{6.3}$$

The cost of storing the test matrix and sketch is $q(m+n)$ floating-point numbers.

6.2. A randomized error estimator. Suppose that we have computed an approximation $\hat{\boldsymbol{A}}_{\text{out}}$ of the input $\boldsymbol{A}$ via any method.[6] We can obtain a probabilistic estimate for the squared Schatten 2-norm error in this approximation:

$$\text{err}_2^2(\hat{\boldsymbol{A}}_{\text{out}}) := \frac{1}{\beta q} \cdot \|\boldsymbol{W} - \boldsymbol{\Theta}\hat{\boldsymbol{A}}_{\text{out}}\|_2^2 = \frac{1}{\beta q} \cdot \|\boldsymbol{\Theta}(\boldsymbol{A} - \hat{\boldsymbol{A}}_{\text{out}})\|_2^2. \tag{6.4}$$

[6] We assume only that the approximation $\hat{\boldsymbol{A}}_{\text{out}}$ does not depend on the matrices $\boldsymbol{\Theta}, \boldsymbol{W}$.

Recall that $\beta = 1$ for $\mathbb{F} = \mathbb{R}$ and $\beta = 2$ for $\mathbb{F} = \mathbb{C}$.

The error estimator can be computed efficiently when the approximation is presented in factored form. To assess a rank-r approximation $\hat{\boldsymbol{A}}_{\text{out}}$, the cost is typically $\mathcal{O}(qr(m+n))$ arithmetic operations. See Algorithm 6.1 for pseudocode.

Remark 6.1 (prior work). The formula (6.4) is essentially a randomized trace estimator; for example, see [39, 7, 61, 31]. Our analysis is similar to the work in these papers. Methods for spectral norm estimation are discussed in [74, sect. 3.4] and in [35, sects. 4.3–4.4]; these results trace their lineage to an early paper of Dixon [24]. The paper [45] discusses bootstrap methods for randomized linear algebra applications.

6.3. The error estimator: Mean and variance. The error estimator delivers reliable information about the squared Schatten 2-norm approximation error:

$$\begin{aligned} \mathbb{E}\left[\operatorname{err}_2^2(\hat{\boldsymbol{A}}_{\text{out}})\right] &= \|\boldsymbol{A} - \hat{\boldsymbol{A}}_{\text{out}}\|_2^2, \\ \operatorname{Var}\left[\operatorname{err}_2^2(\hat{\boldsymbol{A}}_{\text{out}})\right] &= \frac{2}{\beta q}\,\|\boldsymbol{A} - \hat{\boldsymbol{A}}_{\text{out}}\|_4^4. \end{aligned} \tag{6.5}$$

These results follow directly from the rotational invariance of the Schatten norms and of the standard normal distribution. See subsection SM2.1.2.

6.4. The error estimator, in probability. We can also obtain bounds on the probability that the error estimator returns an extreme value. These results justify setting the size q of the error sketch to a constant. They are also useful for placing confidence bands on the approximation error. See section SM2 for the proofs.

First, let us state a bound on the probability that the estimator reports a value that is much too small. We have

$$\mathbb{P}_{\boldsymbol{\Theta}}\left\{\operatorname{err}_2^2(\hat{\boldsymbol{A}}_{\text{out}}) \le (1-\varepsilon)\|\boldsymbol{A} - \hat{\boldsymbol{A}}_{\text{out}}\|_2^2\right\} \le \left[\mathrm{e}^{\varepsilon}(1-\varepsilon)\right]^{\beta q/2} \quad \text{for } \varepsilon \in (0,1). \tag{6.6}$$

For example, the error estimate is smaller than $0.1\times$ the true error value with probability less than $2^{-\beta q}$.

Next, we provide a bound on the probability that the estimator reports a value that is much too large. We have

$$\mathbb{P}_{\boldsymbol{\Theta}}\left\{\operatorname{err}_2^2(\hat{\boldsymbol{A}}_{\text{out}}) \ge (1+\varepsilon)\|\boldsymbol{A} - \hat{\boldsymbol{A}}_{\text{out}}\|_2^2\right\} \le \left[\frac{\mathrm{e}^{\varepsilon}}{1+\varepsilon}\right]^{-\beta q/2} \quad \text{for } \varepsilon > 0. \tag{6.7}$$

For example, the error estimate exceeds $4\times$ the true error value with probability less than $2^{-\beta q}$.

Remark 6.2 (estimating normalized errors). We may wish to compute the error of an approximation $\hat{\boldsymbol{A}}_{\text{out}}$ on the scale of the energy $\|\boldsymbol{A}\|_2^2$ in the input matrix. To that end, observe that $\operatorname{err}_2^2(\mathbf{0})$ is an estimate for $\|\boldsymbol{A}\|_2^2$. Therefore, the ratio $\operatorname{err}_2^2(\hat{\boldsymbol{A}}_{\text{out}})/\operatorname{err}_2^2(\mathbf{0})$ gives a good estimate for the normalized error.

6.5. Diagnosing spectral decay. In many applications, our goal is to estimate a rank-r truncated SVD of the input matrix that captures most of its spectral energy. It is rare, however, that we can prophesy the precise value r of the rank. A natural solution is to use the spectral characteristics of the initial approximation $\hat{\boldsymbol{A}}$, defined in (2.9), to decide where to truncate. We can deploy the error estimator err_2^2 to implement this strategy in a principled way and to validate the results. See subsections 7.9 and 7.10 for numerics.

If we had access to the full input matrix $\boldsymbol{A}$, we would compute the proportion of tail energy remaining after a rank-r approximation:

$$\text{scree}(r) := \left[\frac{\tau_{r+1}(\boldsymbol{A})}{\|\boldsymbol{A}\|_2}\right]^2 = \left[\frac{\|\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r\|_2}{\|\boldsymbol{A}\|_2}\right]^2. \tag{6.8}$$

A visualization of the function (6.8) is called a *scree plot*. A standard technique for rank selection is to identify a "knee" in the scree plot. It is also possible to apply quantitative model selection criteria to the function (6.8). See [42, Chap. 6] for an extensive discussion.

We cannot compute (6.8) without access to the input matrix, but we can use the initial approximation and the error estimator creatively. For $r \ll k$, the tail energy $\tau_{r+1}(\hat{\boldsymbol{A}})$ of the initial approximation is a proxy for the tail energy $\tau_{r+1}(\boldsymbol{A})$ of the input matrix. This observation suggests that we consider the (lower) estimate

$$\underline{\text{scree}}(r) := \left[\frac{\tau_{r+1}(\hat{\boldsymbol{A}})}{\text{err}_2(\boldsymbol{0})}\right]^2 = \left[\frac{\|\hat{\boldsymbol{A}} - [\![\hat{\boldsymbol{A}}]\!]_r\|_2}{\text{err}_2(\boldsymbol{0})}\right]^2. \tag{6.9}$$

This function tracks the actual scree curve (6.8) when $r \ll k$. It typically underestimates the scree curve, and the underestimate is severe for large r.

To design a more rigorous approach, notice that

$$|\tau_{r+1}(\boldsymbol{A}) - \tau_{r+1}(\hat{\boldsymbol{A}})| \le \|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_2 \approx \text{err}_2(\hat{\boldsymbol{A}}).$$

The inequality requires a short justification; see subsection SM2.2. This bound suggests that we consider the (upper) estimator

$$\overline{\text{scree}}(r) := \left[\frac{\tau_{r+1}(\hat{\boldsymbol{A}}) + \text{err}_2(\hat{\boldsymbol{A}})}{\text{err}_2(\boldsymbol{0})}\right]^2. \tag{6.10}$$

This function also tracks the actual scree curve (6.8) when $r \ll k$. It reliably overestimates the scree curve by a modest amount.

7. Numerical experiments.

This section presents computer experiments that are designed to evaluate the performance of the proposed sketching algorithms for low-rank matrix approximation. We include comparisons with alternative methods from the literature to argue that the proposed approach produces superior results. We also explore some applications to scientific simulation and data analysis.

7.1. Alternative sketching and reconstruction methods.

We compare the proposed method (2.10) with three other algorithms that construct a fixed-rank approximation of a matrix from a random linear sketch:

1. The [HMT11] method [35, sect. 5.5, Rem. 5.4] is a simplification of the method from Woolfe et al. [74, sect. 5.2], and they perform similarly. There are two sketches, and the sketch size depends on one parameter k. The total storage cost $T = k(m + n)$.
2. The [TYUC17] method [69, Alg. 7] is a numerically stable and a more fully realized implementation of a proposal due to Clarkson and Woodruff [20, Thm. 4.9]. It involves two sketches, controlled by two parameters k, ℓ. The total storage cost $T = km + \ell n$.

3. The [Upa16] method [71, sect. 3.3] simplifies a complicated approach from Boutsidis, Woodruff, and Zhong [15, Thm. 12]. This algorithm involves three sketches, controlled by two parameters k, s. The total storage cost $T = k(m+n) + s^2$.
4. Our new method (2.10) simultaneously extends [Upa16] and [TYUC17]. It uses three sketches, controlled by two parameters k, s. The total storage cost $T = k(m+n) + s^2$.

See subsection SM4.1 for a more detailed description of these methods. In each case, the storage budget neglects the cost of storing the dimension reduction maps because this cost has lower order than the sketch when we use structured dimension reduction maps. These methods have similar arithmetic costs, so we will not make a comparison of runtimes. Storage is the more significant issue for sketching algorithms. We do not include storage costs for an error estimator in the comparisons.

Our recent paper [69] demonstrates that several other methods ([73, Thm. 4.3, display 2] and [23, sect. 10.1]) are uncompetitive, so we omit them.

7.2. Experimental setup. Our experimental design is quite similar to our previous papers [69, 68] on sketching algorithms for low-rank matrix approximation.

7.2.1. Procedure. Fix an input matrix $\boldsymbol{A} \in \mathbb{F}^{n\times n}$ and a truncation rank r. Select sketch size parameters. For each trial, draw dimension reduction maps from a specified distribution and form the sketch of the input matrix. Compute a rank-r approximation $\hat{\boldsymbol{A}}_{\text{out}}$ using a specified reconstruction algorithm. The approximation error is calculated relative to the best rank-r approximation error in Schatten p-norm:

$$S_p \text{ relative error} \quad = \quad \frac{\|\boldsymbol{A} - \hat{\boldsymbol{A}}_{\text{out}}\|_p}{\|\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r\|_p} - 1. \tag{7.1}$$

We perform 20 independent trials and report the average error. Owing to measure concentration effects, the average error is also the typical error; see subsection 7.9.

In all experiments, we work in double-precision arithmetic (i.e., 8 bytes per real floating-point number). The body of this paper presents a limited selection of results. Section SM4 contains additional numerical evidence. The supplementary materials also include MATLAB code that can reproduce these experiments.

7.2.2. The oracle error. To make fair comparisons among algorithms, we can fix the storage budget and identify the parameter choices that minimize the (average) relative error (7.1) incurred over the repeated trials. We refer to the minimum as the *oracle error* for an algorithm. The oracle error is not attainable in practice.

7.3. Classes of input matrices. As in our previous papers [68, 69], we consider several different types of synthetic and real input matrices. See Figure SM1 for a plot of the spectra of these input matrices.

7.3.1. Synthetic examples. We work over the complex field $\mathbb{C}$. The matrix dimensions $m = n = 10^3$, and we introduce an effective rank parameter $R \in \{5, 10, 20\}$. In each case, we compute an approximation with truncation rank $r = 10$.

1. **Low-rank + noise:** Let $\xi \geq 0$ be a signal-to-noise parameter. These matrices take the form

$$\boldsymbol{A} = \operatorname{diag}(\underbrace{1, \dots, 1}_{R}, 0, \dots, 0) + \xi n^{-1} \boldsymbol{E} \in \mathbb{C}^{n\times n},$$

where $\boldsymbol{E} = \boldsymbol{G}\boldsymbol{G}^*$ for a standard normal matrix $\boldsymbol{G} \in \mathbb{F}^{n\times n}$. We consider several parameter values: `LowRankLowNoise` ($\xi = 10^{-4}$), `LowRankMedNoise` ($\xi = 10^{-2}$), `LowRankHiNoise` ($\xi = 10^{-1}$).

2. **Polynomial decay:** For a decay parameter $p > 0$, consider matrices

$$\boldsymbol{A} = \operatorname{diag}(\underbrace{1, \dots, 1}_{R}, 2^{-p}, 3^{-p}, \dots, (n-R+1)^{-p}) \in \mathbb{C}^{n\times n}.$$

We study three examples: `PolyDecaySlow` ($p = 0.5$), `PolyDecayMed` ($p = 1$), `PolyDecayFast` ($p = 2$).

3. **Exponential decay:** For a decay parameter $q > 0$, consider matrices

$$\boldsymbol{A} = \operatorname{diag}(\underbrace{1, \dots, 1}_{R}, 10^{-q}, 10^{-2q}, \dots, 10^{-(n-R)q}) \in \mathbb{C}^{n\times n}.$$

We consider the cases `ExpDecaySlow` ($q = 0.01$), `ExpDecayMed` ($q = 0.1$), `ExpDecayFast` ($q = 0.5$).

Remark 7.1 (nondiagonal matrices). We have also performed experiments using nondiagonal matrices with the same spectra. The results were essentially identical.

7.3.2. Application examples. Next, we present some low-rank data matrices that arise in applications. The truncation rank r varies, depending on the matrix.

1. **Navier–Stokes:** We test the hypothesis, discussed in subsection 1.2, that sketching methods can be used to perform on-the-fly compression of the output of a PDE simulation. We have obtained a direct numerical simulation (DNS) on a coarse mesh of the 2D Navier–Stokes equations for a low Reynolds number flow around a cylinder. The simulation is started impulsively from a rest state. Transient dynamics emerge in the first third of the simulation, while the remaining time steps capture the limit cycle. Each of the velocity and pressure fields is centered around its temporal mean. This data is courtesy of Beverley McKeon and Sean Symon.

   The real $m \times n$ matrix `StreamVel` contains streamwise velocities at $m = 10,738$ points for each of $n = 5,001$ time instants. The first 20 singular values of the matrix decay by two orders of magnitude, and the rest of the spectrum exhibits slow exponential decay.

2. **Weather:** We test the hypothesis that sketching methods can be used to perform on-the-fly compression of temporal data as it is collected. We have obtained a matrix that tabulates meteorological variables at weather stations across the northeastern United States on days during the years 1981–2016. This data is courtesy of William North.

   The real $m \times n$ matrix `MinTemp` contains the minimum temperature recorded at each of $m = 19,264$ stations on each of $n = 7,305$ days. The first 10 singular values decay by two orders of magnitude, while the rest of the spectrum has medium polynomial decay.

3. **Sketchy decisions:** We also consider matrices that arise from an optimization algorithm for solving large-scale semidefinite programs [75]. In this application, the data matrices are presented as a long series of rank-one updates, and sketching is a key element of the algorithm.

(a) `MaxCut`: This is a real positive-semidefinite (psd) matrix with $m = n = 2,000$ that gives a high-accuracy solution to the MaxCut SDP for a sparse graph [30]. This matrix is effectively rank deficient with $R = 14$, and the spectrum has fast exponential decay after this point.

(b) `PhaseRetrieval`: This is a complex psd matrix with $m = n = 25,921$ that gives a low-accuracy solution to a phase retrieval SDP [38]. This matrix is effectively rank deficient with $R = 5$, and the spectrum has fast exponential decay after this point.

4. **Sea surface temperature data**: Finally, we use a moderately large climate dataset to showcase our overall methodology. This data is provided by the National Oceanic and Atmospheric Administration (NOAA); see [58, 59] for details about the data preparation methodology.

   The real $m \times n$ matrix `SeaSurfaceTemp` consists of daily temperature estimates at $m = 691,150$ regularly spaced points in the ocean for each of $n = 13,670$ days between 1981 and 2018.

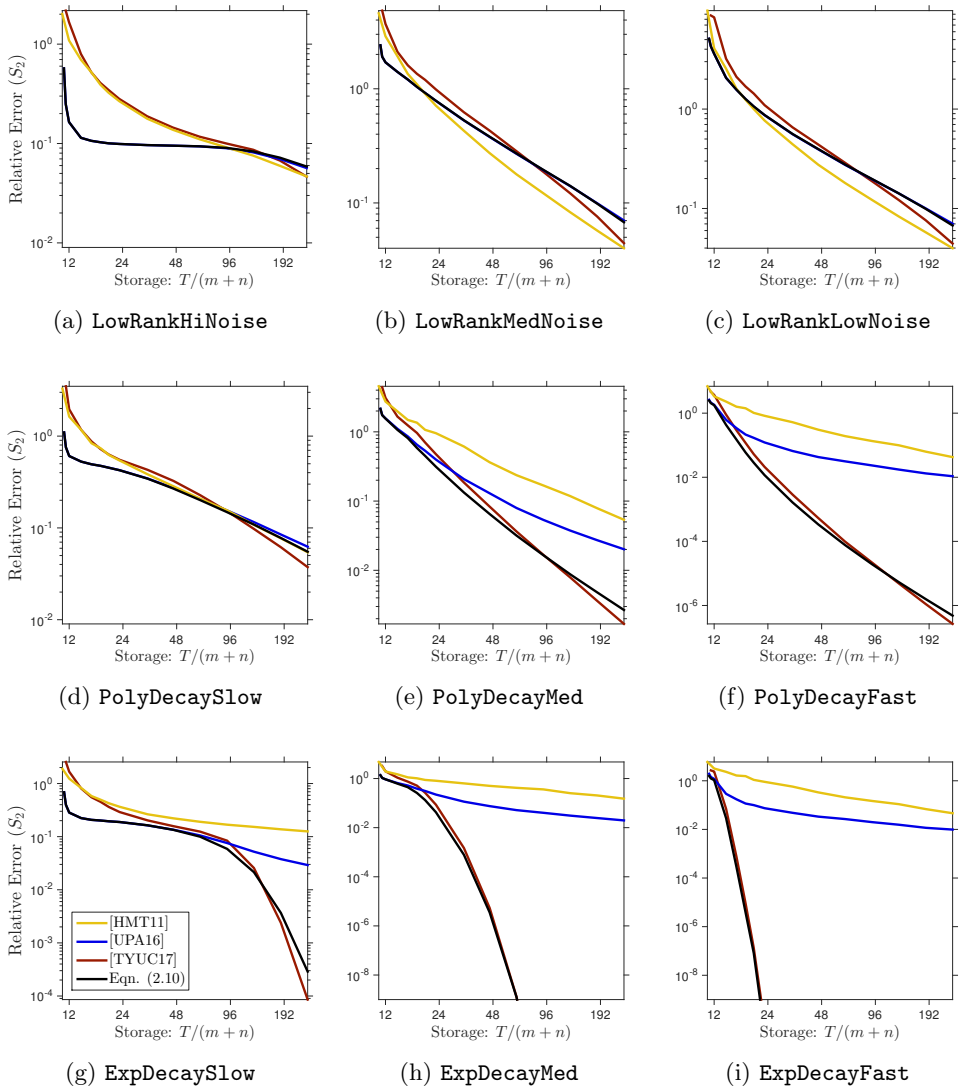
7.4. Insensitivity to dimension reduction map. The proposed reconstruction method (2.10) is insensitive to the choice of dimension reduction map at the oracle parameter values (subsection 7.2.2). As a consequence, we can transfer theoretical and empirical results for Gaussians to SSRFT and sparse dimension reduction maps. See subsection SM4.3 for numerical evidence.

7.5. Approaching the oracle performance. We can almost achieve the oracle error by implementing the reconstruction method (2.10) with sketch size parameters chosen using the theory in subsection 5.4. This observation justifies the use of the theoretical parameters when we apply the algorithm. See subsection SM4.4 for numerical evidence.

7.6. Comparison of reconstruction formulas: Synthetic examples. Let us now compare the proposed rank-r reconstruction formula (2.10) with [HMT11], [Upa16], and [TYUC17] on synthetic data.

Figure 1 presents the results of the following experiment. For synthetic matrices with effective rank $R = 10$ and truncation rank $r = 10$, we compare the relative error (7.1) achieved by each of the four algorithms as a function of storage. We use Gaussian dimension reduction maps in these experiments; similar results are evident for other types of maps. Results for effective rank $R \in \{5, 20\}$ and Schatten ∞-norm appear in subsection SM4.5. Let us make some remarks:

- This experiment demonstrates clearly that the proposed approximation (2.10) improves over the earlier methods for most of the synthetic input matrices, almost uniformly and sometimes by orders of magnitude.
- For input matrices where the spectral tail decays slowly (`PolyDecaySlow`, `LowRankLowNoise`, `LowRankMedNoise`, `LowRankHiNoise`), the newly proposed method (2.10) has identical behavior to [Upa16]. The new method is slightly worse than [HMT11] in several of these cases.
- For input matrices whose spectral tail decays more quickly (`ExpDecaySlow`, `ExpDecayMed`, `ExpDecayFast`, `PolyDecayMed`, `PolyDecayFast`), the proposed method improves dramatically over [HMT11] and [Upa16].
- The new method (2.10) shows its strength over [TYUC17] when the storage budget is small. It also yields superior performance in Schatten ∞-norm.



(a) `LowRankHiNoise` (b) `LowRankMedNoise` (c) `LowRankLowNoise`

(d) `PolyDecaySlow` (e) `PolyDecayMed` (f) `PolyDecayFast`

(g) `ExpDecaySlow` (h) `ExpDecayMed` (i) `ExpDecayFast`

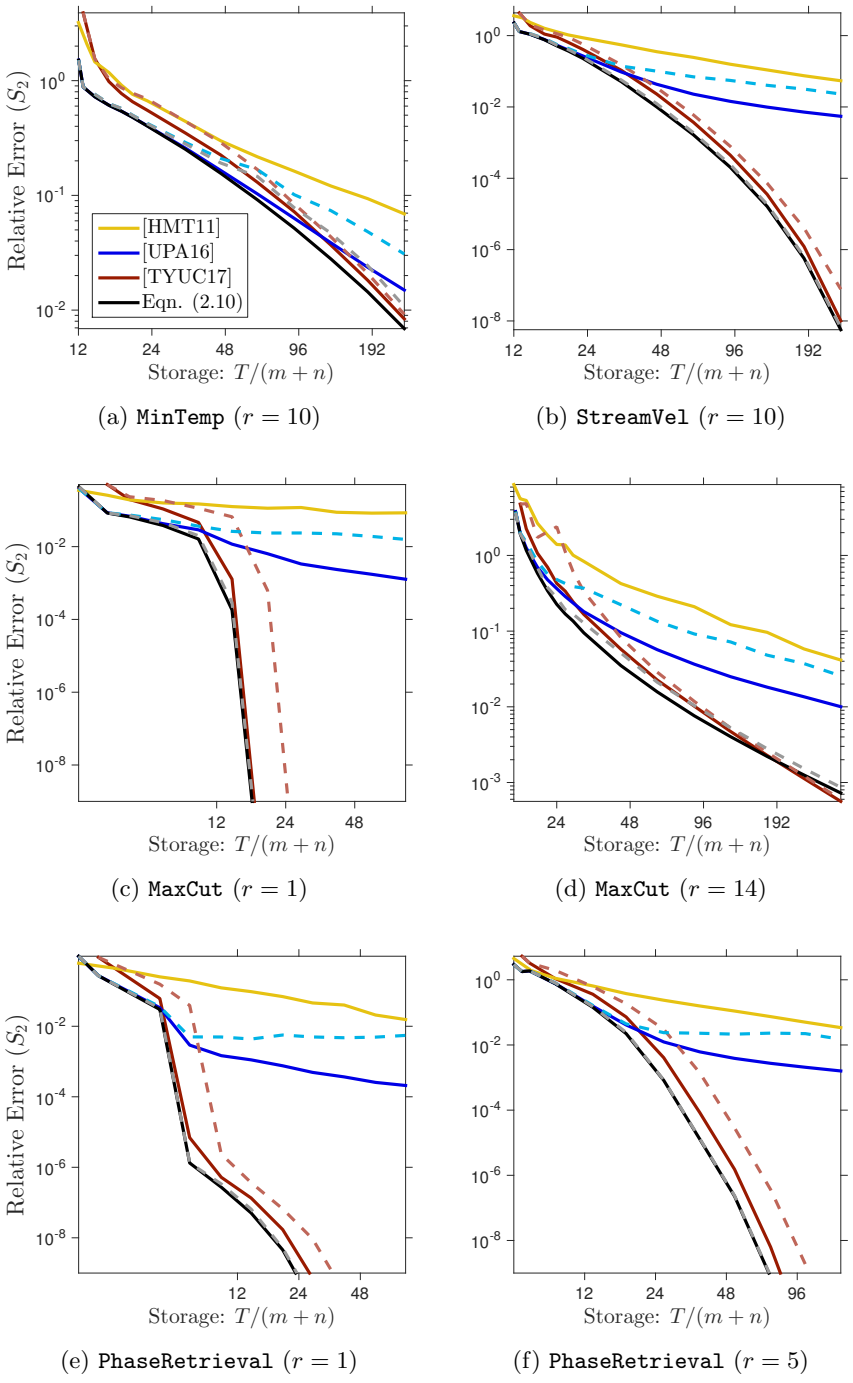
FIG. 1. Comparison of reconstruction formulas: Synthetic examples. *(Gaussian maps, effective rank $R = 10$, approximation rank $r = 10$, Schatten 2-norm.) We compare the oracle error achieved by the proposed fixed-rank approximation* (2.10) *against methods [HMT11], [Upa16], and [TYUC17] from the literature. See subsection* 7.2.2 *for details.*

These differences are most evident for matrices with slow spectral decay.

7.7. Comparison of reconstruction formulas: Real data examples. The next set of experiments compares the behavior of the algorithms for matrices drawn from applications.

Figure 2 contains the results of the following experiment. For each of the four algorithms, we display the relative error (7.1) as a function of storage. We use sparse dimension reduction maps, which is justified by the experiments in subsection 7.4.

We plot the oracle error (subsection 7.2.2) attained by each method. Since the



(a) `MinTemp` ($r = 10$) (b) `StreamVel` ($r = 10$)

(c) `MaxCut` ($r = 1$) (d) `MaxCut` ($r = 14$)

(e) `PhaseRetrieval` ($r = 1$) (f) `PhaseRetrieval` ($r = 5$)

FIG. 2. Comparison of reconstruction formulas: Real data examples. *(Sparse maps, Schatten 2-norm.) We compare the relative error achieved by the proposed fixed-rank approximation* (2.10) *against methods [HMT11], [Upa16], and [TYUC17] from the literature.* Solid lines *are oracle errors;* dashed lines *are errors with "natural" parameter choices. (There is no dashed line for [HMT11].) See subsection* 7.7 *for details.*

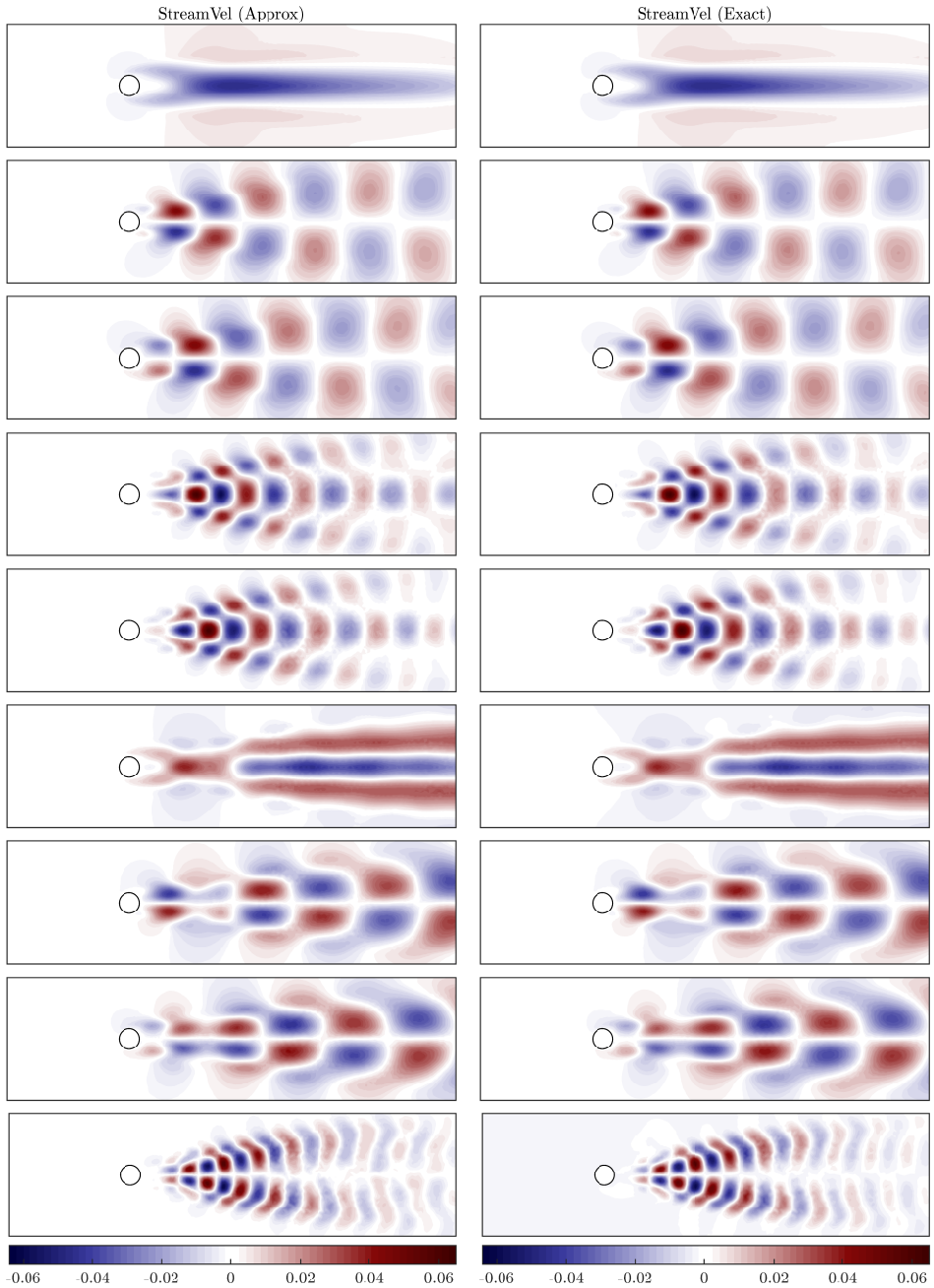


FIG. 3. Left singular vectors of `StreamVel`. *(Sparse maps, approximation rank $r = 10$, storage budget $T = 48(m + n)$.) The columns of the matrix* `StreamVel` *describe the fluctuations of the streamwise velocity field about its mean value as a function of time. From top to bottom, the panels show the first nine computed left singular vectors of the matrix. The* left-hand side *is computed from the sketch, while the* right-hand side *is computed from the exact flow field. The heatmap indicates the magnitude of the fluctuation. See subsection* 7.8 *for details.*

oracle error is not achievable in practice, we also chart the performance of each method at an a priori parameter selection; see subsection SM4.6 for details.

As with the synthetic examples, the proposed method (2.10) improves over the competing methods for all the examples we considered. This is true when we compare oracle errors or when we compare the errors using theoretical parameter choices. The benefits of the new method are least pronounced for the matrix `MinTemp`, whose spectrum has medium polynomial decay. The benefits of the new method are quite clear for the matrix `StreamVel`, which has an exponentially decaying spectrum. The advantages are even more striking for the two matrices `MaxCut` and `PhaseRetrieval`, which are effectively rank deficient.

In summary, we believe that the numerical work here supports the use of our new method (2.10). The methods [HMT11] and [Upa16] cannot achieve a small relative error (7.1), even with a large amount of storage. The method [TYUC17] can yield small relative error, but it often requires more storage to achieve this goal—especially at the a priori parameter choices.

7.8. Example: Flow-field reconstruction. Next, we elaborate on using sketching to compress the Navier–Stokes data matrix `StreamVel`. We compute the best rank-10 approximation of the matrix via (2.10) using storage $T = 48(m + n)$ and the "natural" parameter choices (5.6). For this example, we can use plots of the flow field to make visual comparisons.

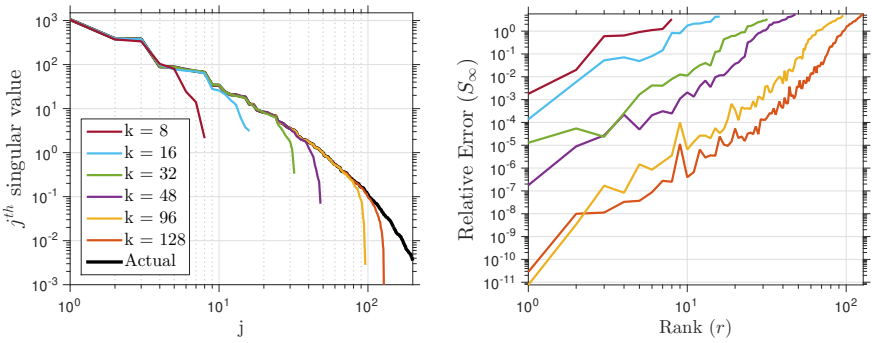
Figure 3 illustrates the leading left singular vectors of the streamwise velocity field `StreamVel`, as computed from the sketch and the full matrix. We see that the approximate left singular vectors closely match the actual left singular vectors, although some small discrepancies appear in the higher singular vectors. See subsection SM4.7 for additional numerics. In particular, we find that the output from the algorithms [HMT11] and [Upa16] changes violently when we adjust the truncation rank r.

We see that our sketching method leads to an excellent rank-10 approximation of the matrix. In fact, the relative error (7.1) in Frobenius norm is under $9.2 \cdot 10^{-3}$. While the sketch uses 5.8 MB of storage in double precision, the full matrix requires 409.7 MB. The compression rate is 70.6×. Therefore, it is possible to compress the output of the Navier–Stokes simulation automatically using sketching.

7.9. Rank truncation and a posteriori error estimation. This section uses the Navier–Stokes data to explore the behavior of the error estimator (subsection 6.2). We also demonstrate that it is important to truncate the rank of the approximation, and we show that the error estimator can assist us.

Let us undertake a single trial of the following experiment with the matrix `StreamVel`. For each sketch size parameter $k \in \{1, 2, \dots, 128\}$, set the other sketch size parameter $s = 2k + 1$. Extract an error sketch with size $q = 10$. In each instance, we use the formula (2.9) to construct an initial rank-k approximation $\hat{\boldsymbol{A}}$ of the data matrix $\boldsymbol{A}$ and the formula (2.10) to construct a truncated rank-r approximation $[\![\hat{\boldsymbol{A}}]\!]_r$. The plots will be indexed with the sketch size parameter k or the rank truncation parameter r, rather than the storage budget.

Figure 4 illustrates the need to truncate the rank of the approximation. Observe that the singular values $\sigma_r(\hat{\boldsymbol{A}})$ of the rank-k approximation significantly underestimate the singular values $\sigma_r(\boldsymbol{A})$ of the matrix when $r \approx k$. As a consequence, the Schatten-∞ error (7.1) in the rank-k approximation $\hat{\boldsymbol{A}}$, relative to the best rank-k approximation of $\boldsymbol{A}$, actually *increases* with k. In contrast, when $r \ll k$, the rank-r truncation $[\![\hat{\boldsymbol{A}}]\!]_r$ can attain very small error, relative to the best rank-r approximation of $\boldsymbol{A}$. In this instance, we achieve relative error below 10^{-4} across a range of



(a) Singular values $\sigma_r(\hat{\boldsymbol{A}})$ (b) Rank-r truncations

FIG. 4. Why truncate? *(`StreamVel`, sparse maps, $s = 2k+1$.) Figure* 4a *compares the singular values $\sigma_r(\hat{\boldsymbol{A}})$ of the rank-k approximation $\hat{\boldsymbol{A}}$ against the singular values $\sigma_r(\boldsymbol{A})$ of the actual matrix $\boldsymbol{A}$. Figure* 4b *shows the Schatten-∞ relative error* (7.1) *in the truncated approximation $[\![\hat{\boldsymbol{A}}]\!]_r$ as a function of rank r. The relative error in the (untruncated) rank-k approximation (*right endpoints *of series)* increases *with k, and it can reach* 500%. *See subsection* 7.9.

TABLE 1

A posteriori error evaluation of sea surface temperature approximations. *This table lists the lower and upper estimates for the true scree curve* (6.8) *of the matrix `SeaSurfaceTemp`. We truncate at rank $r = 5$* (shaded).

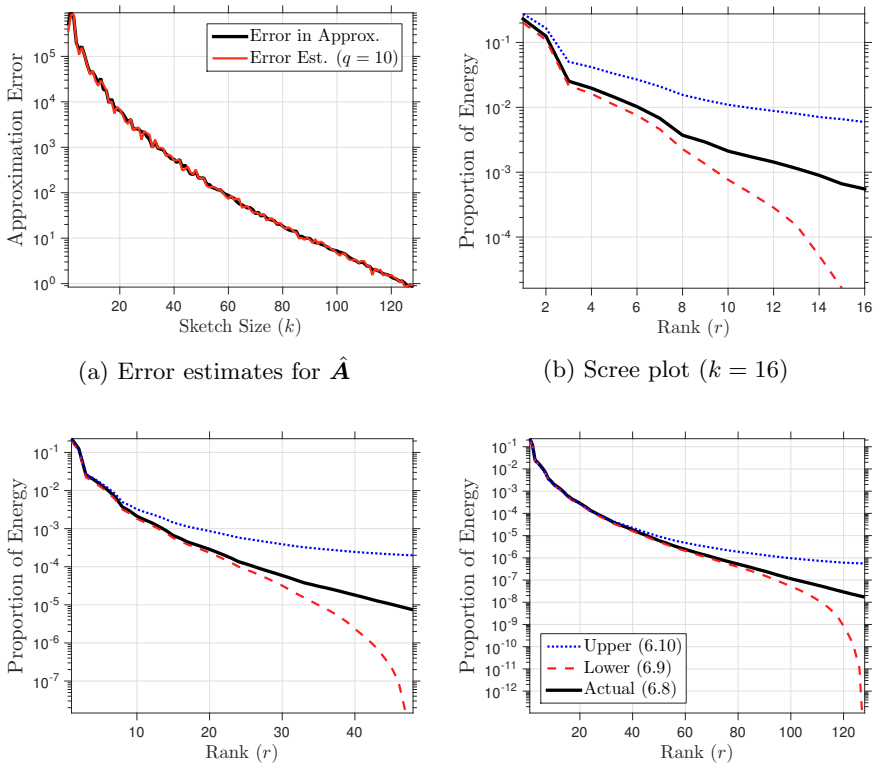
Rank (r)	**Lower estimate (6.9)** $\underline{\text{scree}}(r)$	**Upper estimate (6.10)** $\overline{\text{scree}}(r)$
1	$2.5415 \cdot 10^{-2}$	$5.2454 \cdot 10^{-2}$
2	$2.1068 \cdot 10^{-3}$	$1.3342 \cdot 10^{-2}$
3	$1.2867 \cdot 10^{-3}$	$1.1126 \cdot 10^{-2}$
4	$5.8939 \cdot 10^{-4}$	$8.8143 \cdot 10^{-3}$
5	$3.9590 \cdot 10^{-4}$	$8.0110 \cdot 10^{-3}$
6	$3.0878 \cdot 10^{-4}$	$7.6002 \cdot 10^{-3}$
7	$2.5140 \cdot 10^{-4}$	$7.3039 \cdot 10^{-3}$
8	$2.1541 \cdot 10^{-4}$	$7.1038 \cdot 10^{-3}$
9	$1.8673 \cdot 10^{-4}$	$6.9342 \cdot 10^{-3}$
10	$1.6410 \cdot 10^{-4}$	$6.7926 \cdot 10^{-3}$

parameters k by selecting $r \leq k/4$. Therefore, we can be confident about the quality of our estimates for the first r singular vectors of $\boldsymbol{A}$, given big enough spectral gaps.

Next, let us study the behavior of the error estimator (6.4). Figure 5 compares the actual approximation error $\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_2^2$ and the empirical error estimate $\text{err}_2^2(\hat{\boldsymbol{A}})$ as a function of the sketch size k. The other panels are scree plots. The baseline is the actual scree function (6.8) computed from the input matrix. The remaining curves are the lower (6.9) and upper (6.10) estimates for this curve. We see that the scree estimators give good lower and upper bounds for the energy missed, while tracking the shape of the baseline curve. As a consequence, we can use these empirical estimates to select the truncation rank.

Last, we investigate the sampling distribution of the error in the randomized matrix approximation and the sampling distribution of the error estimator. To do so, we perform 1000 independent trials of the same experiment for select values of k and with error sketch sizes $q \in \{5, 10\}$.

Figure 6 contains scatter plots of the actual approximation error $\|\boldsymbol{A} - \hat{\boldsymbol{A}}_{\text{out}}\|_2^2$



(a) Error estimates for $\hat{\boldsymbol{A}}$

(b) Scree plot ($k = 16$)

(c) Scree plot ($k = 48$)

(d) Scree plot ($k = 128$)

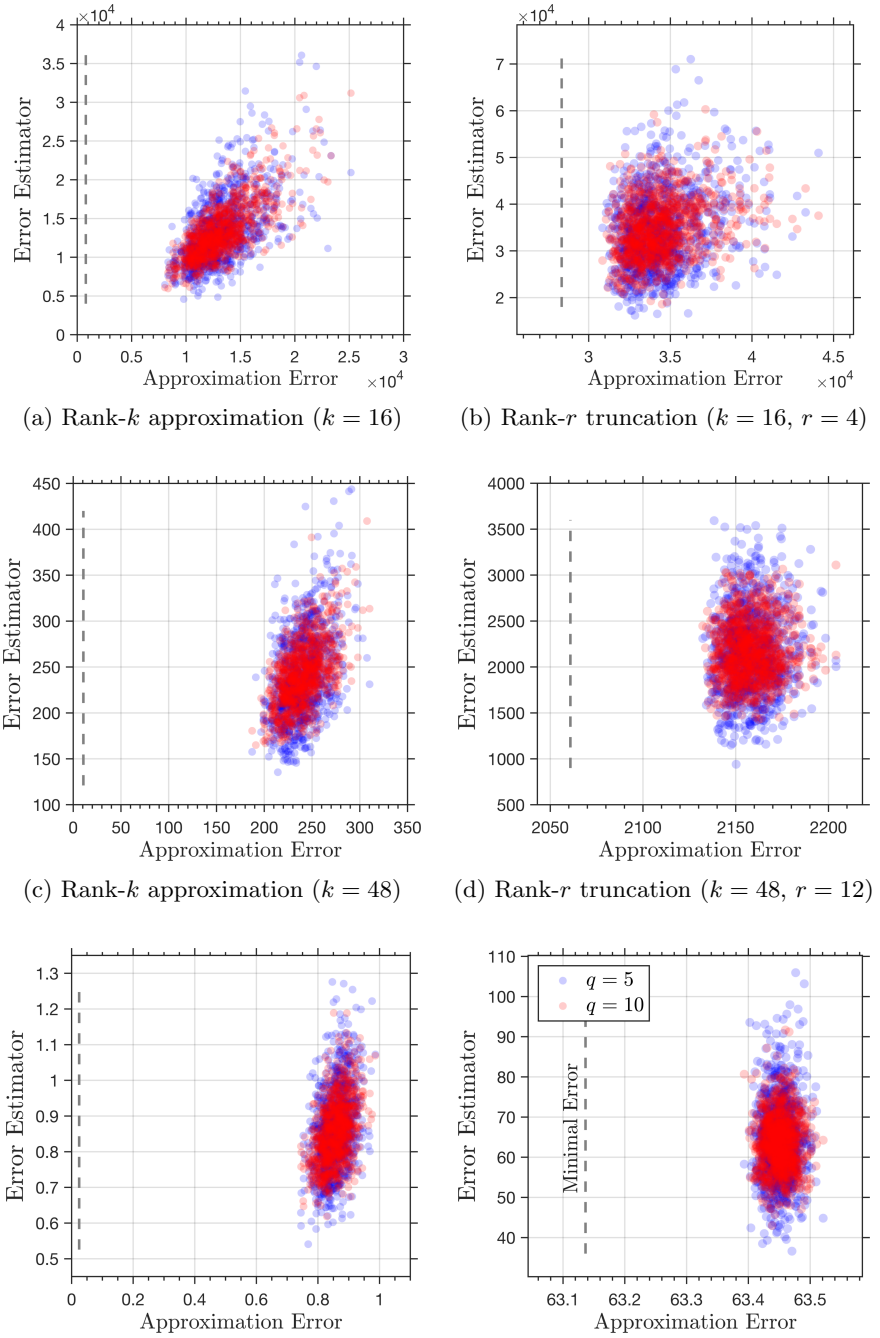
FIG. 5. Error estimation and scree plots. *(`StreamVel`, sparse maps, $s = 2k + 1$.) For an error sketch with size $q = 10$, Figure* 5a *compares the absolute error $\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_2^2$ in the rank-k approximation versus the estimate $\mathrm{err}_2^2(\hat{\boldsymbol{A}})$. The other panels are scree plots of the actual proportion of energy remaining* (6.8) *versus a computable lower estimate* (6.9) *and upper estimate* (6.10). *See subsection* 7.9.

versus the estimated approximation error $\mathrm{err}_2^2(\hat{\boldsymbol{A}}_{\mathrm{out}})$ for $\hat{\boldsymbol{A}}_{\mathrm{out}} = \hat{\boldsymbol{A}}$ and $\hat{\boldsymbol{A}}_{\mathrm{out}} = [\![\hat{\boldsymbol{A}}]\!]_r$. The error estimators are unbiased, but they exhibit a lot of variability. Doubling the error sketch size q reduces the spread of the error estimate by a factor of two. The approximation errors cluster tightly, as we expect from concentration of measure. The plots also highlight that the initial rank-k approximations are far from attaining the minimal rank-k error, while the truncated rank-r approximations are more successful.

7.10. Example: Sea surface temperature data. Finally, we give a complete demonstration of the overall methodology for the matrix `SeaSurfaceTemp`. Like the matrix `MinTemp`, we expect that the sea surface temperature data has medium polynomial decay, so it should be well approximated by a low-rank matrix.

1. **Parameter selection.** We fix the storage budget $T = 48(m + n)$. The natural parameter selection (5.6) yields $k = 47$ and $s = 839$. We use sparse dimension reduction maps. The error sketch has size $q = 10$.

2. **Data collection.** We "stream" the data one year at a time to construct the approximation and error sketches.



(a) Rank-k approximation ($k = 16$)

(b) Rank-r truncation ($k = 16$, $r = 4$)

(c) Rank-k approximation ($k = 48$)

(d) Rank-r truncation ($k = 48$, $r = 12$)

(e) Rank-k approximation ($k = 128$)

(f) Rank-r truncation ($k = 128$, $r = 32$)

FIG. 6. Sampling distributions of the approximation error and the error estimator. *(`StreamVel`, sparse maps, $s = 2k + 1$, Schatten 2-norm.) For error sketches with size $q \in \{5, 10\}$, the* left-hand side *shows the sampling distribution of the error $\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_2^2$ in the rank-k approximation versus the sampling distribution of the error estimator $\mathrm{err}_2^2(\hat{\boldsymbol{A}})$ for several values of k. The* dashed line *marks the error in the best rank-k approximation of $\boldsymbol{A}$. The* right-hand side *contains similar plots with $\hat{\boldsymbol{A}}$ replaced by the rank-r truncation $[\![\hat{\boldsymbol{A}}]\!]_r$. See subsection* 7.9.

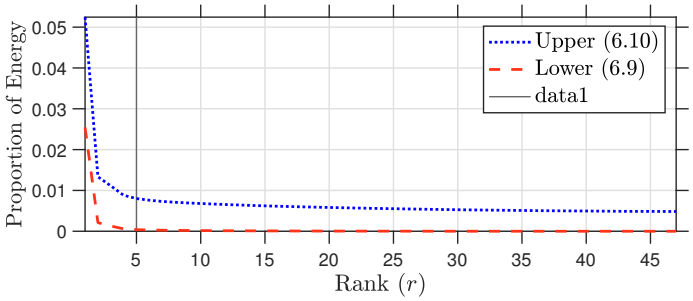


FIG. 7. Empirical scree plot for `SeaSurfaceTemp` approximation. *(Sparse maps,* $k = 48$*,* $s = 839$*,* $q = 10$*.) The lower* (6.9) *and upper* (6.10) *approximations of the scree curve* (6.8)*. The* vertical line *marks the truncation rank* $r = 5$*. See subsection* 7.10 *and Table* 1 *for details.*

3. **Error estimates and rank truncation.** Once the data is collected, we compute the rank-k approximation $\hat{\boldsymbol{A}}$ using the formula (2.9). We present the empirical scree estimates (6.9) and (6.10) in Figure 7 and Table 1. These values should bracket the unknown scree curve (6.8), while mimicking its shape. By visual inspection, we set the truncation rank $r = 5$. We expect that the rank-5 approximation captures all but 0.04% to 0.8% of the energy.

4. **Visualization.** Figure 8 illustrates the first five singular vector pairs of the rank-r approximation of the matrix `SeaSurfaceTemp`. The first left singular vector can be interpreted as the mean temperature profile; a warming trend is visible in the first right singular vector. The second pair reflects the austral/boreal divide. The remaining singular vectors capture long-term climatological features.

The total storage required for the approximation sketch and the error sketch is $4.09 \cdot 10^7$ numbers. This stands in contrast to the $mn = 9.09 \cdot 10^9$ numbers appearing in the matrix itself. The compression ratio is 222×. Moreover, the computational time required to obtain the approximation is modest because we are working with substantially smaller matrices.

8. Conclusions. This paper exhibits a sketching method and a new reconstruction algorithm for low-rank approximation of matrices that are presented as a sequence of linear updates (section 2). The algorithm is accompanied by a priori error bounds that allow us to set algorithm parameters reliably (section 5), as well as an a posteriori error estimator that allows us to validate its performance and to select the final rank of the approximation (section 6). We discuss implementation issues (sections 3 and 4), and we present numerical experiments to show that the new method improves over existing techniques (subsections 7.6 and 7.7).

A potential application of these techniques is for on-the-fly-compression of large-scale scientific simulations and data collection. Our experiments with a Navier–Stokes simulation (subsection 7.8) and with sea surface temperature data (subsection 7.10) both support this hypothesis. We hope that this work motivates researchers to investigate the use of sketching in new applications.

Acknowledgments. The authors wish to thank Beverley McKeon and Sean Symon for providing the Navier–Stokes simulation data and visualization software. William North contributed the weather data. The `NOAA_OI_SST_V2` high-resolution

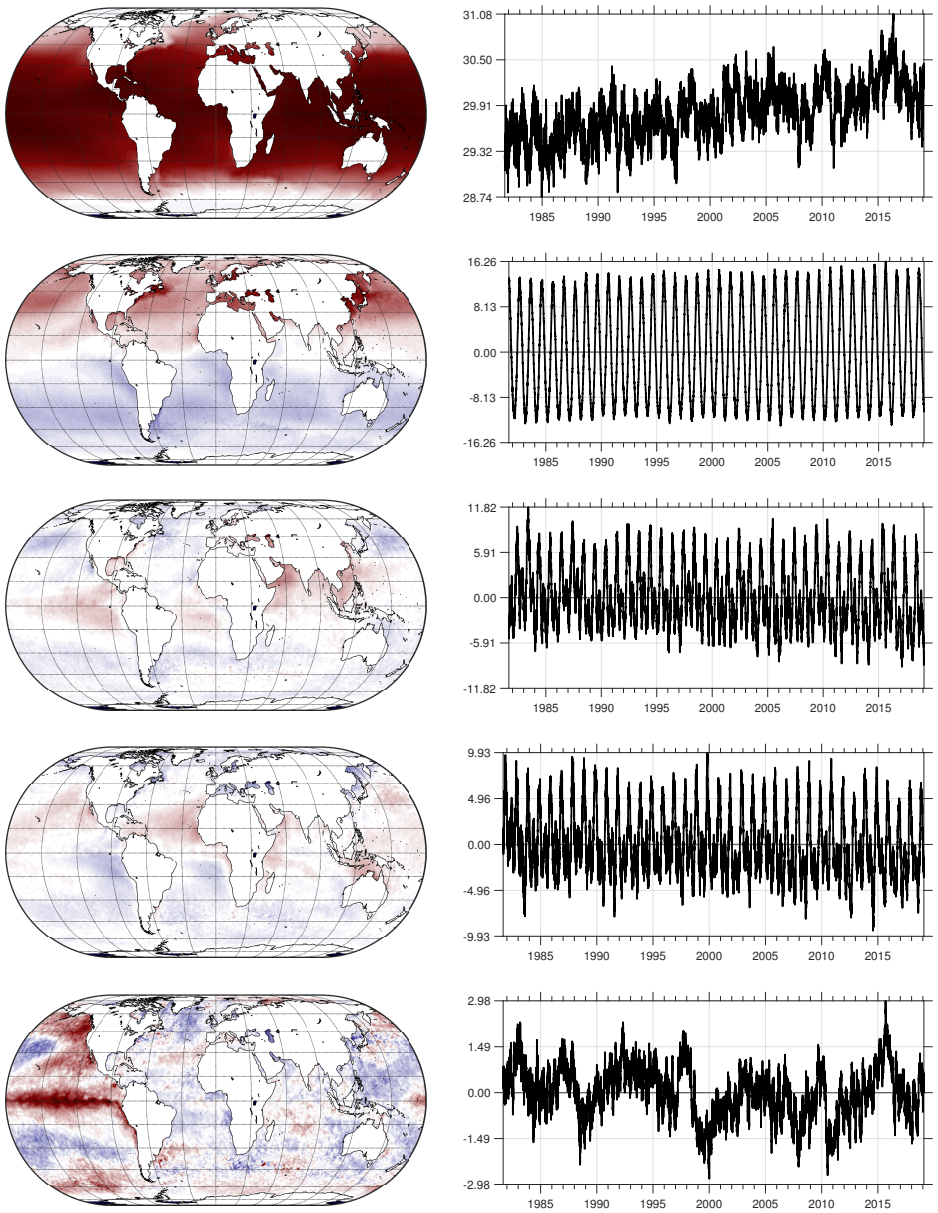


FIG. 8. Singular vectors of `SeaSurfaceTemp`. *(Sparse maps, $k = 48$, $s = 839$, $q = 10$.) The* left column *displays the first five left singular vectors. The heatmaps use* white *to represent zero; each image is scaled independently. The* right column *displays the first five right singular vectors. The horizontal axis marks the year (common era); the vertical axis is temperature (Celsius degrees). The plots are scaled so that the largest peak in the time series equals the largest magnitude temperature in the associated spatial component. See subsection* 7.10.

SST data are provided by NOAA/OAR/ESRL PSD, Boulder, CO, USA. We are also grateful to the anonymous reviewers for a careful reading and thoughtful comments that helped us to improve the manuscript.

REFERENCES

[1] *Physical Sciences Division, National Oceanic and Atmospheric Administration*, Feb. 2019, https://www.esrl.noaa.gov/psd/.

[2] D. Achlioptas, *Database-friendly random projections: Johnson–Lindenstrauss with binary coins*, J. Comput. System Sci., 66 (2003), pp. 671–687.

[3] N. Ailon and B. Chazelle, *The fast Johnson–Lindenstrauss transform and approximate nearest neighbors*, SIAM J. Comput., 39 (2009), pp. 302–322, https://doi.org/10.1137/060673096.

[4] N. Alon, P. B. Gibbons, Y. Matias, and M. Szegedy, *Tracking join and self-join sizes in limited storage*, J. Comput. System Sci., 64 (2002), pp. 719–747, https://doi.org/10.1006/jcss.2001.1813.

[5] N. Alon, Y. Matias, and M. Szegedy, *The space complexity of approximating the frequency moments*, in Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (Philadelphia, PA, 1996), ACM, New York, 1996, pp. 20–29, https://doi.org/10.1145/237814.237823.

[6] W. Austin, G. Ballard, and T. G. Kolda, *Parallel tensor compression for large-scale scientific data*, in Proceedings of the 2016 IEEE International Symposium on Parallel and Distributed Processing, IEEE, Washington, DC, 2016, pp. 912–922.

[7] H. Avron and S. Toledo, *Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix*, J. ACM, 58 (2011), 8, https://doi.org/10.1145/1944345.1944349.

[8] A. H. Baker, H. Xu, J. M. Dennis, M. N. Levy, D. Nychka, S. A. Mickelson, J. Edwards, M. Vertenstein, and A. Wegener, *A methodology for evaluating the impact of data compression on climate simulation data*, in Proceedings of the 23rd ACM International Symposium on High-Performance Parallel and Distributed Computing, ACM, New York, 2014, pp. 203–214.

[9] R. Baurle, *Modeling of high speed reacting flows: Established practices and future challenges*, in Proceedings of the 42nd AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, 2004, p. 267.

[10] A. Bejan, *Convection Heat Transfer*, John Wiley & Sons, New York, 2013.

[11] R. Bhatia, *Matrix Analysis*, Grad. Texts in Math. 169, Springer-Verlag, New York, 1997, https://doi.org/10.1007/978-1-4612-0653-8.

[12] J. Bourgain, S. Dirksen, and J. Nelson, *Toward a unified theory of sparse dimensionality reduction in Euclidean space*, Geom. Funct. Anal., 25 (2015), pp. 1009–1088, https://doi.org/10.1007/s00039-015-0332-9.

[13] C. Boutsidis and A. Gittens, *Improved matrix algorithms via the subsampled randomized Hadamard transform*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 1301–1340, https://doi.org/10.1137/120874540.

[14] C. Boutsidis, D. Woodruff, and P. Zhong, *Optimal Principal Component Analysis in Distributed and Streaming Models*, preprint, https://arxiv.org/abs/1504.06729, 2016.

[15] C. Boutsidis, D. Woodruff, and P. Zhong, *Optimal principal component analysis in distributed and streaming models*, in Proceedings of the 48th ACM Symposium on Theory of Computing (STOC 2016), Cambridge, MA, 2016, pp. 236–249.

[16] M. Brand, *Fast low-rank modifications of the thin singular value decomposition*, Linear Algebra Appl., 415 (2006), pp. 20–30, https://doi.org/10.1016/j.laa.2005.07.021.

[17] J. Calhoun, F. Cappello, L. N. Olson, M. Snir, and W. D. Gropp, *Exploring the feasibility of lossy compression for PDE simulations*, Internat. J. High Perform. Comput. Appl., 33 (2019), https://doi.org/10.1177/1094342018762036.

[18] S. Castruccio and M. G. Genton, *Compressing an ensemble with statistical models: An algorithm for global 3D spatio-temporal temperature*, Technometrics, 58 (2016), pp. 319–328.

[19] CERN, *Processing: What to Record*, Feb. 2019, https://home.cern/science/computing/processing-what-record.

[20] K. L. Clarkson and D. P. Woodruff, *Numerical linear algebra in the streaming model*, in Proceedings of the 41st ACM Symposium on Theory of Computing (STOC), Bethesda, MD, 2009, pp. 205–214.

[21] K. L. Clarkson and D. P. Woodruff, *Low rank approximation and regression in input sparsity time*, in Proceedings of the 45th ACM Symposium on Theory of Computing (STOC), ACM, New York, 2013, pp. 81–90, https://doi.org/10.1145/2488608.2488620.

[22] M. Cohen, *Nearly tight oblivious subspace embeddings by trace inequalities*, in Proceedings of the 27th ACM–SIAM Symposium on Discrete Algorithms (SODA), Arlington, VA, 2016, pp. 278–287, https://doi.org/10.1137/1.9781611974331.ch21.

[23] M. B. Cohen, S. Elder, C. Musco, C. Musco, and M. Persu, *Dimensionality reduction for k-means clustering and low rank approximation*, in Proceedings of the 47th ACM Symposium on Theory of Computing (STOC), ACM, New York, 2015, pp. 163–172.

[24] J. D. Dixon, *Estimating extremal eigenvalues and condition numbers of matrices*, SIAM J. Numer. Anal., 20 (1983), pp. 812–814, https://doi.org/10.1137/0720053.

[25] J. B. Drake, *Climate Modeling for Scientists and Engineers*, Math. Model. Comput. 19, SIAM, Philadelphia, 2014, https://doi.org/10.1137/1.9781611973549.

[26] D. Feldman, M. Volkov, and D. Rus, *Dimensionality reduction of massive sparse datasets using coresets*, in Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS), Barcelona, Spain, 2016, pp. 2774–2782.

[27] A. Frieze, R. Kannan, and S. Vempala, *Fast Monte-Carlo algorithms for finding low-rank approximations*, J. ACM, 51 (2004), pp. 1025–1041, https://doi.org/10.1145/1039488.1039494.

[28] E. Garnier, N. Adams, and P. Sagaut, *Large Eddy Simulation for Compressible Flows*, Springer, Cham, 2009.

[29] M. Ghashami, E. Liberty, J. M. Phillips, and D. P. Woodruff, *Frequent directions: Simple and deterministic matrix sketching*, SIAM J. Comput., 45 (2016), pp. 1762–1792, https://doi.org/10.1137/15M1009718.

[30] M. X. Goemans and D. Williamson, *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming*, J. Assoc. Comput. Mach., 42 (1995), pp. 1115–1145.

[31] S. Gratton and D. Titley-Peloquin, *Improved bounds for small-sample estimation*, SIAM J. Matrix Anal. Appl., 39 (2018), pp. 922–931, https://doi.org/10.1137/17M1137541.

[32] A. Griewank and A. Walther, *Algorithm 799: Revolve: An implementation of checkpointing for the reverse or adjoint mode of computational differentiation*, ACM Trans. Math. Softw., 26 (2000), pp. 19–45, https://doi.org/10.1145/347837.347846.

[33] J. Guinness and D. Hammerling, *Compression and conditional emulation of climate model output*, J. Amer. Stat. Assoc., 113 (2018), pp. 56–67.

[34] N. Halko, P.-G. Martinsson, Y. Shkolnisky, and M. Tygert, *An algorithm for the principal component analysis of large data sets*, SIAM J. Sci. Comput., 33 (2011), pp. 2580–2594, https://doi.org/10.1137/100804139.

[35] N. Halko, P. G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Rev., 53 (2011), pp. 217–288, https://doi.org/10.1137/090771806.

[36] N. J. Higham, *Matrix nearness problems and applications*, in Applications of Matrix Theory (Bradford, 1988), Oxford University Press, New York, 1989, pp. 1–27.

[37] M. Hinze, A. Walther, and J. Sternberg, *An optimal memory-reduced procedure for calculating adjoints of the instationary Navier-Stokes equations*, Optim. Control Appl. Methods, 27 (2006), pp. 19–40, https://doi.org/10.1002/oca.771.

[38] R. Horstmeyer, R. Y. Chen, X. Ou, B. Ames, J. A. Tropp, and C. Yang, *Solving ptychography with a convex relaxation*, New J. Phys., 17 (2015), 053044.

[39] M. F. Hutchinson, *A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines*, Comm. Statist. Simul. Comput., 18 (1989), pp. 1059–1076, https://doi.org/10.1080/03610918908812806.

[40] P. Indyk and R. Motwani, *Approximate nearest neighbors: Towards removing the curse of dimensionality*, in Proceedings of the 30th ACM Symposium on Theory of Computing (STOC), ACM, New York, 1998, pp. 604–613, https://doi.org/10.1145/276698.276876.

[41] W. B. Johnson and J. Lindenstrauss, *Extensions of Lipschitz mapping into Hilbert space*, Contemp. Math., 26 (1984), pp. 189–206.

[42] I. T. Jolliffe, *Principal Component Analysis*, 2nd ed., Springer Series in Statistics, Springer-Verlag, New York, 2002.

[43] Y. Li, H. L. Nguyen, and D. P. Woodruff, *Turnstile streaming algorithms might as well be linear sketches*, in Proceedings of the 46th ACM Symposium on Theory of Computing (STOC), ACM, New York, 2014, pp. 174–183.

[44] E. Liberty, *Accelerated Dense Random Projections*, Ph.D. thesis, Yale University, New Haven, CT, 2009.

[45] M. Lopes, S. Wang, and M. Mahoney, *Error estimation for randomized least-squares algorithms via the bootstrap*, in Proceedings of the 35th International Conference on Machine Learning (ICML), Stockholm, Sweden, 2018, pp. 3223–3232.

[46] M. W. Mahoney, *Randomized algorithms for matrices and data*, Found. Trends Mach. Learning, 3 (2011), pp. 123–224.

[47] M. R. Malik, B. J. Isaac, A. Coussement, P. J. Smith, and A. Parente, *Principal component analysis coupled with nonlinear regression for chemistry reduction*, Combustion and Flame, 187 (2018), pp. 30–41.

[48] P.-G. Martinsson, *Randomized Methods for Matrix Computations*, preprint, https://arxiv.org/abs/1607.01649v3, 2019.

[49] P.-G. Martinsson, V. Rokhlin, and M. Tygert, *A randomized algorithm for the decomposition of matrices*, Appl. Comput. Harmon. Anal., 30 (2011), pp. 47–68, https://doi.org/10.1016/j.acha.2010.02.003.

[50] X. Meng and M. W. Mahoney, *Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression*, in Proceedings of the 45th ACM Symposium on Theory of Computing (STOC), ACM, New York, 2013, pp. 91–100, https://doi.org/10.1145/2488608.2488621.

[51] F. R. Menter, M. Kuntz, and R. Langtry, *Ten years of industrial experience with the SST turbulence model*, Turbulence Heat Mass Transfer, 4 (2003), pp. 625–632.

[52] R. Moarref, A. S. Sharma, J. A. Tropp, and B. J. McKeon, *Model-based scaling of the streamwise energy intensity in high-Reynolds-number turbulent channels*, J. Fluid Mech., 734 (2013), pp. 275–316, https://doi.org/10.1017/jfm.2013.457.

[53] S. Muthukrishnan, *Data streams: Algorithms and applications*, Found. Trends Theor. Comput. Sci., 1 (2005), pp. 117–236.

[54] J. Nelson and H. L. Nguyen, *OSNAP: Faster numerical linear algebra algorithms via sparser subspace embeddings*, in Proceedings of the 54th IEEE Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society, Los Alamitos, CA, 2013, pp. 117–126, https://doi.org/10.1109/FOCS.2013.21.

[55] J. Nelson and H. L. Nguyen, *Lower bounds for oblivious subspace embeddings*, in Automata, Languages, and Programming. Part I, Lecture Notes in Comput. Sci. 8572, Springer, Heidelberg, 2014, pp. 883–894, https://doi.org/10.1007/978-3-662-43948-7_73.

[56] C. H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala, *Latent semantic indexing: A probabilistic analysis*, J. Comput. System Sci., 61 (2000), pp. 217–235, https://doi.org/10.1006/jcss.2000.1711.

[57] S. Patankar, *Numerical Heat Transfer and Fluid Flow*, CRC Press, Boca Raton, FL, 1980.

[58] R. W. Reynolds, N. A. Rayner, T. M. Smith, D. C. Stokes, and W. Wang, *An improved in situ and satellite SST analysis for climate*, J. Climate, 15 (2002), pp. 1609–1625.

[59] R. W. Reynolds, T. M. Smith, C. Liu, D. B. Chelton, K. S. Casey, and M. G. Schlax, *Daily high-resolution-blended analyses for sea surface temperature*, J. Climate, 20 (2007), pp. 5473–5496.

[60] V. Rokhlin, A. Szlam, and M. Tygert, *A randomized algorithm for principal component analysis*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 1100–1124, https://doi.org/10.1137/080736417.

[61] F. Roosta-Khorasani and U. Ascher, *Improved bounds on sample size for implicit matrix trace estimators*, Found. Comput. Math., 15 (2015), pp. 1187–1212, https://doi.org/10.1007/s10208-014-9220-1.

[62] P. Sagaut, *Large Eddy Simulation for Incompressible Flows: An Introduction*, Springer, Berlin, 2006.

[63] T. Sarlós, *Improved approximation algorithms for large matrices via random projections*, in Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS), Berkeley, CA, 2006, https://doi.org/10.1109/FOCS.2006.37.

[64] S. W. Son, Z. Chen, W. Hendrix, A. Agrawal, W.-k. Liao, and A. Choudhary, *Data compression for the exascale computing era—survey*, Supercomput. Frontiers Innovations, 1 (2014), pp. 76–88.

[65] Y. Sun, Y. Guo, J. A. Tropp, and M. Udell, *Low-Rank Tucker Approximation of a Tensor from Streaming Data*, manuscript, 2018.

[66] Y. Sun, Y. Guo, J. A. Tropp, and M. Udell, *Tensor random projection for low memory dimension reduction*, in NeurIPS Workshop on Relational Representation Learning, 2018, https://r2learning.github.io/assets/papers/CameraReadySubmission%2041.pdf.

[67] J. A. Tropp, *Improved analysis of the subsampled randomized Hadamard transform*, Adv. Adapt. Data Anal., 3 (2011), pp. 115–126, https://doi.org/10.1142/S1793536911000787.

[68] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher, *Fixed-rank approximation of a positive-semidefinite matrix from streaming data*, in Advances in Neural Information Processing Systems 30 (NIPS), Long Beach, CA, 2017.

[69] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher, *Practical sketching algorithms for low-rank matrix approximation*, SIAM J. Matrix Anal. Appl., 38 (2017), pp. 1454–1485, https://doi.org/10.1137/17M1111590.

[70] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher, *Randomized Single-View Algorithms for Low-Rank Matrix Approximation*, ACM Report 2017-01, Caltech, Pasadena, CA, 2017; preprint, https://arxiv.org/abs/1609.00048v1, 2016.

[71] J. Upadhyay, *Fast and Space-Optimal Low-Rank Factorization in the Streaming Model with Application in Differential Privacy*, preprint, https://arxiv.org/abs/1604.01429v1, 2016.

[72] J. Woodring, S. Mniszewski, C. Brislawn, D. DeMarle, and J. Ahrens, *Revisiting wavelet compression for large-scale climate data using JPEG* 2000 *and ensuring data precision*, in Proceedings of the 2011 IEEE Symposium on Large Data Analysis and Visualization (LDAV), IEEE, Washington, DC, 2011, pp. 31–38.

[73] D. P. Woodruff, *Sketching as a tool for numerical linear algebra*, Found. Trends Theor. Comput. Sci., 10 (2014).

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

[75] A. Yurtsever, M. Udell, J. A. Tropp, and V. Cevher, *Sketchy decisions: Convex low-rank matrix optimization with optimal storage*, in the 20th International Conference on Artificial Intelligence and Statistics (AISTATS), Fort Lauderdale, FL, 2017.

[76] G. Zhou, A. Cichocki, and S. Xie, *Decomposition of Big Tensors with Low Multilinear Rank*, preprint, https://arxiv.org/abs/1412.1885, 2014.

[77] R. Zimmermann, B. Pedersdorfer, and K. Willcox, *Geometric subspace updates with applications to online adaptive nonlinear model reduction*, SIAM J. Matrix Anal. Appl., 39 (2018), pp. 234–261, https://doi.org/10.1137/17M1123286.