MORE PRACTICAL SKETCHING ALGORITHMS FOR LOW-RANK MATRIX APPROXIMATION

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

Technical Report No. 2018-01
October 2018

APPLIED & COMPUTATIONAL MATHEMATICS
CALIFORNIA INSTITUTE OF TECHNOLOGY
mail code 9-94 · pasadena, ca 91125

MORE PRACTICAL SKETCHING ALGORITHMS FOR LOW-RANK MATRIX APPROXIMATION[*]

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

Abstract. This paper describes new algorithms for constructing a low-rank approximation of an input matrix from a *sketch*, a random low-dimensional linear image of the matrix. These algorithms come with rigorous performance guarantees. Empirically, the proposed methods achieve significantly smaller relative errors than other approaches that have appeared in the literature. For a concrete application, the paper outlines how the algorithms support on-the-fly compression of data from a direct Navier–Stokes (DNS) simulation.

Key words. Dimension reduction; matrix approximation; numerical linear algebra; randomized algorithm; single-pass algorithm; sketching; streaming algorithm; subspace embedding.

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

1. Motivation.

A *sketch* is a compressed data representation that supports updates to the underlying data and provides approximate answers to queries about the data. Over the last decade, sketches have emerged as a powerful tool for large-scale numerical linear algebra [51, 13, 25, 32, 50]. In particular, we can use a sketch to track a matrix that is presented as a sequence of linear updates, and we can extract a low-rank approximation of the induced matrix from the sketch. See [9, 21, 46, 45] for some recent work.

The purpose of this paper is to develop a new sketching method for low-rank matrix approximation in the streaming data model (section 2). We provide an informative mathematical analysis that explains the behavior of our algorithm (section 5). We also discuss implementation issues (section 4), and we present extensive numerical experiments on real and simulated data (section 6). The empirical performance of our technique is significantly better than earlier approaches (see subsection 6.1) that apply in the same setting.

Sketching methods for low-rank matrix approximation have many compelling applications. For instance, we have used these ideas to develop a storage-optimal algorithm for convex low-rank matrix optimization [52]. As a motivating example for this paper, we explain how sketching allows us to perform on-the-fly compression of data generated by large-scale computer simulations.

1.1. Vignette: On-the-Fly Compression for Simulation.

Computer simulations often produce data matrices that are too large to store, process, or transmit in full. This challenge arises in a wide range of areas, including weather and climate forecasting [49, 17, 4], heat transfer and fluid flow [40, 6], computational fluid dynamics [5, 20], and aircraft design [36, 42]. Nevertheless, in these settings, the data matrix

[*]First draft: 22 March 2017. First release: 16 July 2018.

Funding: JAT was supported in part by ONR Awards N00014-11-1002, N00014-17-1-214, N00014-17-1-2146, and the Gordon & Betty Moore Foundation. MU was supported in part by DARPA Award FA8750-17-2-0101. VC has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under the grant agreement number 725594 (time-data) and the Swiss National Science Foundation (SNSF) under the grant number 200021_178865.

[†]California Institute of Technology, Pasadena, CA (jtropp@cms.caltech.edu).

[‡]École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland (alp.yurtsever@epfl.ch, volkan.cevher@epfl.ch).

[§]Cornell University, Ithaca, NY (udell@cornell.edu).

often admits a good low-rank approximation. For many downstream applications, the low-rank approximation serves as well as—or even better then—the full data matrix because the approximation exposes latent structure [43, 11]. This observation raises the question of how to construct a low-rank approximation of simulation data efficiently.

We can model a simulation as a process that computes the state $\boldsymbol{a}_{t+1} \in \mathbb{R}^m$ of a system at time $t+1$ from the state $\boldsymbol{a}_t \in \mathbb{R}^m$ of the system at time t. The dimension m of the state increases with the resolution of the simulation. We may collect the data generated by the simulation into a matrix $\boldsymbol{A} = [\boldsymbol{a}_1, \dots, \boldsymbol{a}_n] \in \mathbb{R}^{m \times n}$.

The standard computational practice is to compute the full matrix $\boldsymbol{A}$ and then to compress it. Methods include direct computation of a low-rank matrix or tensor approximation [53, 3] or fitting a statistical model [12, 23, 33]. These approaches usually involve storage costs of $O(mn)$.

In contrast, we consider replacing these techniques by a sketching algorithm. As each new state is computed, we update the sketch to reflect the arrival of a new column $\boldsymbol{a}_t$ of the data matrix $\boldsymbol{A}$. Then we discard the state $\boldsymbol{a}_t$. Once the simulation is complete, we can extract a provably good rank-r approximation of $\boldsymbol{A}$ from the sketch. As we will see, this approach succeeds using total storage $O(r(m+n))$. For large matrices, the savings can be substantial. Subsection 6.7 contains a numerical demonstration of this idea.

1.2. Summary of Related Work.

Randomized algorithms for low-rank matrix approximation were proposed in the theoretical computer science (TCS) literature in the late 1990s [39, 19]. Soon after, numerical analysts developed practical versions of these algorithms [34, 51, 41, 25, 24]. For more background on the history of randomized linear algebra, see [25, 32, 50].

Sketching algorithms are specifically designed for the streaming model; that is, for data that is presented as a sequence of updates. The paper [51] contains the first algorithm for low-rank approximation that can operate in this setting. The first explicit treatment of numerical linear algebra in the streaming model appears in [13]. Recent papers on low-rank matrix approximation in the streaming model include [9, 21, 46, 45]. We refer the reader to the latter works for additional background and information. This paper also includes detailed citations throughout.

1.3. Notation.

We write $\mathbb{F}$ for the scalar field, which is either 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 operator $[\![\cdot]\!]_r$ returns a (simultaneous) best rank-r approximation of its argument with respect to the Schatten p-norms.

2. Sketching and Low-Rank Approximation of a Matrix.

In this section, we describe the basic procedure for sketching a matrix and for computing a low-rank approximation from the sketch. We postpone the discussion of implementation details and variants to section 4.

2.1. Dimension Reduction Maps.

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

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

In other words, the map reduces a vector of dimension n to dimension k, but it still preserves 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 several concrete examples.

Remark 2.1 (Geometry). The analysis of algorithms that use randomized dimension reduction often depends on more detailed properties than the embedding condition (2.1). See [25, 50] for more discussion.

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 specifies a value r for the target rank of the approximation. The target rank r is typically far smaller than the smaller dimension $\min\{m, n\}$ of the matrix.

2.3. The Sketch. Let us describe the sketching method we propose to acquire data about the input matrix. The sketch is parameterized by two natural numbers k, s that satisfy

$$r \le k \le s \le \min\{m, n\},$$

where r is the target rank. In subsection 5.6, we offer specific parameter recommendations that are supported by theoretical analysis. In subsection 6.5, we demonstrate that these parameter choices are effective in practice.

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

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

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 information about the co-range and the range of $\boldsymbol{A}$. The third matrix $(\boldsymbol{Z})$ contains information about the action of $\boldsymbol{A}$.

Remark 2.2 (Prior Work). The paper [48, Sec. 3] uses a sketch of the form (2.3) and (2.4) for low-rank matrix approximation. Related (but distinct) sketches appear in the papers [51, 13, 25, 50, 16, 10, 47, 46].

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

$$\boldsymbol{A} \leftarrow \theta\boldsymbol{A} + \tau\boldsymbol{H} \tag{2.5}$$

where $\theta, \tau \in \mathbb{F}$ 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

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

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 [30]. There are more restrictive streaming models (e.g., the columns of the matrix are presented in sequence) where it is possible to design other types of algorithms [18, 21].

2.5. Computing a Low-Rank Approximation. Once we have acquired a sketch $(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z})$ of an input matrix $\boldsymbol{A}$, our goal is to produce a low-rank approximation. Let us outline the computations we propose. The intuition appears below in subsection 2.6, and Section 5 presents the theoretical analysis.

The first two components $(\boldsymbol{X}, \boldsymbol{Y})$ of the sketch are used to estimate the co-range and the range of the matrix $\boldsymbol{A}$. Compute thin orthogonal–triangular 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; we discard the triangular parts $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$. The third sketch $\boldsymbol{Z}$ is used to compute the core matrix $\boldsymbol{W}$, which describes the predominant action of the matrix:

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

Last, we construct a rank-k approximation $\hat{\boldsymbol{A}}$ of the input matrix $\boldsymbol{A}$:

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

In some situations, it is more desirable to produce an approximation with exact rank r. To do so, we simply replace $\hat{\boldsymbol{A}}$ by its best rank-r approximation:

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

[The formula (2.10) is an easy consequence of the Eckart–Young Theorem [26, Sec. 6] and the fact that $\boldsymbol{Q}, \boldsymbol{P}$ have orthonormal columns.]

Remark 2.4 (Extensions). We can construct other structured approximations of $\boldsymbol{A}$ by projecting $\hat{\boldsymbol{A}}$ onto a set of structured matrices. See [46, Secs. 5–6] for a discussion of this idea in the context of another sketching technique. See our paper [45] for a sketching method designed for positive-semidefinite matrices.

Remark 2.5 (Prior Work). The reconstruction formulae (2.9) and (2.10) are new. The papers [51, 13, 25, 50, 16, 10, 48, 47, 46] describe alternative methods for low-rank matrix approximation from a sketch. The numerical work in section 6 demonstrates that the performance of our method is uniformly superior to the earlier techniques.

2.6. Intuition. The low-rank approximations (2.9) and (2.10) are based on some well-known insights from randomized linear algebra [25, Sec. 1]. Since $\boldsymbol{P}$ and $\boldsymbol{Q}$ capture the co-range 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 $\boldsymbol{A}$. Even so, we can estimate the core matrix using the action sketch $\boldsymbol{Z}$. Observe that

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

The approximation (2.11) allows us to drop the second term, so

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

Transfer the outer matrices to the left-hand side to discover that

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

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

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

When $\hat{\boldsymbol{A}}$ is a good approximation of $\boldsymbol{A}$, we can project it onto the set of rank-r matrices without increasing the error substantially:

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

Theorem 5.1 and Corollary 5.3 justify these heuristics completely for Gaussian dimension reduction maps.

Remark 2.6 (Prior Work). Our method is inspired by the intuition in [25, Sec. 1], which also motivates the low-rank sketching algorithms in [47, 46]. The sketching techniques in the TCS literature [13, 50, 16, 10, 48] are based on a different idea.

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 [31, 25, 50, 46] for additional discussion and examples.

3.1. Gaussian Maps.

The most basic dimension reduction map is simply a Gaussian matrix. That is, $\boldsymbol{\Xi} \in \mathbb{F}^{k \times n}$ is a $k \times n$ matrix with independent standard normal entries.[1]

Algorithm SM3.6 describes an implementation of Gaussian dimension reduction. The map $\boldsymbol{\Xi}$ requires storage of kn floating-point numbers in the field $\mathbb{F}$. The cost of applying the map to a vector is $O(kn)$ arithmetic operations.

Gaussian dimension reduction maps are simple, and they are effective in randomized algorithms for low-rank matrix approximation [25]. We can also analyze their behavior in full detail; see section 5. 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 for sketching applications.

Remark 3.1 (History). Gaussian dimension reduction has been used as an algorithmic tool since the paper of Indyk & Motwani [28]. In spirit, this approach is quite similar to the earlier theoretical work of Johnson & Lindenstrauss [29], 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

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

The matrices $\boldsymbol{\Pi}, \boldsymbol{\Pi}' \in \mathbb{F}^{n \times n}$ are signed permutations,[2] drawn independently and uniformly at random. The matrix $\boldsymbol{F} \in \mathbb{F}^{n \times n}$ denotes a discrete cosine transform

[1]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.

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

($\mathbb{F} = \mathbb{R}$) or a discrete Fourier transform ($\mathbb{F} = \mathbb{C}$). The matrix $\boldsymbol{R} \in \mathbb{F}^{k \times n}$ is a restriction to k coordinates, chosen uniformly at random.

Algorithm SM3.7 presents an implementation of an SSRFT. The cost of storing $\boldsymbol{\Xi}$ is just $O(n)$ numbers. The cost of applying $\boldsymbol{\Xi}$ to a vector is $O(n \log n)$ arithmetic operations, using the Fast Fourier Transform (FFT) or the Fast Cosine Transform (FCT). This cost can be reduced [51] further to $O(n \log k)$, but the improvement is rarely worth the implementation effort.

In practice, SSRFTs behave almost the same way as Gaussian matrices, but their storage cost does not scale with the output dimension k. On the other hand, the analysis [2, 44, 8] is less complete than in the Gaussian case [25]. A proper implementation requires fast trigonometric transforms. Last, the random permutations and FFTs require data movement, which could be a challenge in the distributed setting.

Remark 3.2 (History). SSRFTs are inspired by the work of Ailon & Chazelle [2] on fast Johnson–Lindstrauss transforms. For applications in randomized linear algebra, see the papers [51, 31, 25, 44, 8].

3.3. Sparse Sign Matrices. Last, we describe another type of randomized dimension reduction map, called a *sparse sign matrix*. We recommend these maps for practical implementations where data movement (i.e., coherency) is a concern.

To construct a sparse sign matrix $\boldsymbol{\Xi} \in \mathbb{F}^{k \times n}$, we fix a sparsity parameter ζ in the range $2 \leq \zeta \leq k$. The columns of the matrix are drawn independently at random. To construct each column, we take ζ iid 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 $\zeta = \min\{k, 2\log(1+n)\}$ is an effective parameter selection. See [15] for some theoretical justification.

Algorithm SM3.8 describes an implementation of sparse dimension reduction. Since the matrix $\boldsymbol{\Xi} \in \mathbb{F}^{k \times n}$ has ζ nonzeros per column, we can store the matrix with $O(\zeta n \log(1 + k/\zeta))$ numbers. The cost of applying the map to a vector is $O(\zeta n)$ arithmetic operations.

Sparse sign matrices have benefits for data coherency because the columns are generated independently and the matrices can be applied using (blocked) matrix multiplication. One weakness is that we must use sparse data structures and arithmetic to enjoy the benefit of these maps.

Remark 3.3 (History). Sparse dimension reduction maps are inspired by the work of Achlioptas [1] on database-friendly random projections. For applications in randomized linear algebra, see [14, 35, 37, 38, 7, 15].

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. All pseudocode appears in section SM2. The supplementary materials include MATLAB code for the algorithms.

4.1. Sketching and Updates. Algorithms SM3.1 and SM3.2 contain the pseudocode for initializing the sketch and for performing the linear update (2.5).

The sketch requires the 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 $\boldsymbol{A} \leftarrow \theta \boldsymbol{A} + \tau \boldsymbol{H}$ is dominated by the minimum cost of computing $\boldsymbol{\Phi H}$ or $\boldsymbol{H\Psi}$. That is, we apply the dimension reduction map to s vectors of length $\min\{m, n\}$. The cost of the update depends heavily on the structure of the matrix $\boldsymbol{H}$ and the type of dimension reduction map.

4.2. Low-Rank Approximation. Algorithm SM3.3 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 k^2 numbers for the core matrix $\boldsymbol{W}$. The arithmetic cost is usually dominated by the computation of the orthogonal–triangular factorizations of $\boldsymbol{X}^*$ and $\boldsymbol{Y}$, which require $O(k^2(m+n))$ operations. When the parameters satisfy $s \gg k$, it is possible that the cost $O(ks^2)$ of forming the core matrix $\boldsymbol{W}$ will be larger.

4.3. Fixed-Rank Approximation. Algorithm SM3.4 presents the pseudocode for computing the rank-r approximation $[\![\hat{\boldsymbol{A}}]\!]_r$ of the matrix $\boldsymbol{A}$ contained in the sketch; see (2.10).

The working storage cost $O(k(m+n))$ is dominated by the call to the routine Algorithm SM3.3. Typically, the arithmetic cost is also dominated by the $O(k^2(m+n))$ cost of the call to Algorithm SM3.3. When $s \gg k$, it is possible that the $O(s^3)$ cost of the truncated SVD will drive the arithmetic cost.

5. Theoretical Results.

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

In this section, we analyze our sketching and 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 the identical performance (subsection 6.4), so our analysis also supports more practical implementations based on SSRFTs or sparse sign matrices. The numerical work in section 6 confirms the value of this analysis.

5.1. The Tail Energy. For each natural number r, define the rth *tail energy* of the input matrix

$$\tau_r^2(\boldsymbol{A}) := \min_{\operatorname{rank} \boldsymbol{B} < r} \|\boldsymbol{A} - \boldsymbol{B}\|_2^2 = \sum_{j \geq r} \sigma_j^2(\boldsymbol{A}),$$

where σ_j returns the jth largest singular value of a matrix. The second identity follows from the Eckart–Young Theorem [26, Sec. 6].

5.2. The Field Parameter. We also introduce a parameter that reflects the field over which we are working:

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

This quantity allows us to capture the behavior of real and complex Gaussian matrices within the same formula.

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

Theorem 5.1 (Low-Rank Approximation: Error Bound). *Let $\boldsymbol{A} \in \mathbb{F}^{m\times n}$ be an arbitrary input matrix. Assume that 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 [46, 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 is the target rank of the approximation. Then the choice

$$k = 5r + \alpha \quad \text{and} \quad s = 2k + \alpha \tag{5.3}$$

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

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

In practice, we have found the parameter selection (5.3) to be effective for a range of examples. Moreover, if $k/r \to \infty$ and $s/k \to \infty$, we drive the leading constant in (5.2) to one.

The true meaning of Theorem 5.1 is more subtle. The minimum over ϱ indicates that the approximation automatically adapts to the spectral decay of the input matrix. This effect is usually more significant than any benefit we may achieve by adjusting the parameters to control the leading constant. In subsection 5.6, we exploit this idea to recommend sketch size parameters for a given storage budget.

Remark 5.2 (Failure probability). It is well known that the expected performance of randomized linear algebra methods also characterizes the typical performance [25, Fig. 7.3]. The probability that the error is significantly larger than (5.2) is negligible.

5.4. Analysis of Fixed-Rank Approximation. Our second result gives a bound for the error in the rank-r approximation $[\![\hat{\boldsymbol{A}}]\!]_r$ of the input matrix $\boldsymbol{A}$.

Corollary 5.3 (Fixed-Rank 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 the result [46, Prop. 6.1]. We omit the details.

Let us elaborate on Corollary 5.3. When the approximation $\hat{\boldsymbol{A}}$ is a good rank-k approximation of $\boldsymbol{A}$, then the matrix $[\![\hat{\boldsymbol{A}}]\!]_r$ is also a good rank-r approximation of $\boldsymbol{A}$. In particular, the rank-r approximation can exploit decay in the spectrum of the input matrix. The empirical work in section 6 highlights the practical importance of this phenomenon.

5.5. The Storage Budget. It is important to understand the storage we need to maintain a sketch of an input matrix. 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 does not increase with the sketch size parameters (k, s). In this case, 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 sketch matrices require a total storage budget of

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

floating-point numbers in the field $\mathbb{F}$. To achieve a rank-r approximation, the minimum allowable values for the sketch size parameters are $k_{\min} = r + \alpha + 1$ and $s_{\min} = 2k_{\min} + \alpha$. Therefore, the minimum storage budget is

$$T_{\min}(r) := (r + \alpha + 1)(m+n) + (2r + 3\alpha + 1)^2.$$

Of course, larger parameters (k, s) support better approximations. In the next section, we offer a more practical approach for choosing (k, s).

5.6. Theoretical Guidance for Sketch Size Parameters. Suppose that we fix the storage budget T, defined in (5.4). We may ask how to apportion the sketch size parameters (k, s) to achieve superior empirical performance. Theorem 5.1 offers insight on this question; see subsection 6.5 for numerical support.

5.6.1. General Spectrum. To control the theoretical bound Theorem 5.1 on the approximation error, it is natural to make the parameter k as large as possible. Indeed, when k is large, the parameter ϱ in the error bound (5.2) has more room to adapt to decay in the spectrum of $\boldsymbol{A}$. Note that the condition $s \geq 2k + \alpha$ ensures that the first fraction in the error bound cannot exceed two.

Therefore, for $T \geq T_{\min}(r)$, we pose the optimization problem

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

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.

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

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

$$\begin{aligned} \min \frac{s-\alpha}{s-k-\alpha} \cdot \frac{k+\hat{r}-\alpha}{k-\hat{r}-\alpha} \quad \text{subject to} \quad & s \geq 2k+\alpha, \quad k \geq \hat{r} + \alpha + 1, \\ & \text{and} \quad k(m+n) + s^2 = T. \end{aligned} \tag{5.7}$$

In fact, this problem admits a closed-form solution, but we have chosen to omit the complicated formula.

6. 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, and we argue that the proposed approach produces superior results.

6.1. Alternative Sketching and Reconstruction Methods.

We compare our approach with two sketching algorithms for low-rank matrix approximation that have appeared in the literature. Our recent work [46] identifies these two algorithms as the best techniques available, so we omit comparisons with additional methods.

6.1.1. A Three-Sketch Method.

Boutsidis et al. [10, Sec. 6] recently introduced a new method for low-rank matrix approximation from a sketch; Upadhyay [48, Sec. 3] later proposed some refinements.

Upadhyay's variant is based on the same kind of sketch (2.2)–(2.4) that we are using in this paper. He develops the following formula for approximating the input matrix. First, compute orthonormal bases $\boldsymbol{Q}$ and $\boldsymbol{P}$ for the range and co-range via (2.7). Then form thin singular value decompositions:

$$\boldsymbol{\Phi Q} = \boldsymbol{U}_1 \boldsymbol{S}_1 \boldsymbol{V}_1^* \quad \text{and} \quad \boldsymbol{\Psi P} = \boldsymbol{U}_2 \boldsymbol{S}_2 \boldsymbol{V}_2^*.$$

Construct the rank-r approximation

$$\hat{\boldsymbol{A}}_{\text{upa}} = \boldsymbol{Q}\boldsymbol{V}_1 \boldsymbol{S}_1^\dagger \, [\![\boldsymbol{U}_1^* \boldsymbol{Z} \boldsymbol{U}_2]\!]_r \, \boldsymbol{S}_2^\dagger \boldsymbol{V}_2^* \boldsymbol{P}^*. \tag{6.1}$$

Superficially, the approximation $\hat{\boldsymbol{A}}_{\text{upa}}$ may look similar to the approximation we developed in (2.10). Nevertheless, they are designed using different principles, and their performance is quite different in practice.

6.1.2. A Two-Sketch Method.

In [46], we developed and analyzed a very simple sketching algorithm for low-rank matrix approximation. This approach uses only two dimension reduction maps:

$$\boldsymbol{\Upsilon} \in \mathbb{F}^{\ell \times m} \quad \text{and} \quad \boldsymbol{\Omega} \in \mathbb{F}^{k \times n} \quad \text{where } k \le \ell.$$

The sketch takes the form

$$\boldsymbol{X} = \boldsymbol{\Upsilon A} \quad \text{and} \quad \boldsymbol{Y} = \boldsymbol{A \Omega}^*.$$

To obtain a rank-r approximation from this sketch, we compute

$$\boldsymbol{Y} = \boldsymbol{QR} \quad \text{and} \quad \hat{\boldsymbol{A}}_{\text{two}} = \boldsymbol{Q}[\![(\boldsymbol{\Upsilon Q})^\dagger \boldsymbol{X}]\!]_r. \tag{6.2}$$

The numerically stable implementation is a little more complicated; see [46, Alg. 7] for details.

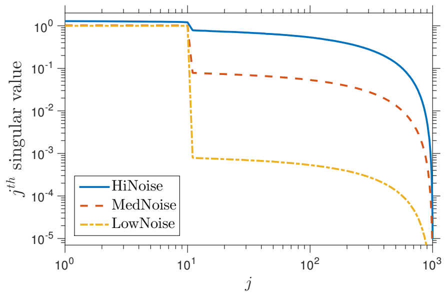
6.2. Experimental Setup.

Our experimental design is quite similar to our previous papers [46, 45] on sketching algorithms for low-rank matrix approximation.

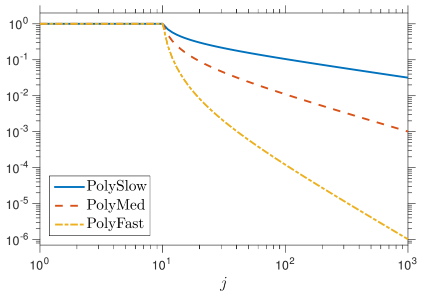
6.2.1. Procedure.

Fix an input matrix $\boldsymbol{A} \in \mathbb{F}^{n \times n}$ and a target rank r. Then select the sketch size parameters (k, s) or (k, ℓ). For each trial, we draw dimension reduction maps from a specified distribution and form the sketch of the input matrix. We compute a rank-r approximation $\hat{\boldsymbol{A}}_{\text{out}}$ using a specified reconstruction formula. The approximation error is calculated relative to the best rank-r approximation error in Schatten p-norm:

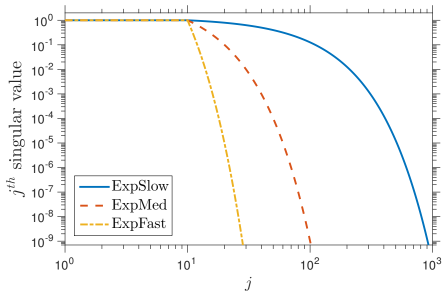
$$S_p \text{ relative error} \quad = \quad \frac{\|\boldsymbol{A} - \hat{\boldsymbol{A}}_{\text{out}}\|_p}{\|\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r\|_p} - 1. \tag{6.3}$$



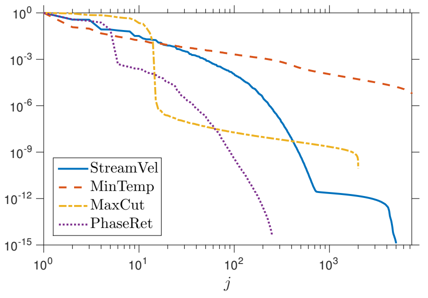
(A) Low-Rank + Noise



(B) Polynomial Decay



(C) Exponential Decay



(D) Data Matrices

Fig. 1: **Spectra of input matrices.** Plots of the singular value spectrum for an input matrix from each of the synthetic classes (`LowRank`, `PolyDecay`, `ExpDecay` with effective rank $R = 10$) and from each of the real data classes (`MinTemp`, `StreamVel`, `MaxCut`, `PhaseRetrieval`) described in subsection 6.3.

We perform 20 independent trials and report the average error. Owing the measure concentration effects, the average error is also the typical error; cf. [25, Fig. 7.3].

The body of this paper presents a limited selection of results. Section SM3 contains additional numerical evidence. The supplementary materials also include MATLAB code that can reproduce these experiments.

6.2.2. The Oracle Error. To make fair comparisons among algorithms, we fix the storage budget and identify the parameter choices that minimize the relative error (6.3) incurred. We refer to the minimum as the *oracle error* for an algorithm.

For our new reconstruction (2.10) and for the Upadhyay method (6.1), we compute the total storage cost as $T = k(m+n)+s^2$ and we require that $k > r+\alpha$ and $s \geq 2k+\alpha$. For the two-sketch method (6.2), the total storage cost is $T = km+\ell n$ and we require that $k > r+\alpha$ and $\ell > k+\alpha$. Note that 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.

6.3. Classes of Input Matrices. As in our previous papers [46, 45], we consider several different types of synthetic and real input matrices. See Figure 1 for a plot of the spectra of these input matrices.

6.3.1. Synthetic Examples. We work over the complex field $\mathbb{C}$. The matrix dimensions $m = n = 10^3$, and we introduce an effective rank parameter $R \in \{5, 10, 20\}$.

We compute an approximation with actual rank $r = 10$.

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

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

   where $\boldsymbol{W} = \boldsymbol{G}\boldsymbol{G}^*$ 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` ($q = 0.5$).

6.3.2. Application Examples. We also consider instances of low-rank data matrices that arise in applications. For these matrices, we consider a range of values for the actual rank r of the approximation.

1. **Navier–Stokes:** We test the hypothesis, discussed in subsection 1.1, that sketching methods can be used to perform on-the-fly compression of the output of a PDE simulation. We have obtained a 2D Direct Navier–Stokes (DNS) simulation of low-Reynolds number flow around a cylinder on a coarse mesh. 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. This is typical for physical models.

2. **Weather:** We also 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. This is typical for measured data.

3. **Sketchy Decisions:** Last, we consider matrices that arise from an optimization algorithm for solving large-scale semidefinite programs [52]. 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 psd matrix with $m = n = 2,000$ that gives a high-accuracy solution to the MaxCut SDP for a sparse graph [22]. This matrix is effectively rank deficient with $R = 14$, and the spectrum has fast exponentially 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 [27]. This matrix is effectively rank deficient with $R = 5$, and the spectrum has fast exponential decay after this point.

6.4. Insensitivity to Dimension Reduction Map. Our first experiment is designed to show that the proposed rank-r reconstruction formula (2.10) is insensitive to the distribution of the dimension reduction map at the oracle parameter values (subsection 6.2.2).

Figure 2 presents experiments with synthetic test matrices with effective rank $R = 10$, approximation rank $r = 10$, and the Schatten 2-norm error (6.3). For most storage budgets T, the Gaussian, SSRFT, and sparse dimension reduction maps yield equivalent values for the oracle error. In fact, because it is unitary, the SSRFT map even performs slightly *better* than the others when the storage budget is very large. See subsection SM3.1 for more numerics, which transmit the same message.

The other reconstruction methods (6.1) and (6.2) are also insensitive to the choice of dimension reduction maps. We omit the numerical evidence. These observations justify the transfer of theoretical and empirical results for Gaussians to SSRFT and sparse dimension reduction maps.

6.5. Approaching the Oracle Performance. Next, we show that theoretical parameter choices in (2.10) produce results almost as good as the oracle performance.

Figures 3 and 4 display the outcome of the following experiment. For synthetic test matrices with effective rank $R = 10$ and approximation rank $r = 10$, we compare the oracle performance (subsection 6.2.2) of our rank-r approximation (2.10) with its performance at the theoretical parameters proposed in subsection 5.6. (In the formula (5.7) for a flat spectrum, we set the tail location $\hat{r} = r$.) We use Gaussian dimension reduction maps, but equivalent results hold for other types of dimension reduction maps. See subsection SM3.2 for effective rank $R = 5$ and $R = 20$.

For most of the examples, the general parameter choice (5.6) is able to deliver a relative error that tracks the oracle error closely. The parameter choice (5.7) for a flat spectrum works somewhat better for matrices whose spectral tail exhibits slow decay (`LowRankLowNoise`, `LowRankMedNoise`, `LowRankHiNoise`). We also learn that the theoretical formulas are not entirely reliable when the storage budget is very small. Matrices with a lot of tail energy (`LowRankHiNoise`, `PolyDecaySlow`) are very hard to approximate accurately with a sketching algorithm, so it is not surprising that our theoretical parameter choices fall short of the oracle parameters in these cases.

6.6. Comparison of Reconstruction Formulas: Synthetic Examples. Let us now compare the proposed rank-r reconstruction formula (2.10) with the Upadhyay approximation (6.1) and the two-sketch approximation (6.2).

Figures 5 and 6 present the results of the following experiment. For synthetic matrices with effective rank $R = 10$ and approximation rank $r = 10$, we compare the relative error (6.3) achieved by each of the three rank-r reconstructions as a function of storage (subsection 6.2.2). We use Gaussian dimension reduction maps in these experiments; similar results are evident for other types of maps. Results for effective rank $R = 5$ and $R = 20$ appear in subsection SM3.3.

Let us make some remarks:

- This experiment demonstrates clearly that the proposed approximation (2.10) dominates the earlier methods for all 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 the Upadhyay method (6.1).
- For input matrices whose spectral tail decays more quickly (`ExpDecaySlow`, `ExpDecayMed`, `ExpDecayFast`, `PolyDecayMed`, `PolyDecayFast`), the proposed method improves massively over Upadhyay (6.1).
- The new method (2.10) shows its strength over the two-sketch method (6.2) when the storage budget is small. It also yields superior performance in Schatten ∞-norm. These differences are most evident for matrices with slow spectral decay.

In summary, the proposed method (2.10) enjoys the advantages of the Upadhyay (6.1) method and our previous approach (6.2), with no evident disadvantages.

6.7. Comparison of Reconstruction Formulas: Real Data Examples. Our last set of experiments is designed to show that our sketching and reconstruction pipeline is effective for real data.

Figures 7 and 8 contains the results of the following experiment. For each of the three rank-r reconstruction methods, we display the relative error (6.3) as a function of storage. We use sparse dimension reduction maps, which is justified by the experiments in subsection 6.4.

We plot the oracle error (subsection 6.2.2) attained by each method. Since the oracle error is not achievable in practice, we also chart the performance of each method at an *a priori* selection of parameters. For the proposed method (2.10), we use the natural parameter choice (5.6) that follows from our theoretical analysis. The Upadhyay sketch takes the same form as ours but lacks a comparable theory, so we instantiate his method with the same parameters (5.6) we used in our sketch. Last, for the two-sketch method (6.2), we assume that the input matrix $\boldsymbol{A} \in \mathbb{F}^{m\times n}$ is tall ($m \geq n$), and we use the theoretically motivated parameter values

$$k = \max\{r + \alpha + 1, \lfloor (T - n\alpha)/(m + 2n)\rfloor\} \quad \text{and} \quad \ell = \lfloor (T - km)/n\rfloor.$$

This choice adapts the arguments in [46, Sec. 4.5.2] to use the current definition of the storage budget T.

As with the synthetic examples, the proposed method (2.10) dominates the competing methods for all the examples we considered. This is true when we compare oracle errors or when we compare the errors using *a priori* 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 Upadhyay (6.1) method cannot achieve a small relative error (6.3), even with a large amount of storage. The two-sketch method (6.2) can achieve small relative error, but it often requires more storage to achieve this goal—especially at the *a priori* parameter choices.

6.8. Example: Flow-Field Reconstruction. Finally, we elaborate on using sketching to compress the DNS data matrix `StreamVel`. We compute the best rank-10 approximation of the matrix via (2.10) using storage $T/(m+n) = 48$ and the parameter choices (5.6). For this example, we can use plots of the flow field to make visual comparisons.

Figure 9 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 errors appear, especially at the inlet (on the left-hand side of the images). See subsection SM3.4 for additional numerics.

If we normalize `StreamVel` so that its largest singular value equals one, then the best rank-10 approximation of `StreamVel` has absolute S_∞ error $2.223 \cdot 10^{-2}$. Meanwhile, the computed rank-10 approximation has absolute S_∞ error $2.226 \cdot 10^{-2}$. (The S_∞ relative error (6.3) is $1.3 \cdot 10^{-3}$.) We can easily improve these numbers by computing a higher-rank approximation and/or increasing the storage budget.

We learn that the sketched matrix supports an excellent rank-10 reconstruction, even though it only uses 5.8 MB of storage in double precision. For comparison, the full matrix requires 409.7 MB of storage. The compression rate is $70.6\times$. This demonstration suggests that it is indeed possible to automatically compress the output of the DNS simulation using sketching.

7. 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). We have described how to implement the method using SSRFTs or sparse dimension reduction methods (section 3), and we have argued that the performance of the method is insensitive to the choice of dimension reduction map (subsection 6.4). In addition, a detailed theoretical analysis (section 5) prescribes how to select parameter values for the sketch *a priori*, and we have shown that these parameter values yield good performance across a range of examples (subsection 6.5). Finally, we have demonstrated that the new reconstruction method dominates existing techniques for both synthetic matrices (subsection 6.6) and real data (subsection 6.7).

A potential application of these techniques is for on-the-fly-compression of data from large-scale simulations. Our work with DNS data indicates that we can achieve significant data reduction A key advantage of our new approach over (6.2) is that it extends to higher-dimensional (i.e., tensor) data. This generalization should allow for higher compression rates, and we plan to explore this idea in a future work.

Acknowledgments. The authors wish to thank Beverley McKeon and Sean Symon for providing DNS simulation data and visualization software. William North contributed the weather data.

REFERENCES

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

[2] 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, http://dx.doi.org/10.1137/060673096.

[3] W. Austin, G. Ballard, and T. G. Kolda, *Parallel tensor compression for large-scale scientific data*, in 2016 IEEE Intl. Symp. Parallel and Distributed Processing, 2016, pp. 912–922.

[4] 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 Proc. 23rd ACM Intl. Symp. High-Performance Parallel and Distributed Computing, 2014, pp. 203–214.

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

[6] A. Bejan, *Convection heat transfer*, John Wiley & Sons, 2013.

[7] 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, http://dx.doi.org/10.1007/s00039-015-0332-9.

[8] 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, http://dx.doi.org/10.1137/120874540.

[9] C. Boutsidis, D. Woodruff, and P. Zhong, *Optimal principal component analysis in distributed and streaming models.* Available at http://arXiv.org/abs/1504.06729, July 2016.

[10] C. Boutsidis, D. Woodruff, and P. Zhong, *Optimal principal component analysis in distributed and streaming models*, in Proc. 48th ACM Symp. Theory of Computing (STOC 2016), Cambridge, MA, 2016.

[11] J. Calhoun, F. Cappello, L. N. Olson, M. Snir, and W. D. Gropp, *Exploring the feasibility of lossy compression for PDE simulations*, The International Journal of High Performance Computing Applications, (2018), p. 1094342018762036, https://doi.org/10.1177/1094342018762036.

[12] 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.

[13] K. L. Clarkson and D. P. Woodruff, *Numerical linear algebra in the streaming model*, in Proc. 41st ACM Symp. Theory of Computing (STOC), Bethesda, 2009.

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

[15] M. Cohen, *Nearly tight oblivious subspace embeddings by trace inequalities*, in Proc. 27th ACM-SIAM Symp. Discrete Algorithms (SODA), Arlington, Jan. 2016, pp. 278–287.

[16] M. B. Cohen, S. Elder, C. Musco, C. Musco, and M. Persu, *Dimensionality reduction for k-means clustering and low rank approximation*, in Proc. 47th ACM Symp. Theory of Computing (STOC), ACM, 2015, pp. 163–172.

[17] J. B. Drake, *Climate modeling for scientists and engineers*, SIAM, 2014.

[18] D. Feldman, M. Volkov, and D. Rus, *Dimensionality reduction of massive sparse datasets using coresets*, in Adv. Neural Information Processing Systems 29 (NIPS), 2016.

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

[20] E. Garnier, N. Adams, and P. Sagaut, *Large eddy simulation for compressible flows*, Springer Science & Business Media, 2009.

[21] 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, https://doi-org.clsproxy.library.caltech.edu/10.1137/15M1009718.

[22] 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.

[23] J. Guinness and D. Hammerling, *Compression and conditional emulation of climate model output*, J. Amer. Stat. Assoc., (2017).

[24] 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, https://doi-org.clsproxy.library.caltech.edu/10.1137/100804139.

[25] 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.

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

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

[28] P. Indyk and R. Motwani, *Approximate nearest neighbors: Towards removing the curse of dimensionality*, in Proc. 30th ACM Symp. Theory of Computing (STOC), STOC '98, New York, NY, USA, 1998, ACM, pp. 604–613, https://doi.org/10.1145/276698.276876, http://doi.acm.org/10.1145/276698.276876.

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

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

[31] E. Liberty, *Accelerated dense random projections*, PhD thesis, Yale Univ., New Haven, 2009.

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

[33] 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.

[34] 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, http://dx.doi.org/10.1016/j.acha.2010.02.003.

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

[36] F. R. Menter, M. Kuntz, and R. Langtry, *Ten years of industrial experience with the sst turbulence model*, Turbulence, heat and mass transfer, 4 (2003), pp. 625–632.

[37] J. Nelson and H. L. Nguyen, *OSNAP: faster numerical linear algebra algorithms via sparser subspace embeddings*, in 2013 IEEE 54th Symp. Foundations of Computer Science (FOCS), IEEE Computer Soc., Los Alamitos, CA, 2013, pp. 117–126, https://doi.org/10.1109/FOCS.2013.21, http://dx.doi.org/10.1109/FOCS.2013.21.

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

[39] 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, http://dx.doi.org/10.1006/jcss.2000.1711. Special issue on the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (Seattle, WA, 1998).

[40] S. Patankar, *Numerical heat transfer and fluid flow*, CRC press, 1980.

[41] 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, https://doi-org.clsproxy.library.caltech.edu/10.1137/080736417.

[42] P. Sagaut, *Large eddy simulation for incompressible flows: an introduction*, Springer Science & Business Media, 2006.

[43] S. W. Son, Z. Chen, W. Hendrix, A. Agrawal, W.-k. Liao, and A. Choudhary, *Data compression for the exascale computing era-survey*, Supercomputing frontiers and innovations, 1 (2014), pp. 76–88.

[44] 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, http://dx.doi.org/10.1142/S1793536911000787.

[45] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher, *Fixed-rank approximation of a positive-semidefinite matrix from streaming data*, in Adv. Neural Information Processing

Systems 30 (NIPS), Long Beach, Dec. 2017.

[46] 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.

[47] 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, Jan. 2017. Available at http://arXiv.org/abs/1609.00048, v1.

[48] J. Upadhyay, *Fast and space-optimal low-rank factorization in the streaming model with application in differential privacy*. Available at http://arXiv.org/abs/1604.01429, Apr. 2016.

[49] 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 2011 IEEE Symp. Large Data Analysis and Visualization (LDAV), 2011, pp. 31–38.

[50] D. P. Woodruff, *Sketching as a tool for numerical linear algebra*, Found. Trends Theor. Comput. Sci., 10 (2014), pp. iv+157.

[51] 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.

[52] A. Yurtsever, M. Udell, J. A. Tropp, and V. Cevher, *Sketchy decisions: Convex low-rank matrix optimization with optimal storage*, in 2017 Intl. Conf. Artificial Intelligence and Statistics (AISTATS), 2017.

[53] G. Zhou, A. Cichocki, and S. Xie, *Decomposition of big tensors with low multilinear rank*, arXiv preprint arXiv:1412.1885, (2014).

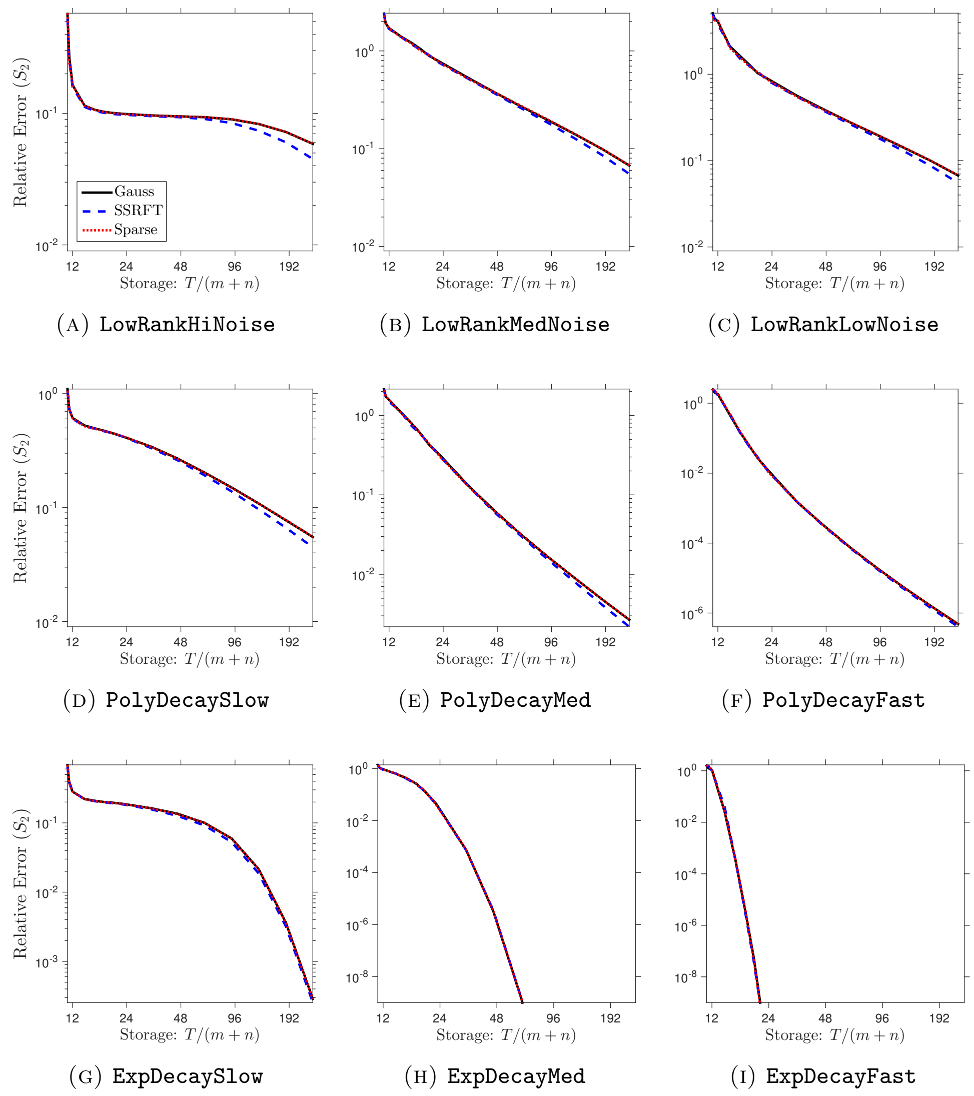
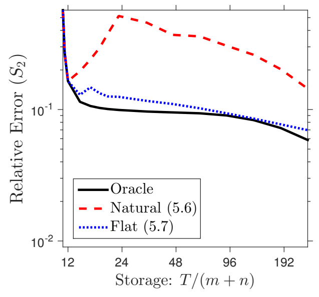
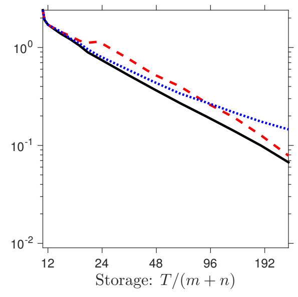


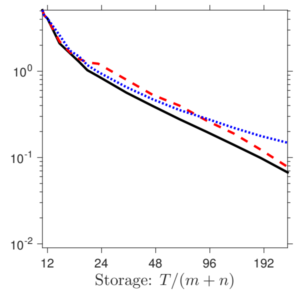
FIG. 2: **Insensitivity of proposed method to the dimension reduction map.** (Effective rank $R = 10$, approximation rank $r = 10$, Schatten 2-norm.) We compare the oracle performance of the proposed fixed-rank approximation (2.10) implemented with Gaussian, SSRFT, or sparse dimension reduction maps. See subsection 6.4 for details.



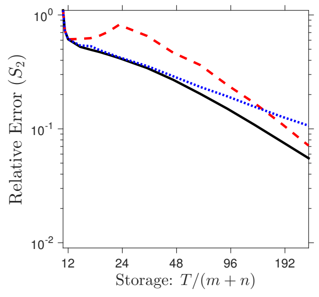
(A) LowRankHiNoise



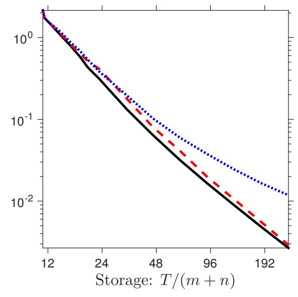
(B) LowRankMedNoise



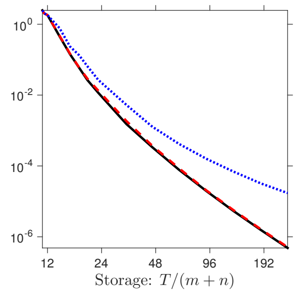
(C) LowRankLowNoise



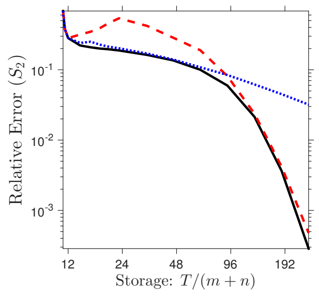
(D) PolyDecaySlow



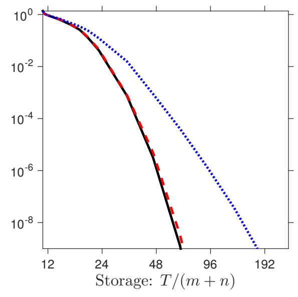
(E) PolyDecayMed



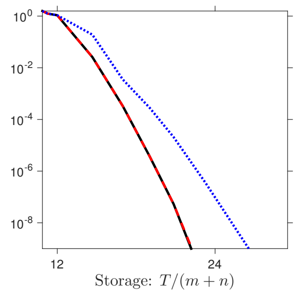
(F) PolyDecayFast



(G) ExpDecaySlow

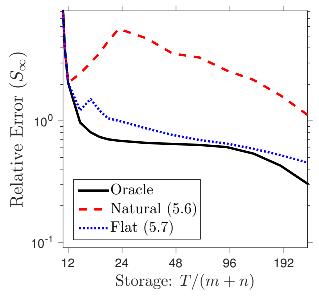


(H) ExpDecayMed

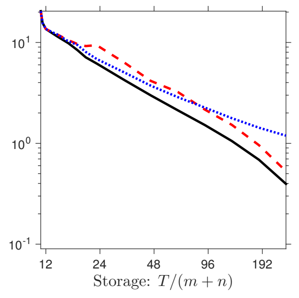


(I) ExpDecayFast

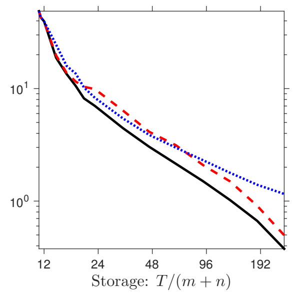
FIG. 3: **Relative error for proposed method with *a priori* parameters.** (Gaussian maps, effective rank $R = 10$, approximation rank $r = 10$, **Schatten 2-norm**.) We compare the oracle performance of the proposed fixed-rank approximation (2.10) with its performance at theoretically justified parameter values. See subsection 6.5 for details.



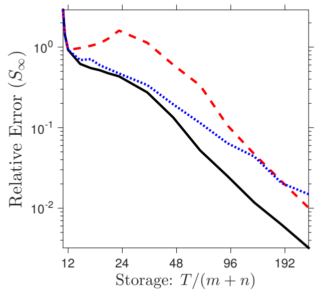
(A) `LowRankHiNoise`



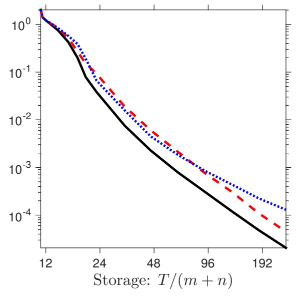
(B) `LowRankMedNoise`



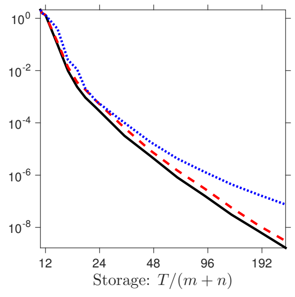
(C) `LowRankLowNoise`



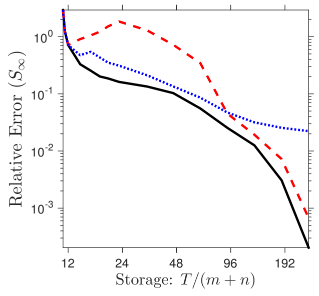
(D) `PolyDecaySlow`



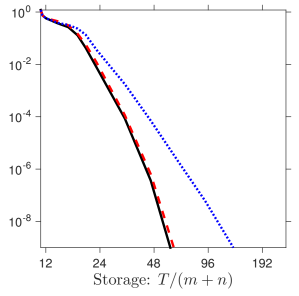
(E) `PolyDecayMed`



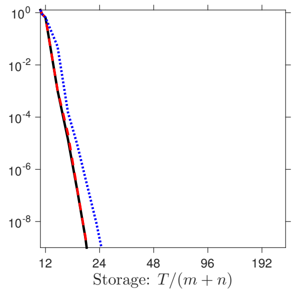
(F) `PolyDecayFast`



(G) `ExpDecaySlow`

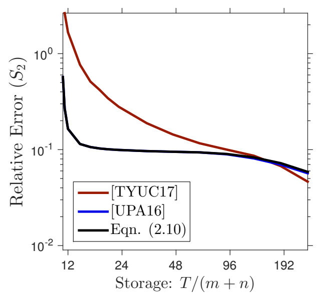


(H) `ExpDecayMed`

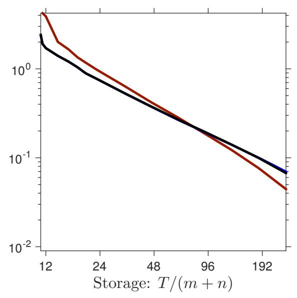


(I) `ExpDecayFast`

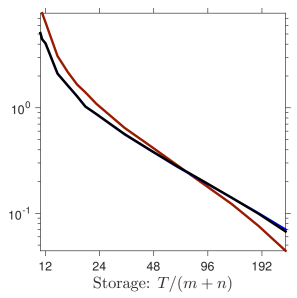
FIG. 4: **Relative error for proposed method with *a priori* parameters.** (Gaussian maps, effective rank $R = 10$, approximation rank $r = 10$, **Schatten ∞-norm.**) We compare the oracle performance of the proposed fixed-rank approximation (2.10) with its performance at theoretically justified parameter values. See subsection 6.5 for details.



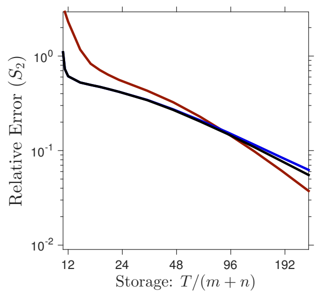
(A) `LowRankHiNoise`



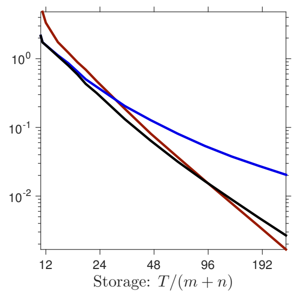
(B) `LowRankMedNoise`



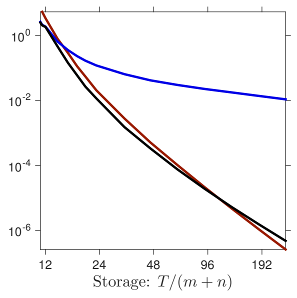
(C) `LowRankLowNoise`



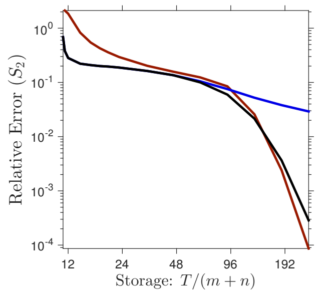
(D) `PolyDecaySlow`



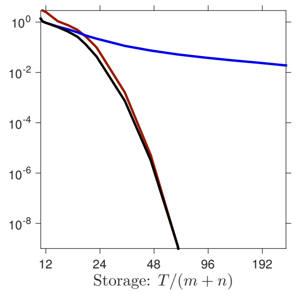
(E) `PolyDecayMed`



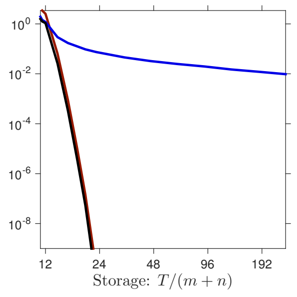
(F) `PolyDecayFast`



(G) `ExpDecaySlow`

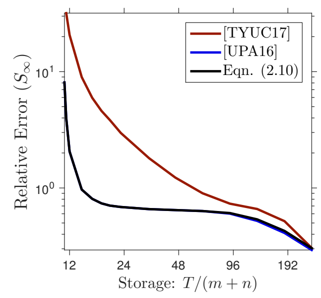


(H) `ExpDecayMed`

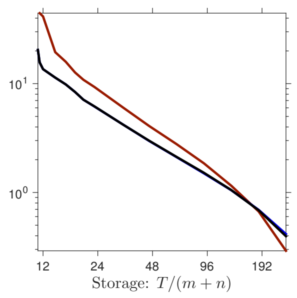


(I) `ExpDecayFast`

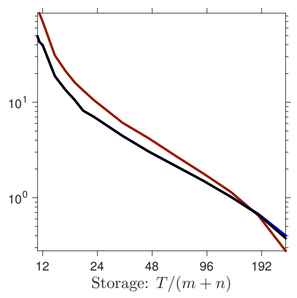
FIG. 5: **Comparison of reconstruction formulas: Synthetic examples.** (Gaussian maps, effective rank $R = 10$, approximation rank $r = 10$, **Schatten 2-norm**.) We compare the oracle error achieved by the proposed fixed-rank approximation (2.10) against methods (6.1) and (6.2) from the literature. See subsection 6.2.2 for details.



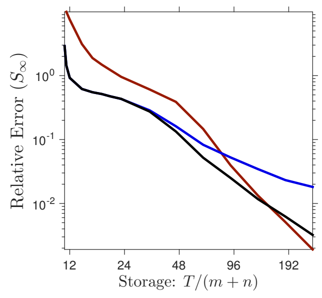
(A) `LowRankHiNoise`



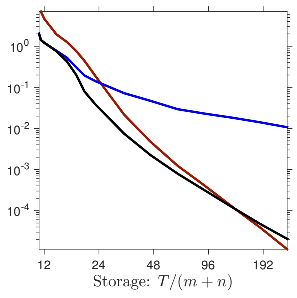
(B) `LowRankMedNoise`



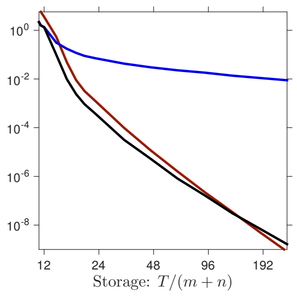
(C) `LowRankLowNoise`



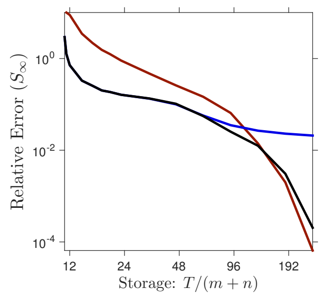
(D) `PolyDecaySlow`



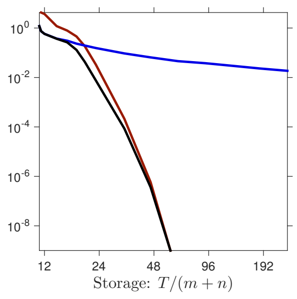
(E) `PolyDecayMed`



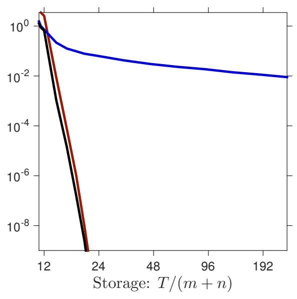
(F) `PolyDecayFast`



(G) `ExpDecaySlow`

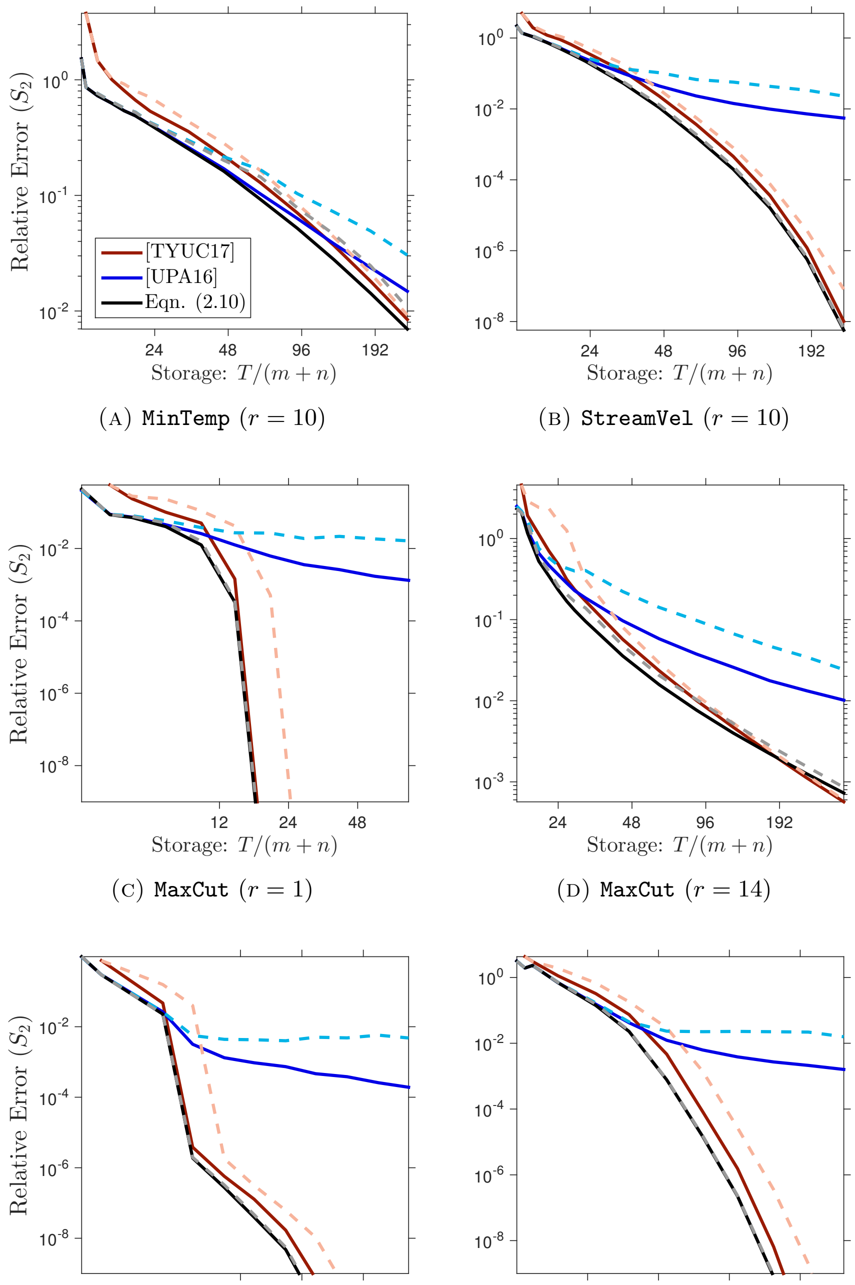


(H) `ExpDecayMed`



(I) `ExpDecayFast`

FIG. 6: **Comparison of reconstruction formulas: Synthetic examples.** (Gaussian maps, effective rank $R = 10$, approximation rank $r = 10$, **Schatten ∞-norm.**) We compare the oracle error achieved by the proposed fixed-rank approximation (2.10) against methods (6.1) and (6.2) from the literature. See subsection 6.2.2 for details.



(A) `MinTemp` ($r = 10$)

(B) `StreamVel` ($r = 10$)

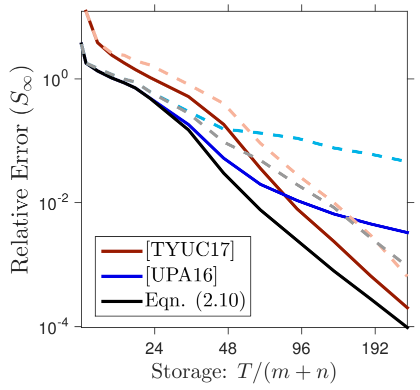
(C) `MaxCut` ($r = 1$)

(D) `MaxCut` ($r = 14$)

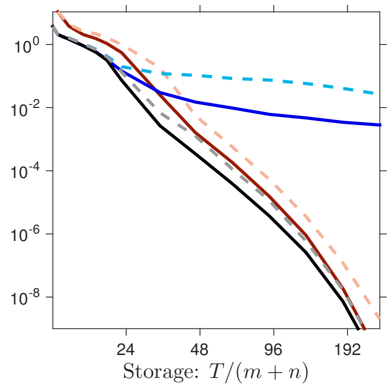
(E) `PhaseRetrieval` ($r = 1$)

(F) `PhaseRetrieval` ($r = 5$)

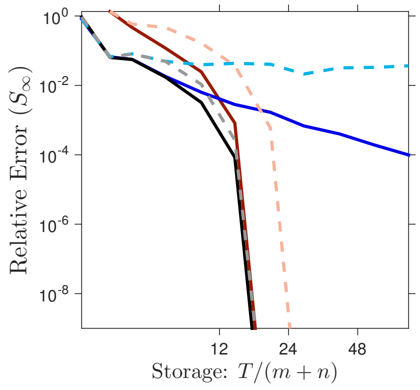
FIG. 7: **Comparison of reconstruction formulas: Real data examples.** (Sparse maps, **Schatten 2-norm**.) We compare the relative error achieved by the proposed fixed-rank approximation (2.10) against methods (6.1) and (6.2) from the literature. **Solid lines** are oracle errors; **dashed lines** are errors with "natural" parameter choices. See subsection 6.7 for details.



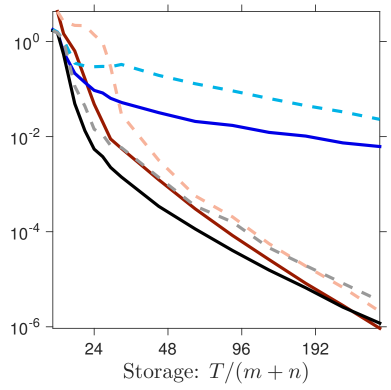
(A) `MinTemp` ($r = 10$)



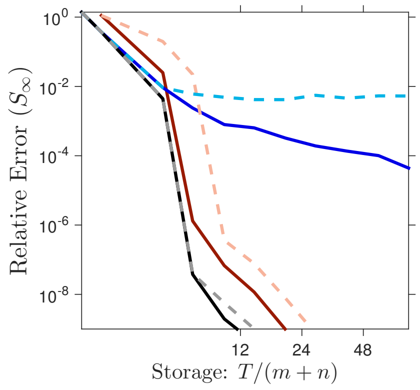
(B) `StreamVel` ($r = 10$)



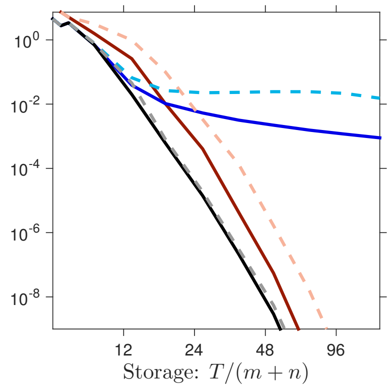
(C) `MaxCut` ($r = 1$)



(D) `MaxCut` ($r = 14$)



(E) `PhaseRetrieval` ($r = 1$)



(F) `PhaseRetrieval` ($r = 5$)

FIG. 8: **Comparison of reconstruction formulas: Real data examples.** (Sparse maps, **Schatten ∞-norm.**) We compare the relative error achieved by the proposed fixed-rank approximation (2.10) against methods (6.1) and (6.2) from the literature. **Solid lines** are oracle errors; **dashed lines** are errors with "natural" parameter choices. See subsection 6.7 for details.

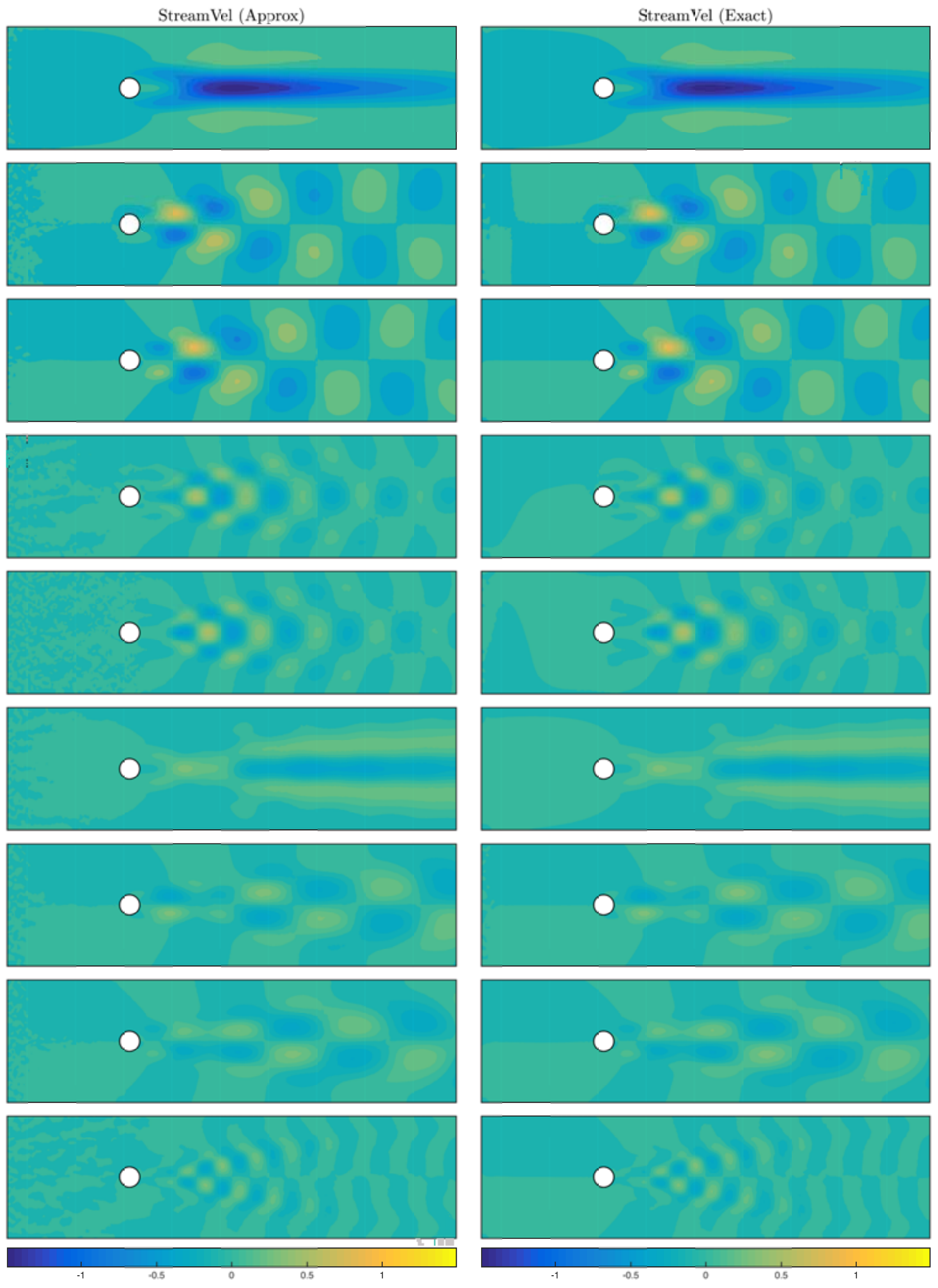


FIG. 9: **Left singular vectors of `StreamVel`.** (Sparse maps, approximation rank $r = 10$, storage budget $T = 48(m + n)$.) The columns of the matrix `StreamVel` describe the fluctuations of the streamwise velocity field about its mean value as a function of time. From top to bottom, the panels show 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 6.8 for details.

SUPPLEMENTARY MATERIALS: MORE PRACTICAL SKETCHING ALGORITHMS FOR LOW-RANK MATRIX APPROXIMATION[*]

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

SM1. Analysis of the Low-Rank Approximation. This section contains the proof of Theorem 5.1, the theoretical result on the behavior of the basic low-rank approximation (2.9). We maintain the notation from section 2.

SM1.1. Facts about Random Matrices. First, let us state a useful formula that allows us to compute some expectations involving a Gaussian random matrix. This identity is drawn from [SM1, Prop. A.1 and A.6]. See also [SM2, Fact A.1].

FACT SM1.1. *Assume that $t > q + \alpha$. Let $\boldsymbol{G}_1 \in \mathbb{F}^{t\times q}$ and $\boldsymbol{G}_2 \in \mathbb{F}^{t\times p}$ be independent standard normal matrices. For any matrix $\boldsymbol{B}$ with conforming dimensions,*

$$\mathbb{E}\,\|\boldsymbol{G}_1^\dagger \boldsymbol{G}_2 \boldsymbol{B}\|_2^2 = \frac{q}{t-q-\alpha}\cdot\|\boldsymbol{B}\|_2^2.$$

The number α is given by (5.1).

SM1.2. Results from Randomized Linear Algebra. Our argument also depends on the analysis of randomized low-rank approximation developed in [SM1, Sec. 10].

FACT SM1.2 (Halko et al. 2011). *Fix $\boldsymbol{A} \in \mathbb{F}^{m\times n}$. Let ϱ be a natural number such that $\varrho < k - \alpha$. Draw the random test matrix $\boldsymbol{\Omega} \in \mathbb{F}^{k\times n}$ from the standard normal distribution. Then the matrix $\boldsymbol{Q}$ computed by (2.7) satisfies*

$$\mathbb{E}_{\boldsymbol{\Omega}}\,\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\|_2^2 \le \left(1 + \frac{\varrho}{k-\varrho-\alpha}\right)\cdot \tau_{\varrho+1}^2(\boldsymbol{A}).$$

The number α is given by (5.1).

This result follows immediately from the proof of [SM1, Thm. 10.5] using Fact SM1.1 to handle both the real and complex case simultaneously. See also [SM3, Sec. 8.2].

SM1.3. Decomposition of the Core Matrix Approximation Error. The first step in the argument is to obtain a formula for the error in the approximation $\boldsymbol{W} - \boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{P}$. The core matrix $\boldsymbol{W} \in \mathbb{F}^{s\times s}$ is defined in (2.8). We constructed the orthonormal matrices $\boldsymbol{P} \in \mathbb{F}^{n\times k}$ and $\boldsymbol{Q} \in \mathbb{F}^{m\times k}$ in (2.7).

Let us introduce matrices whose ranges are complementary to those of $\boldsymbol{P}$ and $\boldsymbol{Q}$:

$$\begin{aligned} \boldsymbol{P}_\perp\boldsymbol{P}_\perp^* &:= \mathbf{I} - \boldsymbol{P}\boldsymbol{P}^* \quad\text{where}\quad \boldsymbol{P}_\perp \in \mathbb{F}^{n\times(n-k)}; \\ \boldsymbol{Q}_\perp\boldsymbol{Q}_\perp^* &:= \mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^* \quad\text{where}\quad \boldsymbol{Q}_\perp \in \mathbb{F}^{m\times(m-k)}. \end{aligned}$$

[*]First draft: 22 March 2017. First release: 16 July 2018.

Funding: JAT was supported in part by ONR Awards N00014-11-1002, N00014-17-1-214, N00014-17-1-2146, and the Gordon & Betty Moore Foundation. MU was supported in part by DARPA Award FA8750-17-2-0101. VC has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under the grant agreement number 725594 (time-data) and the Swiss National Science Foundation (SNSF) under the grant number 200021_178865.

[†]California Institute of Technology, Pasadena, CA (jtropp@cms.caltech.edu).

[‡]École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland (alp.yurtsever@epfl.ch, volkan.cevher@epfl.ch).

[§]Cornell University, Ithaca, NY (udell@cornell.edu).

The columns of $\boldsymbol{P}_\perp$ and $\boldsymbol{Q}_\perp$ are orthonormal. Next, introduce the submatrices

$$\begin{aligned} \boldsymbol{\Phi}_1 &= \boldsymbol{\Phi Q} \in \mathbb{F}^{s\times k} \quad \text{and} \quad \boldsymbol{\Phi}_2 = \boldsymbol{\Phi Q}_\perp \in \mathbb{F}^{s\times (m-k)}; \\ \boldsymbol{\Psi}_1^* &= \boldsymbol{P}^*\boldsymbol{\Psi}^* \in \mathbb{F}^{k\times s} \quad \text{and} \quad \boldsymbol{\Psi}_2^* = \boldsymbol{P}_\perp^*\boldsymbol{\Psi}^* \in \mathbb{F}^{(n-k)\times s}. \end{aligned} \tag{SM1.1}$$

With this notation at hand, we can state and prove the first result.

Lemma SM1.3 (Decomposition of the Core Matrix Approximation). *Assume that the matrices $\boldsymbol{\Phi}_1$ and $\boldsymbol{\Psi}_1$ have full column rank. Then*

$$\begin{aligned} \boldsymbol{W} - \boldsymbol{Q}^*\boldsymbol{AP} = \boldsymbol{\Phi}_1^\dagger\boldsymbol{\Phi}_2(\boldsymbol{Q}_\perp^*\boldsymbol{AP}) + (\boldsymbol{Q}^*\boldsymbol{AP}_\perp)\boldsymbol{\Psi}_2^*(\boldsymbol{\Psi}_1^\dagger)^* \\ + \boldsymbol{\Phi}_1^\dagger\boldsymbol{\Phi}_2(\boldsymbol{Q}_\perp^*\boldsymbol{AP}_\perp)\boldsymbol{\Psi}_2^*(\boldsymbol{\Psi}_1^\dagger)^*. \end{aligned}$$

Proof. Adding and subtracting terms, we write the core sketch $\boldsymbol{Z}$ as

$$\boldsymbol{Z} = \boldsymbol{\Phi A\Psi}^* = \boldsymbol{\Phi}(\boldsymbol{A} - \boldsymbol{QQ}^*\boldsymbol{APP}^*)\boldsymbol{\Psi}^* + (\boldsymbol{\Phi Q})(\boldsymbol{Q}^*\boldsymbol{AP})(\boldsymbol{P}^*\boldsymbol{\Psi}^*).$$

Using (SM1.1), we identify the matrices $\boldsymbol{\Phi}_1$ and $\boldsymbol{\Psi}_1$. Then left-multiply by $\boldsymbol{\Phi}_1^\dagger$ and right-multiply by $(\boldsymbol{\Psi}_1^\dagger)^*$ to arrive at

$$\boldsymbol{W} = \boldsymbol{\Phi}_1^\dagger\boldsymbol{Z}(\boldsymbol{\Psi}_1^\dagger)^* = \boldsymbol{\Phi}_1^\dagger\boldsymbol{\Phi}(\boldsymbol{A} - \boldsymbol{QQ}^*\boldsymbol{APP}^*)\boldsymbol{\Psi}^*(\boldsymbol{\Psi}_1^\dagger)^* + \boldsymbol{Q}^*\boldsymbol{AP}.$$

We have identified the core matrix $\boldsymbol{W}$, defined in (2.8). Move the term $\boldsymbol{Q}^*\boldsymbol{AP}$ to the left-hand side to isolate the approximation error.

To continue, notice that

$$\boldsymbol{\Phi}_1^\dagger\boldsymbol{\Phi} = \boldsymbol{\Phi}_1^\dagger\boldsymbol{\Phi QQ}^* + \boldsymbol{\Phi}_1^\dagger\boldsymbol{\Phi Q}_\perp\boldsymbol{Q}_\perp^* = \boldsymbol{Q}^* + \boldsymbol{\Phi}_1^\dagger\boldsymbol{\Phi}_2\boldsymbol{Q}_\perp^*.$$

Likewise,

$$\boldsymbol{\Psi}^*(\boldsymbol{\Psi}_1^\dagger)^* = \boldsymbol{PP}^*\boldsymbol{\Psi}^*(\boldsymbol{\Psi}_1^\dagger)^* + \boldsymbol{P}_\perp\boldsymbol{P}_\perp^*\boldsymbol{\Psi}^*(\boldsymbol{\Psi}_1^\dagger)^* = \boldsymbol{P} + \boldsymbol{P}_\perp\boldsymbol{\Psi}_2^*(\boldsymbol{\Psi}_1^\dagger)^*.$$

Combine the last three displays to arrive at

$$\boldsymbol{W} - \boldsymbol{Q}^*\boldsymbol{AP} = (\boldsymbol{Q}^* + \boldsymbol{\Phi}_1^\dagger\boldsymbol{\Phi}_2\boldsymbol{Q}_\perp^*)(\boldsymbol{A} - \boldsymbol{QQ}^*\boldsymbol{APP}^*)(\boldsymbol{P} + \boldsymbol{P}_\perp\boldsymbol{\Psi}_2^*(\boldsymbol{\Psi}_1^\dagger)^*).$$

Expand the expression and use the orthogonality relations $\boldsymbol{Q}^*\boldsymbol{Q} = \mathbf{I}$ and $\boldsymbol{Q}_\perp^*\boldsymbol{Q} = \mathbf{0}$ and $\boldsymbol{P}^*\boldsymbol{P} = \mathbf{I}$ and $\boldsymbol{P}^*\boldsymbol{P}_\perp = \mathbf{0}$ to arrive at the desired representation. ∎

SM1.4. Probabilistic Analysis of the Core Matrix. Next, we make distributional assumptions on the dimension reduction maps $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$. We can then study the probabilistic behavior of the error $\boldsymbol{W} - \boldsymbol{Q}^*\boldsymbol{AP}$.

Lemma SM1.4 (Probabilistic Analysis of the Core Matrix). *Assume that the dimension reduction matrices $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ are drawn independently from the standard normal distribution. When $s \geq k$, it holds that*

$$\mathbb{E}_{\boldsymbol{\Phi},\boldsymbol{\Psi}}[\boldsymbol{W} - \boldsymbol{Q}^*\boldsymbol{AP}] = \mathbf{0}.$$

When $s > k + \alpha$, we can express the error as

$$\begin{aligned} \mathbb{E}_{\boldsymbol{\Phi},\boldsymbol{\Psi}}\|\boldsymbol{W} - \boldsymbol{Q}^*\boldsymbol{AP}\|_2^2 = \frac{k}{s-k-\alpha}\cdot\|\boldsymbol{A} - \boldsymbol{QQ}^*\boldsymbol{APP}^*\|_2^2 \\ + \frac{k(2k+\alpha-s)}{(s-k-\alpha)^2}\cdot\|\boldsymbol{Q}_\perp^*\boldsymbol{AP}_\perp\|_2^2. \end{aligned}$$

When $s \geq 2k + \alpha$, the last term is always nonpositive.

Proof. Since $\boldsymbol{\Phi}$ is standard normal, the orthogonal submatrices $\boldsymbol{\Phi}_1$ and $\boldsymbol{\Phi}_2$ are statistically independent standard normal matrices because of the marginal property of the normal distribution. Likewise, $\boldsymbol{\Psi}_1$ and $\boldsymbol{\Psi}_2$ are statistically independent standard normal matrices. Provided that $s \geq k$, both matrices have full column rank with probability one.

To establish the first point, notice that

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\Phi},\boldsymbol{\Psi}}[\boldsymbol{W} - \boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{P}] &= \mathbb{E}_{\boldsymbol{\Phi}_1}\mathbb{E}_{\boldsymbol{\Phi}_2}[\boldsymbol{\Phi}_1^\dagger\boldsymbol{\Phi}_2(\boldsymbol{Q}_\perp^*\boldsymbol{A}\boldsymbol{P})] + \mathbb{E}_{\boldsymbol{\Psi}_1}\mathbb{E}_{\boldsymbol{\Psi}_2}[(\boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{P}_\perp)\boldsymbol{\Psi}_2^*(\boldsymbol{\Psi}_1^\dagger)^*] \\ &\quad + \mathbb{E}\,\mathbb{E}_{\boldsymbol{\Phi}_2}[\boldsymbol{\Phi}_1^\dagger\boldsymbol{\Phi}_2(\boldsymbol{Q}_\perp^*\boldsymbol{A}\boldsymbol{P}_\perp)\boldsymbol{\Psi}_2^*(\boldsymbol{\Psi}_1^\dagger)^*].\end{aligned}$$

We have used the decomposition of the approximation error from Lemma SM1.3. Then we invoke independence to write the expectations as iterated expectations. Since $\boldsymbol{\Phi}_2$ and $\boldsymbol{\Psi}_2$ have mean zero, this formula makes it clear that the approximation error has mean zero.

To study the fluctuations, apply the independence and zero-mean property of $\boldsymbol{\Phi}_2$ and $\boldsymbol{\Psi}_2$ to decompose

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\Phi},\boldsymbol{\Psi}}\|\boldsymbol{W} - \boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{P}\|_2^2 &= \mathbb{E}_{\boldsymbol{\Phi}}\|\boldsymbol{\Phi}_1^\dagger\boldsymbol{\Phi}_2(\boldsymbol{Q}_\perp^*\boldsymbol{A}\boldsymbol{P})\|_2^2 + \mathbb{E}_{\boldsymbol{\Psi}}\|(\boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{P}_\perp)\boldsymbol{\Psi}_2^*(\boldsymbol{\Psi}_1^\dagger)^*\|_2^2 \\ &\quad + \mathbb{E}_{\boldsymbol{\Phi}}\mathbb{E}_{\boldsymbol{\Psi}}\|\boldsymbol{\Phi}_1^\dagger\boldsymbol{\Phi}_2(\boldsymbol{Q}_\perp^*\boldsymbol{A}\boldsymbol{P}_\perp)\boldsymbol{\Psi}_2^*(\boldsymbol{\Psi}_1^\dagger)^*\|_2^2.\end{aligned}$$

Continuing, we invoke Fact SM1.1 four times to see that

$$\begin{aligned}&\mathbb{E}_{\boldsymbol{\Phi},\boldsymbol{\Psi}}\|\boldsymbol{W} - \boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{P}\|_2^2 \\ &\qquad = \frac{k}{s-k-\alpha}\cdot\left[\|\boldsymbol{Q}_\perp^*\boldsymbol{A}\boldsymbol{P}\|_2^2 + \|\boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{P}_\perp\|_2^2 + \frac{k}{s-k-\alpha}\cdot\|\boldsymbol{Q}_\perp^*\boldsymbol{A}\boldsymbol{P}_\perp\|_2^2\right].\end{aligned}$$

Add and subtract $\|\boldsymbol{Q}_\perp^*\boldsymbol{A}\boldsymbol{P}_\perp\|_2^2$ in the bracket to arrive at

$$\begin{aligned}\mathbb{E}\|\boldsymbol{W} - \boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{P}\|_2^2 = \frac{k}{s-k-\alpha}\cdot\Bigg[&\|\boldsymbol{Q}_\perp^*\boldsymbol{A}\boldsymbol{P}\|_2^2 + \|\boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{P}_\perp\|_2^2 + \|\boldsymbol{Q}_\perp^*\boldsymbol{A}\boldsymbol{P}_\perp\|_2^2 \\ &+ \frac{2k+\alpha-s}{s-k-\alpha}\cdot\|\boldsymbol{Q}_\perp^*\boldsymbol{A}\boldsymbol{P}_\perp\|_2^2\Bigg].\end{aligned}$$

Use the Pythagorean Theorem to combine the terms on the first line. ∎

SM1.5. Probabilistic Analysis of the Compression Error.

Next, we establish a bound for the expected error in the compression of the matrix $\boldsymbol{A}$ onto the range of the matrices $\boldsymbol{Q}$ and $\boldsymbol{P}$, computed in (2.7). This result is similar in spirit to the analysis in [SM1], so we pass lightly over the details.

LEMMA SM1.5 (Probabilistic Analysis of the Compression Error). *For any natural number $\varrho < k - \alpha$, it holds that*

$$\mathbb{E}\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{P}\boldsymbol{P}^*\|_2^2 \leq \left(1 + \frac{2\varrho}{k-\varrho-\alpha}\right)\cdot\tau_{\varrho+1}^2(\boldsymbol{A}).$$

Proof Sketch. Introduce the partitioned SVD of the matrix $\boldsymbol{A}$:

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^* = \begin{bmatrix}\boldsymbol{U}_1 & \boldsymbol{U}_2\end{bmatrix}\begin{bmatrix}\boldsymbol{\Sigma}_1 & \\ & \boldsymbol{\Sigma}_2\end{bmatrix}\begin{bmatrix}\boldsymbol{V}_1^* \\ \boldsymbol{V}_2^*\end{bmatrix} \quad \text{where} \quad \boldsymbol{\Sigma}_1 \in \mathbb{F}^{\varrho\times\varrho}.$$

Define the matrices

$$\begin{aligned} \boldsymbol{\Upsilon}_1 &:= \boldsymbol{\Upsilon}\boldsymbol{U}_1 \in \mathbb{F}^{s\times\varrho} \quad\text{and}\quad \boldsymbol{\Upsilon}_2 := \boldsymbol{\Upsilon}\boldsymbol{U}_2 \in F^{s\times(m-\varrho)};\\ \boldsymbol{\Omega}_1^* &:= \boldsymbol{V}_1^*\boldsymbol{\Omega}^* \in \mathbb{F}^{\varrho\times s} \quad\text{and}\quad \boldsymbol{\Omega}_2^* := \boldsymbol{V}_2^*\boldsymbol{\Omega}^* \in \mathbb{F}^{(n-\varrho)\times s};\\ \boldsymbol{P}_1 &:= \boldsymbol{V}_1^*\boldsymbol{P} \in \mathbb{F}^{\varrho\times k} \quad\text{and}\quad \boldsymbol{P}_2 := \boldsymbol{V}_2^*\boldsymbol{P} \in \mathbb{F}^{(n-\varrho)\times k}. \end{aligned}$$

With this notation, we proceed to the proof.

First, add and subtract terms and apply the Pythagorean Theorem to obtain

$$\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{P}\boldsymbol{P}^*\|_2^2 = \|\boldsymbol{A}(\mathbf{I} - \boldsymbol{P}\boldsymbol{P}^*)\|_2^2 + \|(\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^*)\boldsymbol{A}\boldsymbol{P}\boldsymbol{P}^*\|_2^2.$$

Use the SVD to decompose the matrix $\boldsymbol{A}$ in the first term, and apply the Pythagorean Theorem again:

$$\begin{aligned} \|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{P}\boldsymbol{P}^*\|_2^2 &= \|(\boldsymbol{U}_2\boldsymbol{\Sigma}_2\boldsymbol{V}_2^*)(\mathbf{I} - \boldsymbol{P}\boldsymbol{P}^*)\|_2^2 \\ &\qquad + \|(\boldsymbol{U}_1\boldsymbol{\Sigma}_1\boldsymbol{V}_1^*)(\mathbf{I} - \boldsymbol{P}\boldsymbol{P}^*)\|_2^2 + \|(\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^*)\boldsymbol{A}\boldsymbol{P}\|_2^2. \end{aligned}$$

The result [SM3, Prop. 9.2] implies that the second term satisfies

$$\|(\boldsymbol{U}_1\boldsymbol{\Sigma}_1\boldsymbol{V}_1^*)(\mathbf{I} - \boldsymbol{P}\boldsymbol{P}^*)\|_2^2 \le \|\boldsymbol{\Upsilon}_1^\dagger\boldsymbol{\Upsilon}_2\boldsymbol{\Sigma}_2\|_2^2.$$

We can obtain a bound for the third term using the formula [SM1, p. 270, disp. 1]. After a short computation, this result yields

$$\begin{aligned} \|(\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^*)\boldsymbol{A}\boldsymbol{P}\|_2^2 &\le \|\boldsymbol{\Sigma}_2\boldsymbol{P}_2\|_2^2 + \|\boldsymbol{\Sigma}_2\boldsymbol{\Omega}_2^*(\boldsymbol{\Omega}_1^*)^\dagger\boldsymbol{P}_1\|_2^2 \\ &\le \|(\boldsymbol{U}_2\boldsymbol{\Sigma}_2\boldsymbol{V}_2^*)\boldsymbol{P}\|_2^2 + \|\boldsymbol{\Sigma}_2\boldsymbol{\Omega}_2^*(\boldsymbol{\Omega}_1^*)^\dagger\|_2^2. \end{aligned}$$

We can remove $\boldsymbol{P}_1$ because its spectral norm is bounded by one, being a submatrix of an orthonormal matrix. Combine the last three displays to obtain

$$\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{P}\boldsymbol{P}^*\|_2^2 \le \|\boldsymbol{U}_2\boldsymbol{\Sigma}_2\boldsymbol{V}_2^*\|_2^2 + \|\boldsymbol{\Upsilon}_1^\dagger\boldsymbol{\Upsilon}_2\boldsymbol{\Sigma}_2\|_2^2 + \|\boldsymbol{\Sigma}_2\boldsymbol{\Omega}_2^*(\boldsymbol{\Omega}_1^*)^\dagger\|_2^2.$$

We have used the Pythagorean Theorem again.

Take the expectation with respect to $\boldsymbol{\Upsilon}$ and $\boldsymbol{\Omega}$ to arrive at

$$\begin{aligned} \mathbb{E}\,\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{P}\boldsymbol{P}^*\|_2^2 &\le \|\boldsymbol{\Sigma}_2\|_2^2 + \mathbb{E}\,\|\boldsymbol{\Upsilon}_1^\dagger\boldsymbol{\Upsilon}_2\boldsymbol{\Sigma}_2\|_2^2 + \mathbb{E}\,\|\boldsymbol{\Sigma}_2\boldsymbol{\Omega}_2^*(\boldsymbol{\Omega}_1^*)^\dagger\|_2^2 \\ &= \|\boldsymbol{\Sigma}_2\|_2^2 + \frac{2\varrho}{k-\varrho-\alpha}\cdot\|\boldsymbol{\Sigma}_2\|_2^2. \end{aligned}$$

Finally, note that $\|\boldsymbol{\Sigma}_2\|_2^2 = \tau_{\varrho+1}^2(\boldsymbol{A})$. ∎

SM1.6. The Endgame. At last, we are prepared to finish the proof of Theorem 5.1. Fix a natural number $\varrho < k - \alpha$. Using the formula (2.9) for the approximation $\hat{\boldsymbol{A}}$, we see that

$$\begin{aligned} \|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_2^2 &= \|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{W}\boldsymbol{P}^*\|_2^2 \\ &= \|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{P}\boldsymbol{P}^* + \boldsymbol{Q}(\boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{P} - \boldsymbol{W})\boldsymbol{P}^*\|_2^2 \\ &= \|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{P}\boldsymbol{P}^*\|_2^2 + \|\boldsymbol{Q}(\boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{P} - \boldsymbol{W})\boldsymbol{P}^*\|_2^2. \end{aligned}$$

The last identity is the Pythagorean theorem. Drop the orthonormal matrices in the last term. Then take the expectation with respect to $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$:

$$\mathbb{E}_{\boldsymbol{\Phi},\boldsymbol{\Psi}}\,\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_2^2 = \|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{P}\boldsymbol{P}^*\|_2^2 + \mathbb{E}_{\boldsymbol{\Phi},\boldsymbol{\Psi}}\,\|\boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{P} - \boldsymbol{W}\|_2^2$$

We treat the two terms sequentially.

To continue, invoke the expression Lemma SM1.4 for the expected error in the core matrix $\boldsymbol{W}$:

$$\mathbb{E}_{\boldsymbol{\Phi},\boldsymbol{\Psi}} \|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_2^2 \leq \left(1 + \frac{k}{s-k-\alpha}\right) \cdot \|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{P}\boldsymbol{P}^*\|_2^2 + \frac{k(2k+\alpha-s)}{(s-k-\alpha)^2} \cdot \|\boldsymbol{Q}_\perp^*\boldsymbol{A}\boldsymbol{P}_\perp\|_2^2.$$

Now, take the expectation with respect to $\boldsymbol{\Upsilon}$ and $\boldsymbol{\Omega}$ to arrive at

(SM1.2)
$$\mathbb{E}\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_2^2 \leq \left(1 + \frac{k}{s-k-\alpha}\right) \cdot \left(1 + \frac{2\varrho}{k-\varrho-\alpha}\right) \cdot \tau_{\varrho+1}^2(\boldsymbol{A}) + \frac{k(2k+\alpha-s)}{(s-k-\alpha)^2} \cdot \mathbb{E}\|\boldsymbol{Q}_\perp^*\boldsymbol{A}\boldsymbol{P}_\perp\|_2^2.$$

We have invoked Lemma SM1.5. The last term is nonpositive because we require $s \geq 2k + \alpha$, so we may drop it from consideration. Finally, we optimize over eligible choices $\varrho < k - \alpha$ to complete the argument. The result stated in Theorem 5.1 is algebraically equivalent.

SM2. Code & Pseudocode.

This supplement contains pseudocode for the sketching and low-rank reconstruction algorithms described in this paper. In many places, we use the same mathematical notation as the rest of the paper. We also rely on MATLAB R2018A commands, which appear in typewriter font. The electronic materials include a MATLAB implementation of these methods.

- Algorithm SM3.1 contains the constructor for the SKETCH object. It draws random test matrices and initializes the sketch for the zero input matrix. This code implements (2.2)–(2.4).
- Algorithm SM3.2 implements a general rank-one linear update (2.5) to the input matrix contained in the sketch.
- Algorithm SM3.3 implements the basic low-rank reconstruction formula (2.9). It returns the approximation in factored form.
- Algorithm SM3.4 implements the rank-r reconstruction formula (2.10). It returns the approximation in factored form.
- Algorithm SM3.5 is the template for the dimension reduction (DIMREDUX) class for input matrices over the field $\mathbb{F}$. It outlines the methods that a DIMREDUX needs to implement.
- Algorithm SM3.6 defines the Gaussian dimension reduction (GAUSS) class, which is a subclass of DIMREDUX. It describes the constructor and the left and right action of this dimension reduction map. See subsection 3.1 for the explanation.
- Algorithm SM3.7 defines the SSRFT dimension reduction (SSRFT) class, which is a subclass of DIMREDUX. It describes the constructor and the left and right action of this dimension reduction map. See subsection 3.2 for the explanation.
- Algorithm SM3.8 defines the sparse dimension reduction (SPARSE) class, which is a subclass of DIMREDUX. It describes the constructor and the left and right action of this dimension reduction map. See subsection 3.3 for the explanation.

SM3. Supplemental Numerical Results.

This section summarizes the additional numerical results that are presented in this supplement. The MATLAB code in the electronic materials can reproduce these experiments.

SM3.1. Insensitivity to the Dimension Reduction Map.

We undertook a more comprehensive set of experiments to demonstrate that our reconstruction formula (2.10) is insensitive to the choice of dimension reduction map at the oracle parameters. See subsection 6.4 for details.

Figures SM1 to SM5 contain the results for matrices with effective rank $R = 5$, $R = 10$, and $R = 20$ with relative error measured in Schatten 2-norm and Schatten ∞-norm.

SM3.2. Achieving the Oracle Performance.

We also performed experiments to see how closely the theoretical parameter choices allow us to approach the oracle performance of our reconstruction formula (2.10). See subsection 6.5 for details.

Figures SM6 to SM9 contain the results for matrices with effective rank $R = 5$ and $R = 20$ with relative error measured in Schatten 2-norm and Schatten ∞-norm.

SM3.3. Algorithm Comparisons for Synthetic Instances.

We compared all three of the reconstruction formulas (2.10), (6.1), and (6.2) at the oracle parameters for a wide range of synthetic problem instances. See subsection 6.6 for details.

Figures SM10 to SM13 contain the results for matrices with effective rank $R = 5$ and $R = 20$ with relative error measured in Schatten 2-norm and Schatten ∞-norm.

SM3.4. Flow-Field Reconstruction.

Figure SM14 illustrates the streamwise velocity field `StreamVel` and its rank-10 approximation via (2.10) using storage budget $T/(m + n) = 48$ and the parameter choices (5.6). We see that the approximation captures the large-scale features of the flow, although there are small errors visible, especially at the inlet (on the left-hand side of the images).

REFERENCES

[SM1] 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.

[SM2] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher, *Practical sketching algorithms for low-rank matrix approximation*. Submitted, Jan. 2017.

[SM3] 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, Jan. 2017. Available at http://arXiv.org/abs/1609.00048, v1.

Algorithm SM3.1 *Sketch for Low-Rank Approximation.* Implements (2.2)–(2.4).

Input: Input matrix dimensions $m \times n$; sketch size parameters $k \leq s \leq \min\{m, n\}$
Output: Draw dimension reduction maps (2.2); sketch (2.3) and (2.4) of $\boldsymbol{A} = \mathbf{0}$

```
class Sketch
    local variables $\boldsymbol{\Upsilon}, \boldsymbol{\Omega}, \boldsymbol{\Phi}, \boldsymbol{\Psi}$ (DimRedux)
    local variables $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ (matrices)

    function Sketch($m, n, k, s$; DR)            ▷ Constructor; DR is a DimRedux
        $\boldsymbol{\Upsilon} \leftarrow \mathrm{DR}(k, m)$          ▷ Draw new dimension reduction maps
        $\boldsymbol{\Omega} \leftarrow \mathrm{DR}(k, n)$
        $\boldsymbol{\Phi} \leftarrow \mathrm{DR}(s, m)$
        $\boldsymbol{\Psi} \leftarrow \mathrm{DR}(s, n)$
        $\boldsymbol{X} \leftarrow \texttt{zeros}(k, n)$                  ▷ Sketch of zero matrix
        $\boldsymbol{Y} \leftarrow \texttt{zeros}(m, k)$
        $\boldsymbol{Z} \leftarrow \texttt{zeros}(s, s)$
```

Algorithm SM3.2 *Linear Update to Sketch.* Implements (2.5).

Input: Innovation $\boldsymbol{H} \in \mathbb{F}^{m \times n}$; scalars $\theta, \tau \in \mathbb{F}$
Output: Modifies sketch to reflect linear update $\boldsymbol{A} \leftarrow \theta \boldsymbol{A} + \tau \boldsymbol{H}$

```
function Sketch.LinearUpdate($\boldsymbol{H}; \theta, \tau$)
    $\boldsymbol{X} \leftarrow \theta \boldsymbol{X} + \tau \boldsymbol{\Upsilon} \boldsymbol{H}$
    $\boldsymbol{Y} \leftarrow \theta \boldsymbol{Y} + \tau \boldsymbol{H} \boldsymbol{\Omega}^*$
    $\boldsymbol{Z} \leftarrow \theta \boldsymbol{Z} + \tau \boldsymbol{\Phi} \boldsymbol{H} \boldsymbol{\Psi}^*$
```

Algorithm SM3.3 *Low-Rank Approximation.* Implements (2.9).

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

```
function Sketch.LowRankApprox( )
    $(\boldsymbol{Q}, \sim) \leftarrow \texttt{qr}(\boldsymbol{Y}, 0)$
    $(\boldsymbol{P}, \sim) \leftarrow \texttt{qr}(\boldsymbol{X}^*, 0)$
    $\boldsymbol{W} \leftarrow ((\boldsymbol{\Phi}\boldsymbol{Q}) \backslash \boldsymbol{Z}) / ((\boldsymbol{\Psi}\boldsymbol{P})^*)$          ▷ Least-squares via QR or SVD
    return $(\boldsymbol{Q}, \boldsymbol{W}, \boldsymbol{P})$
```

Algorithm SM3.4 *Fixed-Rank Approximation.* Implements (2.10).

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

```
function SKETCH.FIXEDRANKAPPROX($r$)
    $(\boldsymbol{Q}, \boldsymbol{W}, \boldsymbol{P}) \leftarrow$ SKETCH.LOWRANKAPPROX( )
    $(\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}) \leftarrow$ svds($\boldsymbol{W}, r$)        ▷ Truncate full SVD to rank $r$
    $\boldsymbol{U} \leftarrow \boldsymbol{Q}\boldsymbol{U}$        ▷ Consolidate unitary factors
    $\boldsymbol{V} \leftarrow \boldsymbol{P}\boldsymbol{V}$
    return $(\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V})$
```

Algorithm SM3.5 *Dimension Reduction Map Class.*

```
class DIMREDUX ($\mathbb{F}$)        ▷ Dimension reduction map over field $\mathbb{F}$
    function DIMREDUX($k, n$)        ▷ Construct map $\boldsymbol{\Xi} : \mathbb{F}^n \to \mathbb{F}^k$
    function DIMREDUX.MTIMES(DRmap, $\boldsymbol{M}$)        ▷ Left action of map
    function DIMREDUX.MTIMES($\boldsymbol{M}$, DRmap$^*$)        ▷ Right action of adjoint
        return (DIMREDUX.MTIMES(DRmap, $\boldsymbol{M}^*$))$^*$        ▷ Default behavior
```

Algorithm SM3.6 *Gaussian Dimension Reduction Map.* (subsection 3.1)

```
class GAUSS (DIMREDUX)        ▷ Subclass of DIMREDUX
    local variable $\boldsymbol{\Xi}$ (dense matrix)

    function RANDN($k, n; \mathbb{F}$)        ▷ Gaussian matrix over field $\mathbb{F}$
        if $\mathbb{F} = \mathbb{R}$ then return randn($k, n$)
        if $\mathbb{F} = \mathbb{C}$ then return randn($k, n$) + 1i * randn($k, n$)

    function GAUSS($k, n$)        ▷ Constructor
        $\boldsymbol{\Xi} \leftarrow$ RANDN($k, n; \mathbb{F}$)        ▷ Gaussian over $\mathbb{F}$

    function GAUSS.MTIMES(DRmap, $\boldsymbol{M}$)
        return mtimes($\boldsymbol{\Xi}$, $\boldsymbol{M}$)
```

Algorithm SM3.7 *SSRFT Dimension Reduction Map.* (subsection 3.2)

```
class SSRFT (DimRedux)                                   ▷ Subclass of DimRedux
local variables coords, $\texttt{perm}_j$, $\varepsilon_j$ for $j = 1, 2$

function SSRFT($k, n$)                                   ▷ Constructor
    coords ← randperm($n, k$)
    $\texttt{perm}_j$ ← randperm($n$) for $j = 1, 2$
    $\varepsilon_j$ ← sign(randn($n, 1; \mathbb{F}$)) for $j = 1, 2$

function SSRFT.mtimes(DRmap, $\boldsymbol{M}$)
    if $\mathbb{F} = \mathbb{R}$ then
        $\boldsymbol{M}$ ← dct(diag($\varepsilon_1$)$\boldsymbol{M}$($\texttt{perm}_1$, :))
        $\boldsymbol{M}$ ← dct(diag($\varepsilon_2$)$\boldsymbol{M}$($\texttt{perm}_2$, :))
    if $\mathbb{F} = \mathbb{C}$ then
        $\boldsymbol{M}$ ← dft(diag($\varepsilon_1$)$\boldsymbol{M}$($\texttt{perm}_1$, :))
        $\boldsymbol{M}$ ← dft(diag($\varepsilon_2$)$\boldsymbol{M}$($\texttt{perm}_2$, :))
    return $\boldsymbol{M}$(coords, :)
```

Algorithm SM3.8 *Sparse Dimension Reduction Map.* (subsection 3.3)

```
class Sparse (DimRedux)                                  ▷ Subclass of DimRedux
local variable $\boldsymbol{\Xi}$ (sparse matrix)

function Sparse($k, n$)                                  ▷ Constructor
    $\zeta \leftarrow \min\{k, \lfloor 2\log(1+n) \rfloor\}$          ▷ Sparsity of each column
    for $j = 1, \dots, n$ do
        $\boldsymbol{\Xi}$(randperm($k, \zeta$), $j$) ← sign(randn($\zeta, 1; \mathbb{F}$))

function Sparse.mtimes(DRmap, $\boldsymbol{M}$)
    return mtimes($\boldsymbol{\Xi}$, $\boldsymbol{M}$)
```

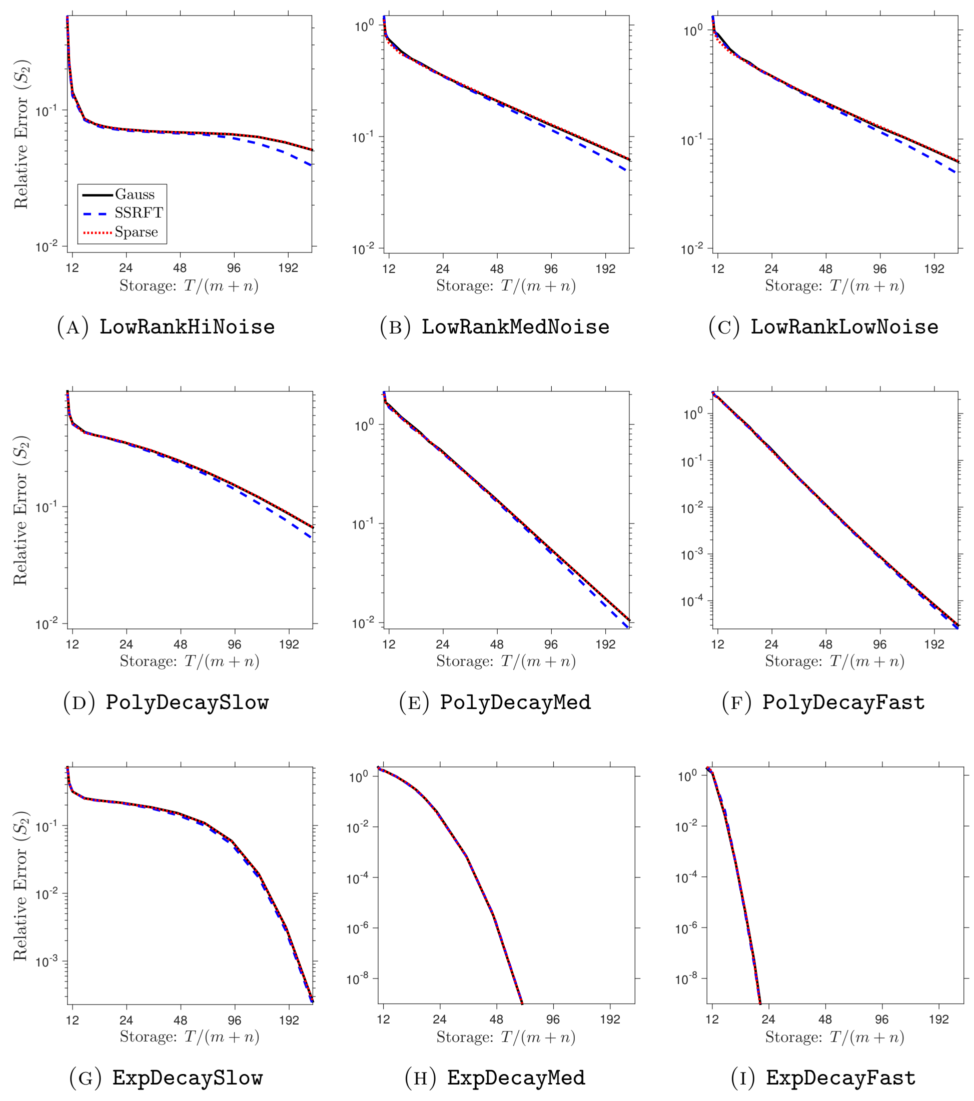
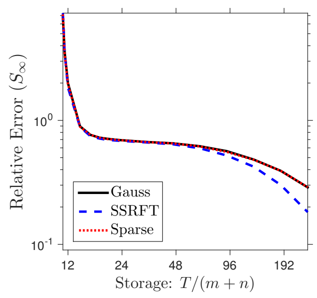
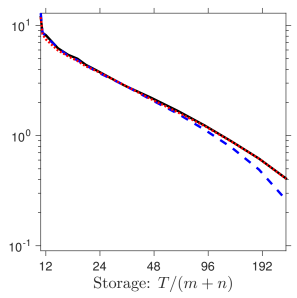


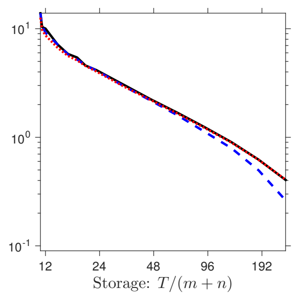
FIG. SM1: **Insensitivity of proposed method to the dimension reduction map.** (Effective rank $R = 5$, approximation rank $r = 10$, Schatten 2-norm.) We compare the oracle performance of the proposed fixed-rank approximation (2.10) implemented with Gaussian, SSRFT, or sparse dimension reduction maps. See subsection 6.4 for details.



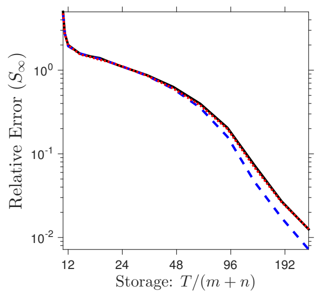
(A) `LowRankHiNoise`



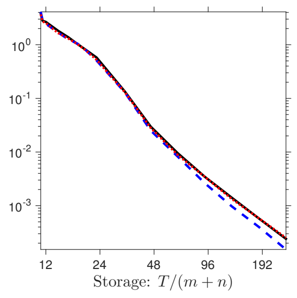
(B) `LowRankMedNoise`



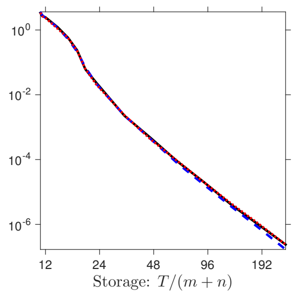
(C) `LowRankLowNoise`



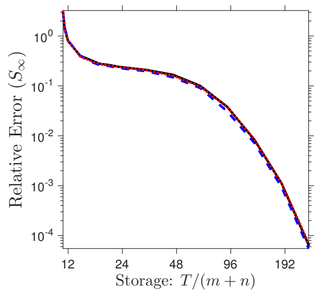
(D) `PolyDecaySlow`



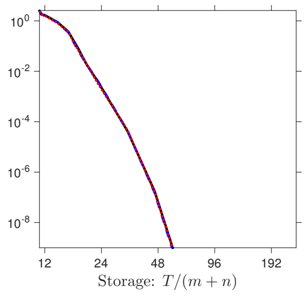
(E) `PolyDecayMed`



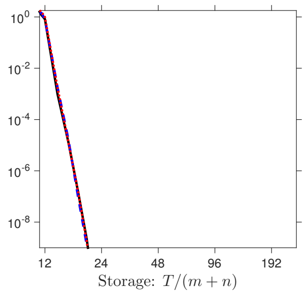
(F) `PolyDecayFast`



(G) `ExpDecaySlow`

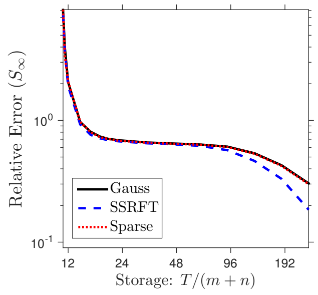


(H) `ExpDecayMed`

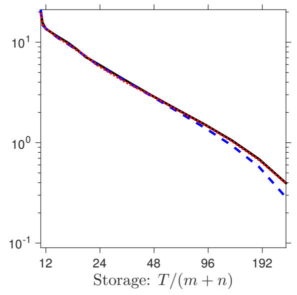


(I) `ExpDecayFast`

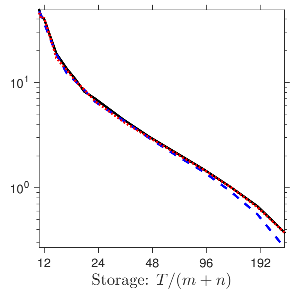
FIG. SM2: **Insensitivity of proposed method to the dimension reduction map.** (Effective rank $R = 5$, approximation rank $r = 10$, Schatten ∞-norm.) We compare the oracle performance of the proposed fixed-rank approximation (2.10) implemented with Gaussian, SSRFT, or sparse dimension reduction maps. See subsection 6.4 for details.



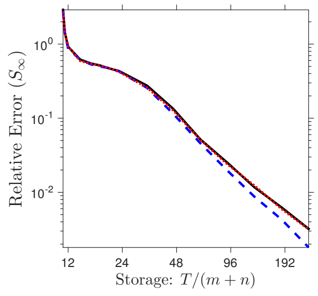
(A) `LowRankHiNoise`



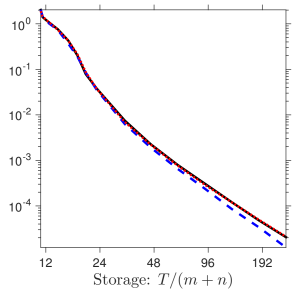
(B) `LowRankMedNoise`



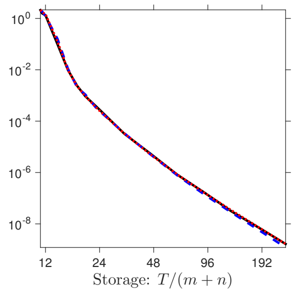
(C) `LowRankLowNoise`



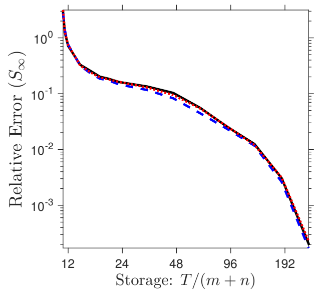
(D) `PolyDecaySlow`



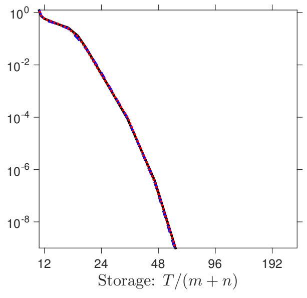
(E) `PolyDecayMed`



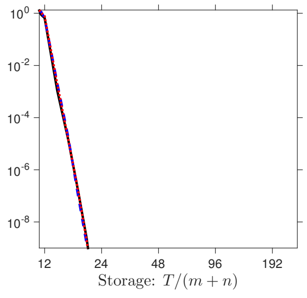
(F) `PolyDecayFast`



(G) `ExpDecaySlow`

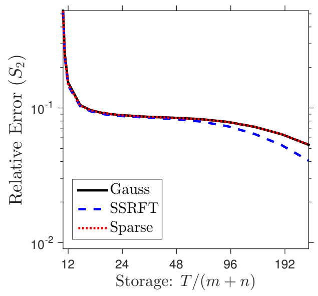


(H) `ExpDecayMed`

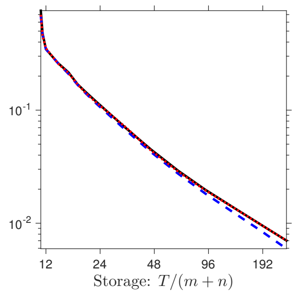


(I) `ExpDecayFast`

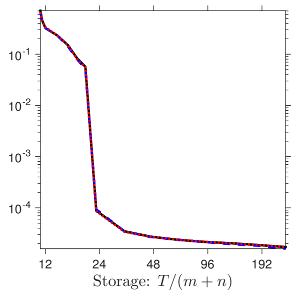
FIG. SM3: **Insensitivity of proposed method to the dimension reduction map.** (Effective rank $R = 10$, approximation rank $r = 10$, Schatten ∞-norm.) We compare the oracle performance of the proposed fixed-rank approximation (2.10) implemented with Gaussian, SSRFT, or sparse dimension reduction maps. See subsection 6.4 for details.



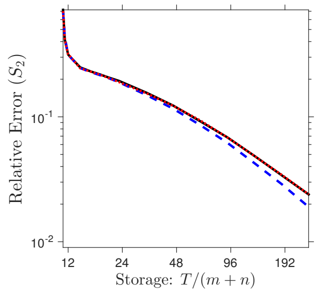
(A) `LowRankHiNoise`



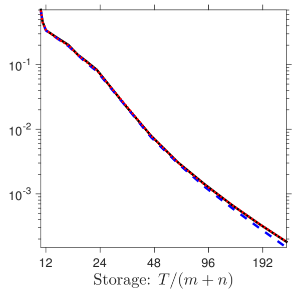
(B) `LowRankMedNoise`



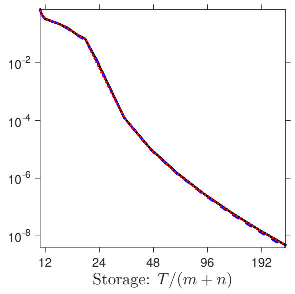
(C) `LowRankLowNoise`



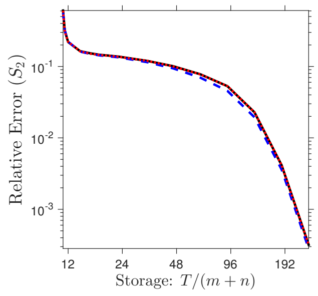
(D) `PolyDecaySlow`



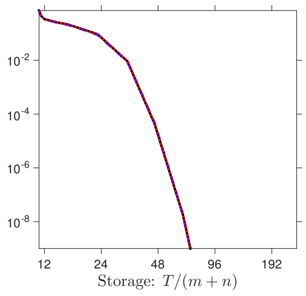
(E) `PolyDecayMed`



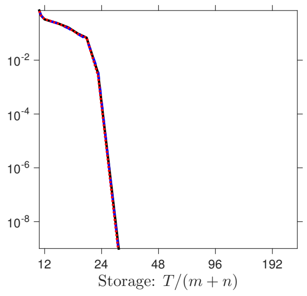
(F) `PolyDecayFast`



(G) `ExpDecaySlow`

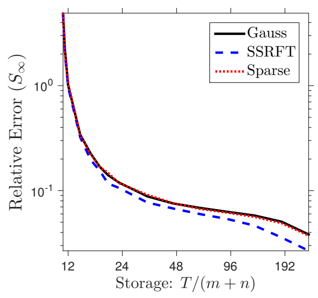


(H) `ExpDecayMed`

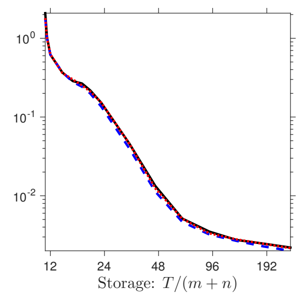


(I) `ExpDecayFast`

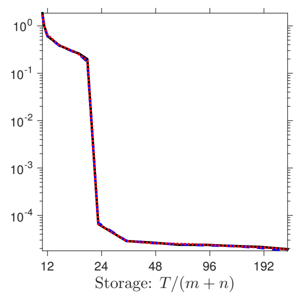
FIG. SM4: **Insensitivity of proposed method to the dimension reduction map.** (Effective rank $R = 20$, approximation rank $r = 10$, Schatten 2-norm.) We compare the oracle performance of the proposed fixed-rank approximation (2.10) implemented with Gaussian, SSRFT, or sparse dimension reduction maps. See subsection 6.4 for details.



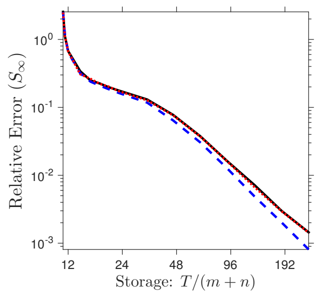
(A) `LowRankHiNoise`



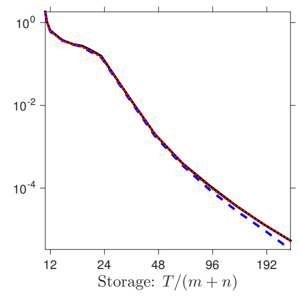
(B) `LowRankMedNoise`



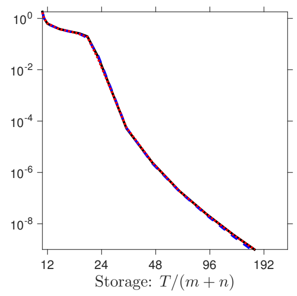
(C) `LowRankLowNoise`



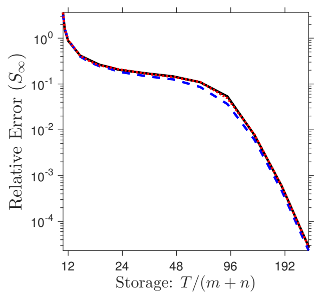
(D) `PolyDecaySlow`



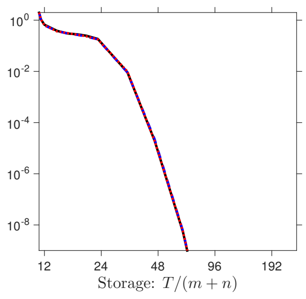
(E) `PolyDecayMed`



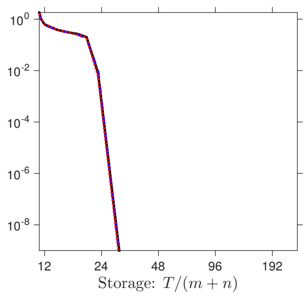
(F) `PolyDecayFast`



(G) `ExpDecaySlow`

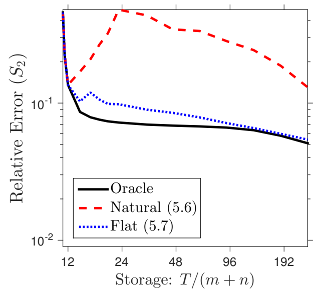


(H) `ExpDecayMed`

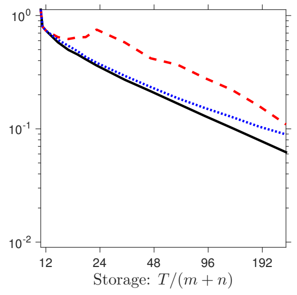


(I) `ExpDecayFast`

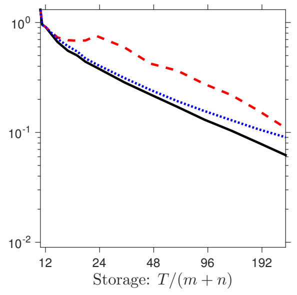
FIG. SM5: **Insensitivity of proposed method to the dimension reduction map.** (Effective rank $R = 20$, approximation rank $r = 10$, Schatten ∞-norm.) We compare the oracle performance of the proposed fixed-rank approximation (2.10) implemented with Gaussian, SSRFT, or sparse dimension reduction maps. See subsection 6.4 for details.



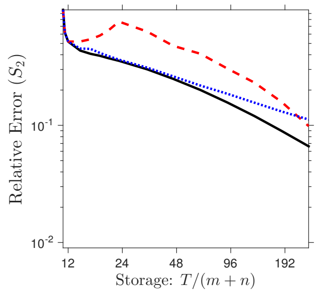
(A) `LowRankHiNoise`



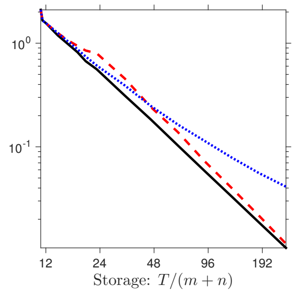
(B) `LowRankMedNoise`



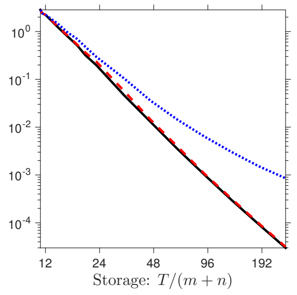
(C) `LowRankLowNoise`



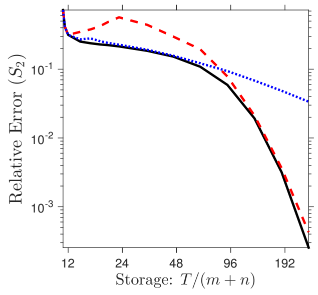
(D) `PolyDecaySlow`



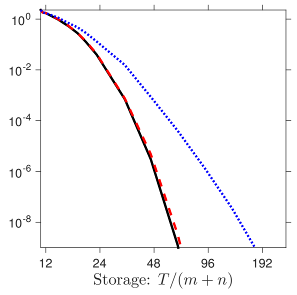
(E) `PolyDecayMed`



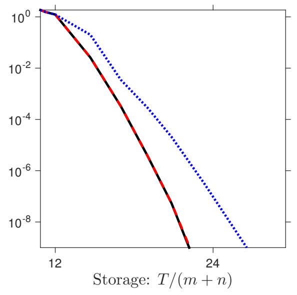
(F) `PolyDecayFast`



(G) `ExpDecaySlow`

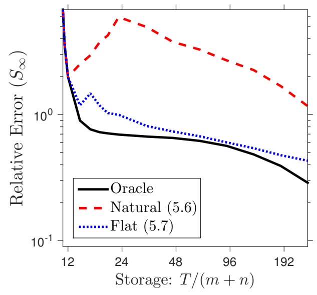


(H) `ExpDecayMed`

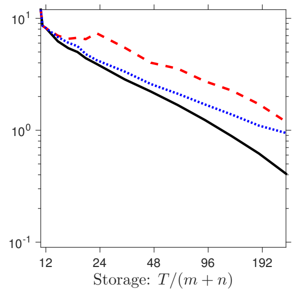


(I) `ExpDecayFast`

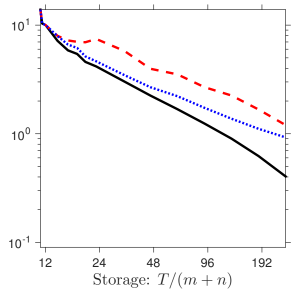
FIG. SM6: **Relative error for proposed method with *a priori* parameters.** (Gaussian maps, effective rank $R = 5$, approximation rank $r = 10$, Schatten 2-norm.) We compare the oracle performance of the proposed fixed-rank approximation (2.10) with its performance at theoretically justified parameter values. See subsection 6.5 for details.



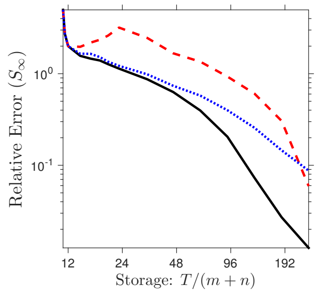
(A) `LowRankHiNoise`



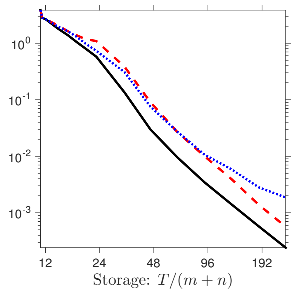
(B) `LowRankMedNoise`



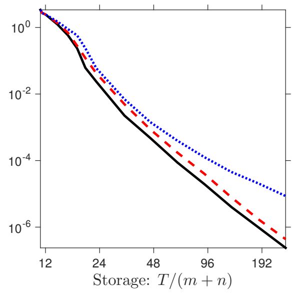
(C) `LowRankLowNoise`



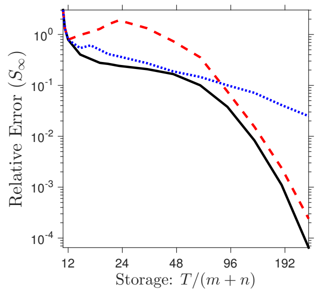
(D) `PolyDecaySlow`



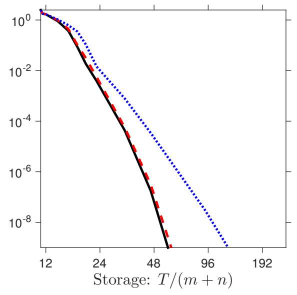
(E) `PolyDecayMed`



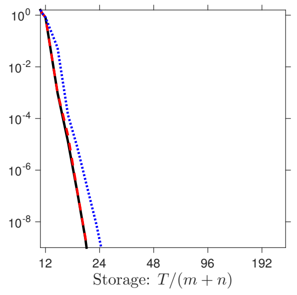
(F) `PolyDecayFast`



(G) `ExpDecaySlow`

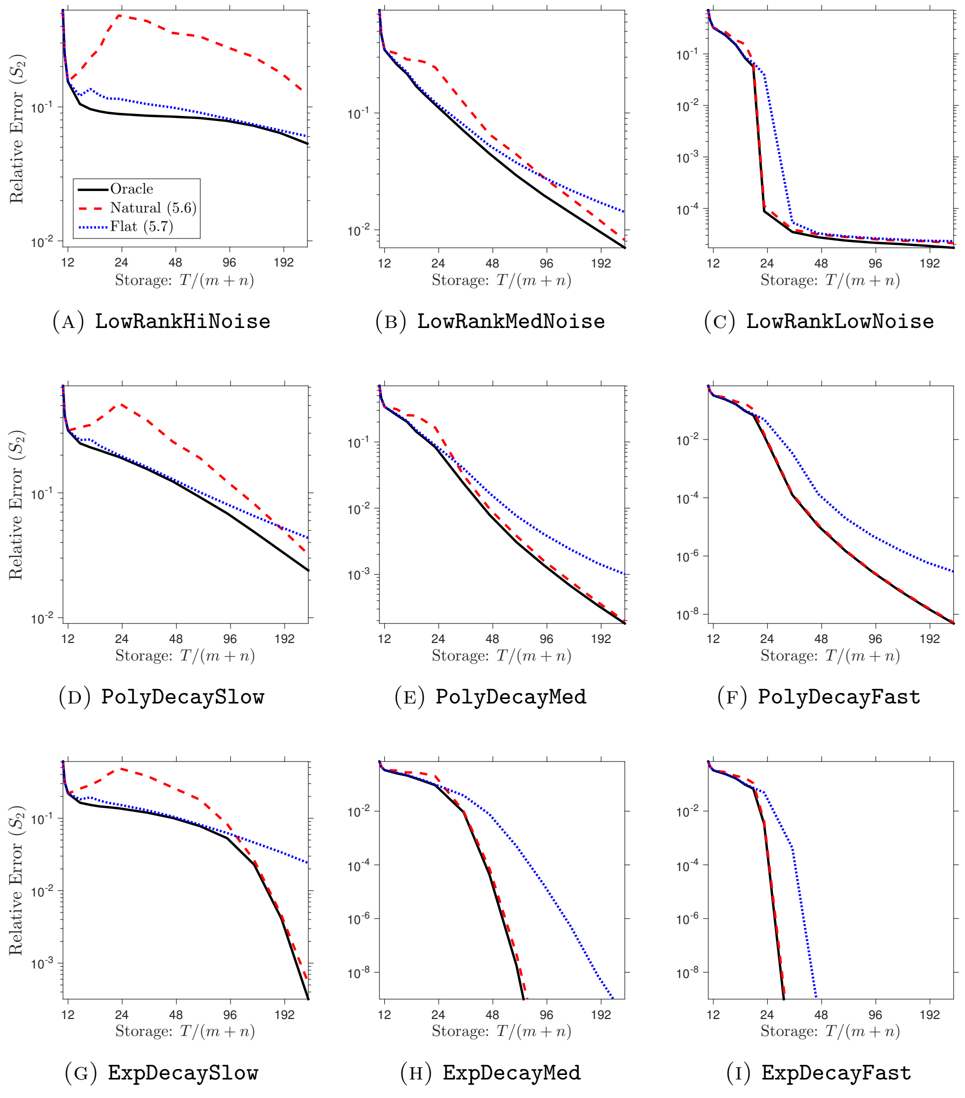


(H) `ExpDecayMed`



(I) `ExpDecayFast`

FIG. SM7: **Relative error for proposed method with *a priori* parameters.** (Gaussian maps, effective rank $R = 5$, approximation rank $r = 10$, Schatten ∞-norm.) We compare the oracle performance of the proposed fixed-rank approximation (2.10) with its performance at theoretically justified parameter values. See subsection 6.5 for details.



(A) `LowRankHiNoise` (B) `LowRankMedNoise` (C) `LowRankLowNoise`

(D) `PolyDecaySlow` (E) `PolyDecayMed` (F) `PolyDecayFast`

(G) `ExpDecaySlow` (H) `ExpDecayMed` (I) `ExpDecayFast`

FIG. SM8: **Relative error for proposed method with *a priori* parameters.** (Gaussian maps, effective rank $R = 20$, approximation rank $r = 10$, Schatten 2-norm.) We compare the oracle performance of the proposed fixed-rank approximation (2.10) with its performance at theoretically justified parameter values. See subsection 6.5 for details.

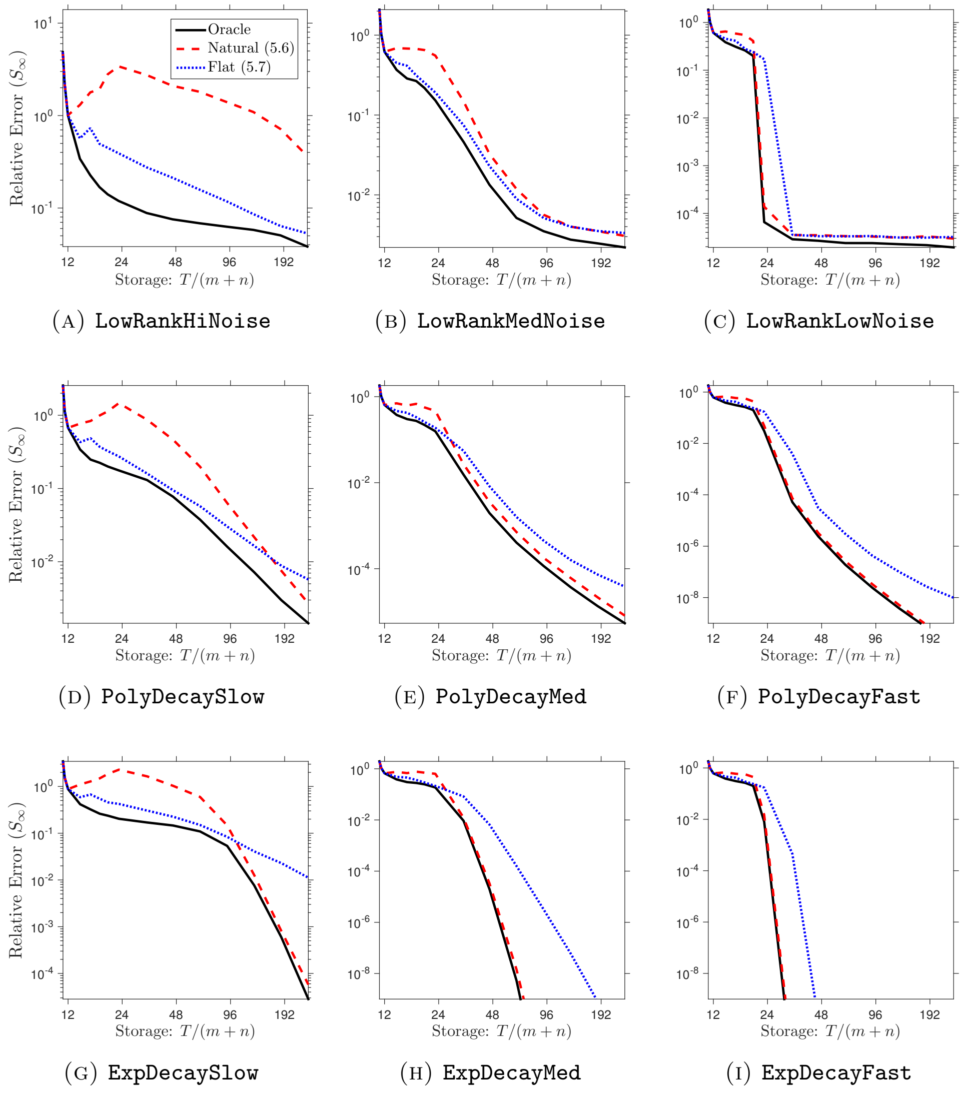
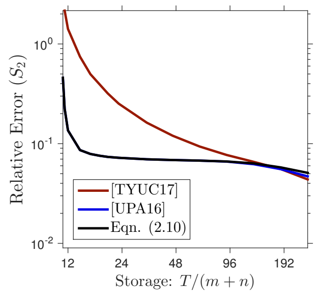
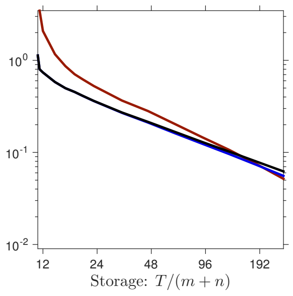


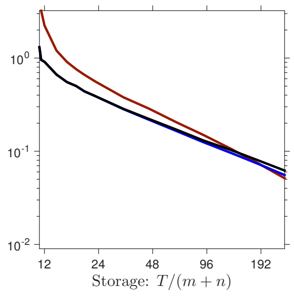
FIG. SM9: **Relative error for proposed method with *a priori* parameters.** (Gaussian maps, effective rank $R = 20$, approximation rank $r = 10$, Schatten ∞-norm.) We compare the oracle performance of the proposed fixed-rank approximation (2.10) with its performance at theoretically justified parameter values. See subsection 6.5 for details.



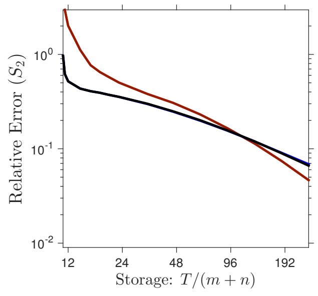
(A) `LowRankHiNoise`



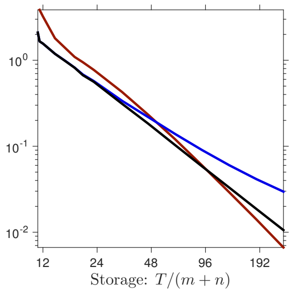
(B) `LowRankMedNoise`



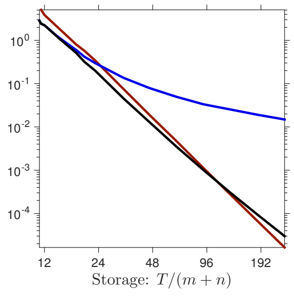
(C) `LowRankLowNoise`



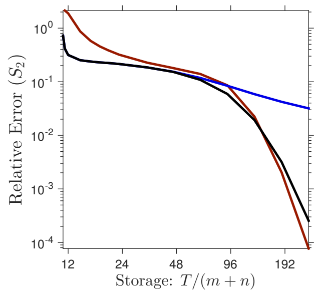
(D) `PolyDecaySlow`



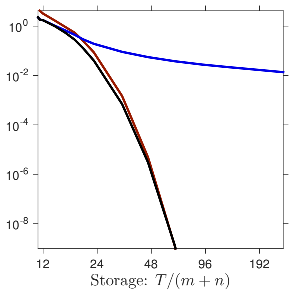
(E) `PolyDecayMed`



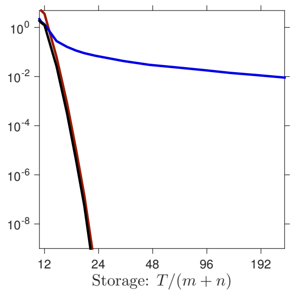
(F) `PolyDecayFast`



(G) `ExpDecaySlow`

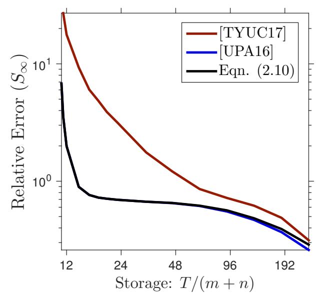


(H) `ExpDecayMed`

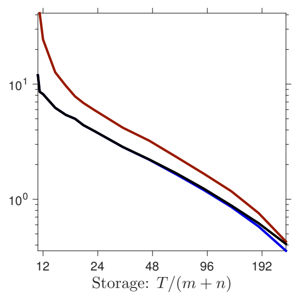


(I) `ExpDecayFast`

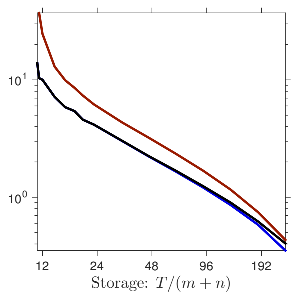
FIG. SM10: **Comparison of reconstruction formulas: Synthetic examples.** (Gaussian maps, effective rank $R = 5$, approximation rank $r = 10$, Schatten 2-norm.) We compare the oracle error achieved by the proposed fixed-rank approximation (2.10) against methods (6.1) and (6.2) from the literature. See subsection 6.2.2 for details.



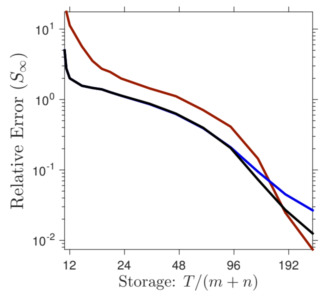
(A) `LowRankHiNoise`



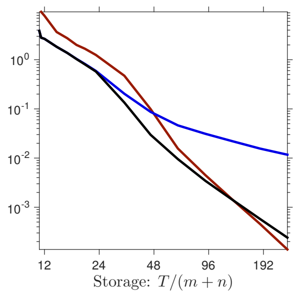
(B) `LowRankMedNoise`



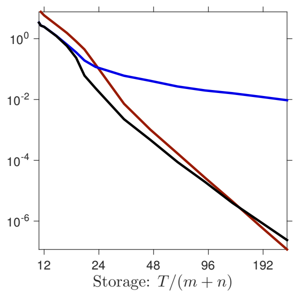
(C) `LowRankLowNoise`



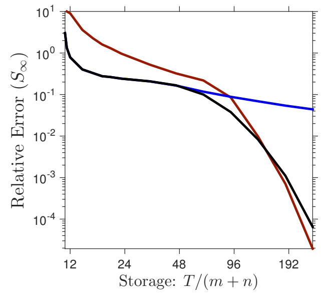
(D) `PolyDecaySlow`



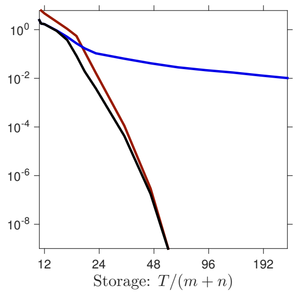
(E) `PolyDecayMed`



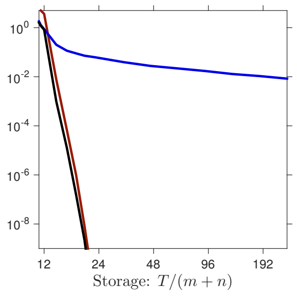
(F) `PolyDecayFast`



(G) `ExpDecaySlow`

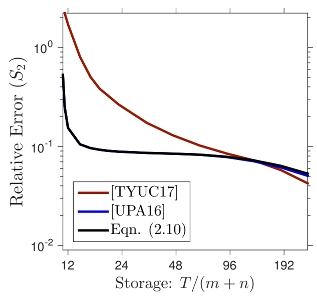


(H) `ExpDecayMed`

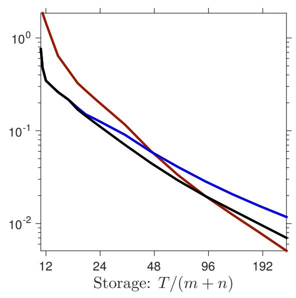


(I) `ExpDecayFast`

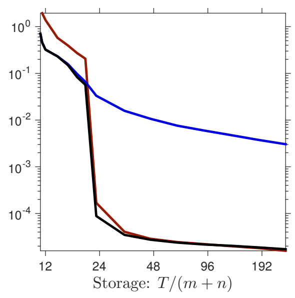
FIG. SM11: **Comparison of reconstruction formulas: Synthetic examples.** (Gaussian maps, effective rank $R = 5$, approximation rank $r = 10$, Schatten ∞-norm.) We compare the oracle error achieved by the proposed fixed-rank approximation (2.10) against methods (6.1) and (6.2) from the literature. See subsection 6.2.2 for details.



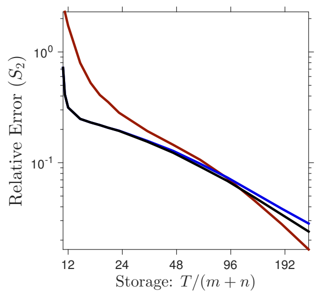
(A) `LowRankHiNoise`



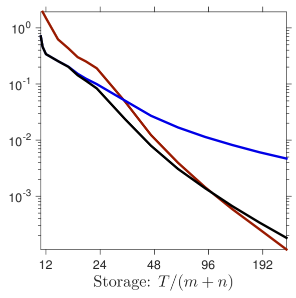
(B) `LowRankMedNoise`



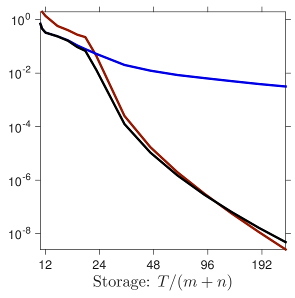
(C) `LowRankLowNoise`



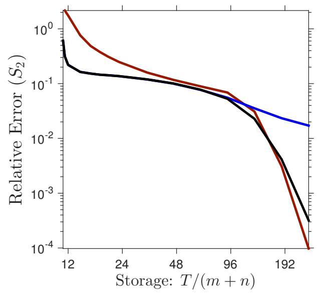
(D) `PolyDecaySlow`



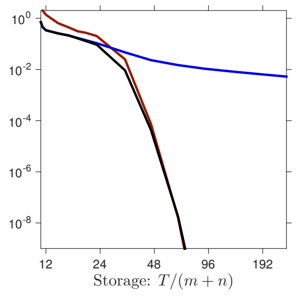
(E) `PolyDecayMed`



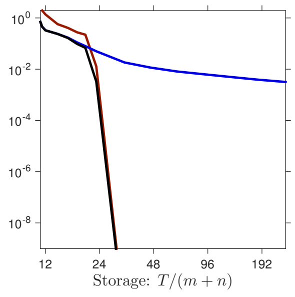
(F) `PolyDecayFast`



(G) `ExpDecaySlow`

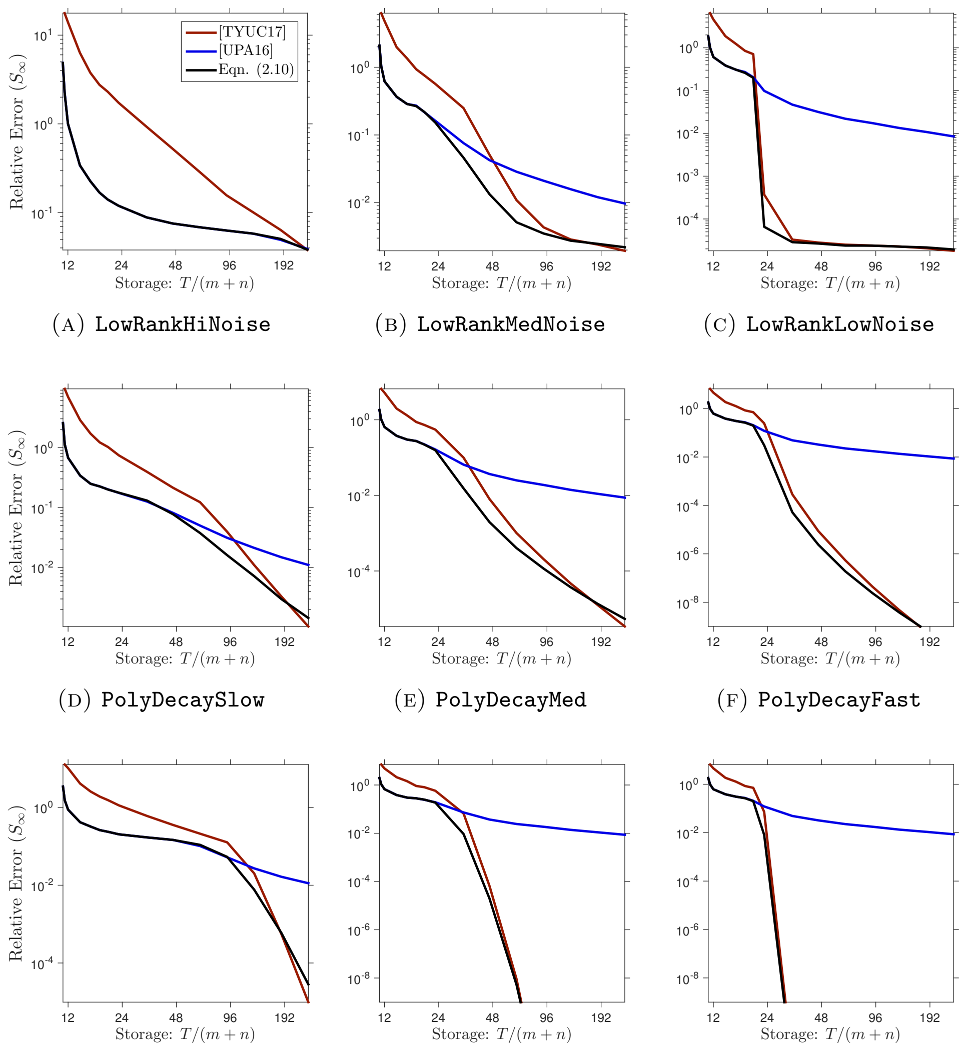


(H) `ExpDecayMed`



(I) `ExpDecayFast`

FIG. SM12: **Comparison of reconstruction formulas: Synthetic examples.** (Gaussian maps, effective rank $R = 20$, approximation rank $r = 10$, Schatten 2-norm.) We compare the oracle error achieved by the proposed fixed-rank approximation (2.10) against methods (6.1) and (6.2) from the literature. See subsection 6.2.2 for details.



(A) `LowRankHiNoise` (B) `LowRankMedNoise` (C) `LowRankLowNoise`

(D) `PolyDecaySlow` (E) `PolyDecayMed` (F) `PolyDecayFast`

(G) `ExpDecaySlow` (H) `ExpDecayMed` (I) `ExpDecayFast`

FIG. SM13: **Comparison of reconstruction formulas: Synthetic examples.** (Gaussian maps, effective rank $R = 20$, approximation rank $r = 10$, Schatten ∞-norm.) We compare the oracle error achieved by the proposed fixed-rank approximation (2.10) against methods (6.1) and (6.2) from the literature. See subsection 6.2.2 for details.

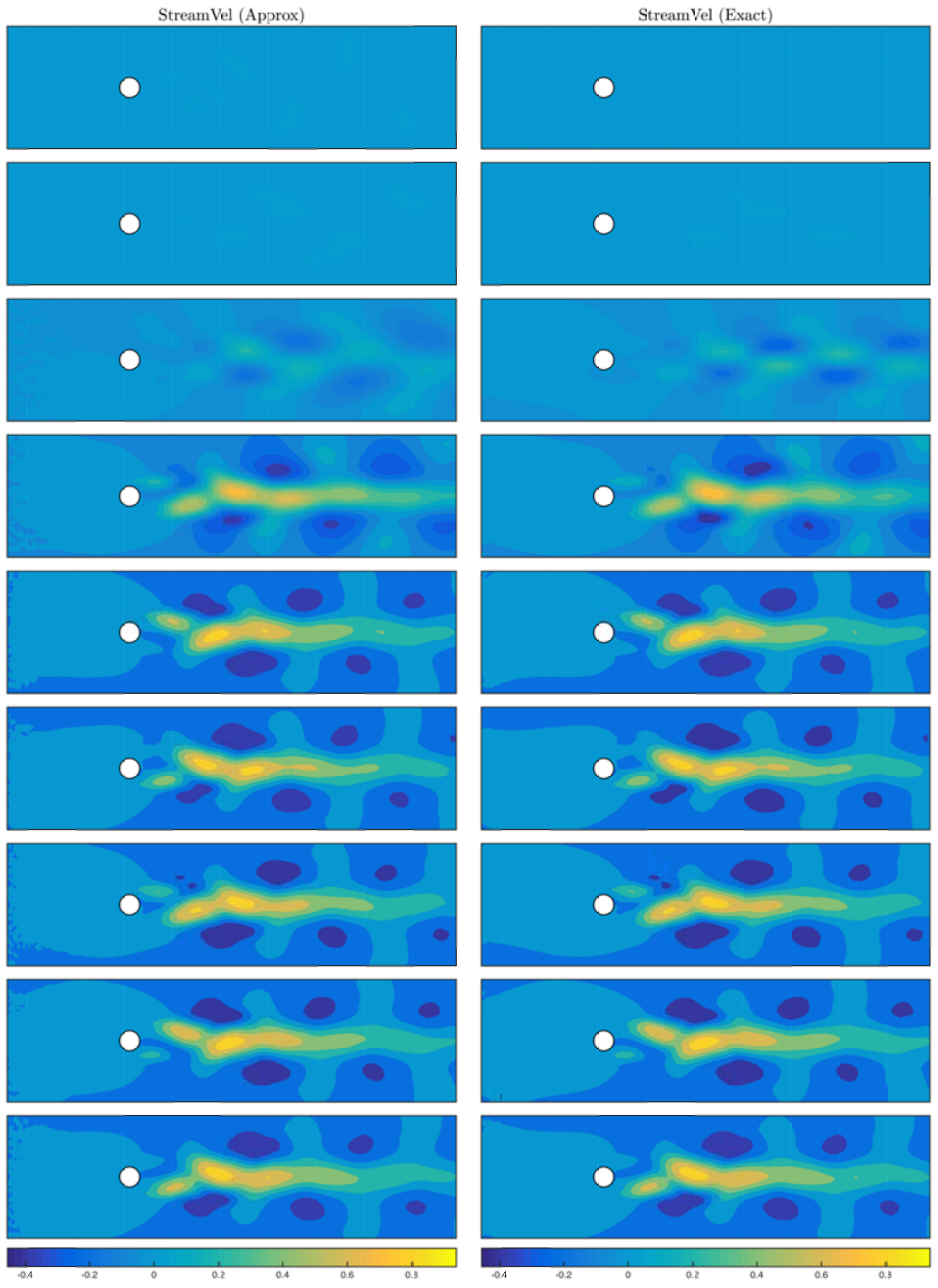


FIG. SM14: **Approximation of `StreamVel`.** (Sparse maps, approximation rank $r = 10$, storage budget $T = 48(m + n)$.) The columns of the matrix `StreamVel` describe the fluctuations of the streamwise velocity field about its mean value as a function of time. From top to bottom, the panels show columns $1, 501, 1001, 1501, 2001, 2501, 3001, 3501, 4001$. The **left-hand side** displays the approximation of the flow field, and the **right-hand side** displays the exact flow field. The heatmap indicates the magnitude of the fluctuation. See subsection 6.8 for details.