SIAM J. SCI. COMPUT.
Vol. 46, No. 1, pp. A508–A528

EFFICIENT ERROR AND VARIANCE ESTIMATION FOR RANDOMIZED MATRIX COMPUTATIONS*

ETHAN N. EPPERLY† AND JOEL A. TROPP†

Abstract. Randomized matrix algorithms have become workhorse tools in scientific computing and machine learning. To use these algorithms safely in applications, they should be coupled with posterior error estimates to assess the quality of the output. To meet this need, this paper proposes two diagnostics: a leave-one-out error estimator for randomized low-rank approximations and a jackknife resampling method to estimate the variance of the output of a randomized matrix computation. Both of these diagnostics are rapid to compute for randomized low-rank approximation algorithms such as the randomized SVD and randomized Nyström approximation, and they provide useful information that can be used to assess the quality of the computed output and guide algorithmic parameter choices.

Key words. jackknife resampling, low-rank approximation, error estimation, randomized algorithms

MSC codes. 62F40, 65F55, 68W20

DOI. 10.1137/23M1558537

1. Introduction.

In recent years, randomness has become an essential tool in the design of matrix algorithms [2, 7, 12, 13, 29], with randomized algorithms proving especially effective for matrix low-rank approximation. To use these algorithms safely in practice, they should be supported by *posterior error estimates* and other quality metrics that inform the user about the accuracy of the computational output.

This paper presents two diagnostic tools for randomized matrix computations:

- First, we provide a *leave-one-out posterior estimate* for the error $\|\boldsymbol{A} - \boldsymbol{X}\|_{\mathrm{F}}$ for a low-rank approximation $\boldsymbol{X}$ to a matrix $\boldsymbol{A}$ produced by randomized algorithms such as the randomized SVD or randomized Nyström aproximation.
- Second, we present a jackknife method for estimating the *variance* $\mathrm{Var}(\boldsymbol{X}) := \mathbb{E}\,\|\boldsymbol{X} - \mathbb{E}\,\boldsymbol{X}\|_{\mathrm{F}}^2$ of the matrix output $\boldsymbol{X}$ of a randomized algorithm. The variance is a useful diagnostic: If the computation is sensitive to the randomness used by the algorithm, the computed output should be treated with suspicion.

By using novel downdating formulas (see (4.1) and (4.3) below), we can rapidly compute both of these estimators for widely used low-rank approximation methods such as the randomized SVD and randomized Nyström approximation. Our diagnostics are also *sample-efficient* in the sense that they require no additional matrix-vector products or other information beyond that used in the original algorithm. The speed and efficiency of these diagnostics make them compelling additions to workflows involving randomized matrix computation.

*Submitted to the journal's Numerical Algorithms for Scientific Computing section March 13, 2023; accepted for publication (in revised form) October 30, 2023; published electronically February 8, 2024.

https://doi.org/10.1137/23M1558537

Funding: This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Department of Energy Computational Science Graduate Fellowship under award DE-SC0021110. The second author was supported in part by ONR awards N00014-17-1-2146 and N00014-18-1-2363 and NSF FRG award 1952777.

†Department of Computing and Mathematical Sciences, California Institute of Technology, Pasadena, CA 91125 USA (eepperly@caltech.edu, jtropp@cms.caltech.edu).

1.1. Leave-one-out error estimation. We begin by motivating our first diagnostic, a leave-one-out estimator for the error of a low-rank matrix approximation. For concreteness, we introduce this estimate in the context of Nyström approximation of positive-semidefinite (psd) matrices; see section 2 for a more general setting.

Suppose we want to approximate a psd matrix $\boldsymbol{A} \in \mathbb{R}^{d\times d}$, which we can only access by the matrix-vector product operation $\boldsymbol{\omega} \mapsto \boldsymbol{A}\boldsymbol{\omega}$. Using the matrix-vector product operation, we can compute the matrix-matrix product $\boldsymbol{A}\boldsymbol{\Phi}$ with a (random) matrix $\boldsymbol{\Phi} \in \mathbb{R}^{d\times s}$ with s columns and form the Nyström approximation [6, 24, 28]

$$\boldsymbol{A}\langle\boldsymbol{\Phi}\rangle := \boldsymbol{A}\boldsymbol{\Phi}\left(\boldsymbol{\Phi}^*\boldsymbol{A}\boldsymbol{\Phi}\right)^{\dagger}(\boldsymbol{A}\boldsymbol{\Phi})^*. \tag{1.1}$$

Here, * denotes the transpose and † denotes the Moore–Penrose pseudoinverse. The result is a psd, rank-s approximation $\boldsymbol{A}\langle\boldsymbol{\Phi}\rangle$ to the matrix $\boldsymbol{A}$. We will generate $\boldsymbol{\Phi}$ by applying $q \geq 0$ steps of subspace iteration to a random test matrix $\boldsymbol{\Omega}$

$$\boldsymbol{\Phi} = \boldsymbol{A}^q\boldsymbol{\Omega}, \tag{1.2}$$

where $\boldsymbol{\Omega}$ is populated with statistically independent standard Gaussian entries. The quality of the Nyström approximation improves with a higher approximation rank s or number of subspace iteration steps q. Using the forthcoming Algorithm 4.1, we can compute $\boldsymbol{A}\langle\boldsymbol{\Phi}\rangle$ in the form of a compact eigenvalue decomposition:

$$\boldsymbol{A}\langle\boldsymbol{\Phi}\rangle = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^*, \tag{1.3}$$

where $\boldsymbol{V} \in \mathbb{R}^{d\times s}$ has orthonormal columns and $\boldsymbol{\Lambda} \in \mathbb{R}_+^{s\times s}$ is diagonal. The cost of this procedure is $\mathcal{O}(qs)$ matrix-vector products with $\boldsymbol{A}$ and $\mathcal{O}(ds^2)$ additional operations.

To use the Nyström approximation with confidence in applications and to guide the choice of parameters s and q, we need to understand the accuracy of the approximation $\boldsymbol{A} \approx \boldsymbol{A}\langle\boldsymbol{\Phi}\rangle$. This motivates our question:

> What is the most efficient way to estimate the error $\|\boldsymbol{A} - \boldsymbol{A}\langle\boldsymbol{\Phi}\rangle\|_{\mathrm{F}}$?

Inspired by this question, this article proposes the *leave-one-out error estimator,* which provides an estimate of $\|\boldsymbol{A} - \boldsymbol{A}\langle\boldsymbol{\Phi}\rangle\|_{\mathrm{F}}$ using only the information already collected from $\boldsymbol{A}$ to form the Nyström approximation.

The leave-one-out estimator is built by recomputing the Nyström approximation using subsamples of the columns of the matrix $\boldsymbol{\Omega}$. We can regard the Nyström approximation as a function of the test matrix $\boldsymbol{\Omega}$:

$$\boldsymbol{\Omega} \mapsto \boldsymbol{X} = \boldsymbol{X}(\boldsymbol{\Omega}) := \boldsymbol{A}\langle\boldsymbol{A}^q\boldsymbol{\Omega}\rangle.$$

Let $\boldsymbol{\Omega}^{(j)}$ denote $\boldsymbol{\Omega}$ without its jth column. Introduce *replicates* $\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(s)}$ by recomputing $\boldsymbol{X}$ with each column of $\boldsymbol{\Omega}$ left out in turn:

$$\boldsymbol{X}^{(j)} = \boldsymbol{X}\big(\boldsymbol{\Omega}^{(j)}\big) \quad \text{for } j = 1, 2, \ldots, s.$$

Letting $\boldsymbol{\omega}_j$ denote the jth column of $\boldsymbol{\Omega}$ and $\|\cdot\|$ denote the Euclidean norm, we define the *leave-one-out error estimator*

$$\widehat{\mathrm{Err}}^2(\boldsymbol{X}, \boldsymbol{A}) := \frac{1}{s}\sum_{i=1}^{s}\big\|\big(\boldsymbol{A} - \boldsymbol{X}^{(j)}\big)\boldsymbol{\omega}_j\big\|^2.$$

As we show in Theorem 2.1, this error estimator is an *unbiased estimator for the mean-square error of the rank-*$(s-1)$ *Nyström approximation*:

$$\mathbb{E}\,\widehat{\mathrm{Err}}^2(\boldsymbol{X},\boldsymbol{A}) = \mathbb{E}\left\|\boldsymbol{A}-\boldsymbol{X}(\boldsymbol{\Omega}^{(s)})\right\|_{\mathrm{F}}^2.$$

Once the Nyström approximation has been computed, $\widehat{\mathrm{Err}}(\boldsymbol{X},\boldsymbol{A})$ is cheap to form, requiring at most $\mathcal{O}(ds^2)$ additional operations (and only $\mathcal{O}(s^3)$ operations if $q=0$). See subsection 4.1 for details. The error estimate $\widehat{\mathrm{Err}}(\boldsymbol{X},\boldsymbol{A})$ requires no information about $\boldsymbol{A}$ beyond what is required to form the approximation $\boldsymbol{X}$.

A good point of comparison for the leave-one-out error estimator is provided by the Girard–Hutchinson norm estimate [12, section 4.8]

$$\widehat{\mathrm{Err}}^2_{\mathrm{GH}}(\boldsymbol{X},\boldsymbol{A}) = \frac{1}{t}\sum_{i=1}^{t}\|(\boldsymbol{A}-\boldsymbol{X})\boldsymbol{\nu}_i\|^2 \approx \|\boldsymbol{A}-\boldsymbol{X}\|_{\mathrm{F}}^2\,. \tag{1.4}$$

Here, $\boldsymbol{\nu}_1,\dots,\boldsymbol{\nu}_t$ are independent standard Gaussian test vectors. This estimator requires t matrix-vector products with $\boldsymbol{A}$ (beyond those used to form the approximation $\boldsymbol{X}$), and the norm estimate becomes more accurate with larger t. The Girard–Hutchinson norm estimator is equivalent to the Girard–Hutchinson trace estimator [12, section 4.2] applied to $(\boldsymbol{A}-\boldsymbol{X})^*(\boldsymbol{A}-\boldsymbol{X})$ and served as our inspiration for the leave-one-out error estimator. The leave-one-out estimator improves on the Girard–Hutchinson estimator as it requires no additional matrix-vector products with $\boldsymbol{A}$ to compute. In addition, the quality of the leave-one-out estimator automatically improves when the approximation rank s increases, whereas the Girard–Hutchinson estimate only improves by using more matrix-vector products.

Figure 1 demonstrates the leave-one-out error estimator. In this figure, we apply single-pass Nyström approximation ($q=0$) to approximate psd kernel matrix $\boldsymbol{A}\in\mathbb{R}^{10^4\times 10^4}$ formed from a random subsample of 10^4 points from the QM9 dataset [18, 19] using approximation ranks $5\le s\le 150$. In the left panel, we plot the mean error

$$\mathrm{Err}(\boldsymbol{X},\boldsymbol{A}) := \mathbb{E}\,\|\boldsymbol{A}-\boldsymbol{X}\|_{\mathrm{F}} \tag{1.5}$$

and the leave-one-out error estimate $\widehat{\mathrm{Err}}(\boldsymbol{X},\boldsymbol{A})$, estimated using 1000 trials. We see that the estimate $\widehat{\mathrm{Err}}(\boldsymbol{X},\boldsymbol{A})$ closely tracks the true error. Moreover, the error estimate $\widehat{\mathrm{Err}}(\boldsymbol{X},\boldsymbol{A})$ is fast to compute, with the error estimate taking less than 1% of the total runtime to form. The right panel compares the mean relative error

$$\text{mean relative error} = \mathbb{E}\left[\frac{\left|\|\boldsymbol{A}-\boldsymbol{X}\|_{\mathrm{F}}-\mathrm{Est}\right|}{\|\boldsymbol{A}-\boldsymbol{X}\|_{\mathrm{F}}}\right],\quad \mathrm{Est}\in\left\{\widehat{\mathrm{Err}}(\boldsymbol{X},\boldsymbol{A}),\widehat{\mathrm{Err}}_{\mathrm{GH}}(\boldsymbol{X},\boldsymbol{A})\right\}$$

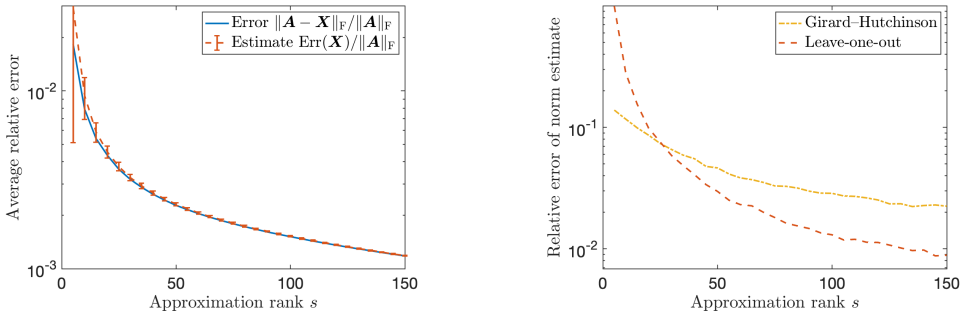


FIG. 1. ***Leave-one-out error estimator.*** *Approximation error* $\mathrm{Err}(\boldsymbol{X},\boldsymbol{A})/\|\boldsymbol{A}\|_{\mathrm{F}}$ *and normalized error estimate* $\widehat{\mathrm{Err}}(\boldsymbol{X},\boldsymbol{A})/\|\boldsymbol{A}\|_{\mathrm{F}}$ *(left) and relative error for the leave-one-out and Girard–Hutchinson error estimators (right) for Nyström approximation* $\boldsymbol{X}$ *to a psd kernel matrix* $\boldsymbol{A}\in\mathbb{R}^{10^4\times 10^4}$. *Error and error estimate were computed by an average of* 1000 *trials. Error bars show one standard deviation.*

for both the leave-one-out and Girard–Hutchinson error estimators. Following [25, section 7.9], we use $t = 10$ matrix-vector products for the Girard–Hutchinson estimator. For $s \geq 25$, the leave-one-out estimator is more accurate than Girard–Hutchinson and, for all values of s, the leave-one-out estimator is cheaper to form than Girard–Hutchinson, requiring just $\mathcal{O}(s^3)$ operations and no additional matrix-vector products.

1.2. Matrix jackknife motivating example: Spectral clustering. Randomized low-rank approximations can also be used for *spectral computations* (i.e., to approximate eigenvalues, eigenvectors, singular values, etc.). In this case, the low-rank approximation error $\|\boldsymbol{A} - \boldsymbol{X}\|_{\mathrm{F}}$ may only provide indirect information about the accuracy of the computation. For situations such as this, we propose a *matrix jackknife variance estimate* as a diagnostic tool. In this section, we illustrate the value of this matrix jackknife approach in a spectral clustering application, before introducing the method in generality in section 3.

1.2.1. Nyström-accelerated spectral clustering. Spectral clustering [27] is an algorithm that uses eigenvectors to assign data points, say, $\boldsymbol{c}_1, \dots, \boldsymbol{c}_d \in \mathbb{R}^m$, into groups. To measure similarity between points, we employ a nonnegative, positive definite kernel function $\kappa : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}_+$. One popular choice is the square-exponential kernel

$$\kappa(\boldsymbol{c}, \boldsymbol{c}') = \exp\left(-\frac{\|\boldsymbol{c} - \boldsymbol{c}'\|^2}{2\sigma^2}\right). \tag{1.6}$$

To cluster the data points into groups, we perform the following steps:

1. Form the kernel matrix $\boldsymbol{K}$ with entries $k_{ij} = \kappa(\boldsymbol{c}_i, \boldsymbol{c}_j)$.
2. Assemble the diagonal matrix $\boldsymbol{D} = \operatorname{diag}\big(\sum_{j=1}^{d} k_{ij} : i = 1, \dots, d\big)$.
3. Compute the n_{dim} dominant eigenvectors $\boldsymbol{U}$ of $\boldsymbol{A} := \boldsymbol{D}^{-1/2}\boldsymbol{K}\boldsymbol{D}^{-1/2}$.
4. Set $\boldsymbol{W} := \boldsymbol{D}^{-1/2}\boldsymbol{U}$.
5. Apply a general-purpose clustering algorithm, such as k-means [1] with n_{cen} centers, to the rows of $\boldsymbol{W}$.

Parameters n_{dim} and n_{cen} set the clustering space dimension and number of clusters.

If one uses direct methods for the eigenvalue problem, the cost of spectral clustering is dominated by the $\mathcal{O}(d^3)$ cost for the eigenvector calculation in step 3. We can accelerate spectral clustering by using Nyström approximation (1.1). The modification is simple: Use the n_{dim} dominant eigenvectors of the Nyström approximation, accessible from the eigendecomposition (1.3), in place of the eigenvectors of $\boldsymbol{A}$.

1.2.2. Variance estimation for spectral clustering. As we refine the approximation by increasing s, the approximate eigenvectors $\widehat{\boldsymbol{U}}$ will converge to the true eigenvectors $\boldsymbol{U}$ (provided $\boldsymbol{U}$ is unique). But how do we know when we have taken s large enough? Our guiding principle is:

> In order to trust the answer provided by a randomized algorithm, the output should be insensitive to the randomness used by the algorithm.

The *variance* of the matrix output $\boldsymbol{X}$ of a randomized algorithm, defined as

$$\operatorname{Var}(\boldsymbol{X}) := \mathbb{E}\,\|\boldsymbol{X} - \mathbb{E}\,\boldsymbol{X}\|_{\mathrm{F}}^2, \tag{1.7}$$

provides a quantitative measurement of the sensitivity of the algorithmic output to randomness used by the algorithm. In the context of spectral clustering, we can use a variance estimate to guide our choice of the rank s.

To understand the sensitivity of Nyström-accelerated spectral clustering to randomness in the algorithm, we need to specify a target matrix $\boldsymbol{X}$ for variance estimation. The input to k-means clustering are the *coordinates*

$$\widehat{\boldsymbol{W}} := \boldsymbol{D}^{-1/2}\widehat{\boldsymbol{U}}.$$

To respect the invariance of k-means to scaling and rotation of the coordinates, our target for variance estimation will be

$$\boldsymbol{X} = \widehat{\boldsymbol{W}}\widehat{\boldsymbol{W}}^* / \big\|\widehat{\boldsymbol{W}}\widehat{\boldsymbol{W}}^*\big\|_{\mathrm{F}}.$$

1.2.3. Jackknife variance estimation. Jackknife variance estimation is similar to the leave-one-out error estimator in that we use replicates $\boldsymbol{X}^{(1)}, \dots, \boldsymbol{X}^{(s)}$ recomputed by successively leaving out columns of the random test matrix $\boldsymbol{\Omega}$. As before, we view the target $\boldsymbol{X}$ as a function

$$\boldsymbol{X} = \boldsymbol{X}(\boldsymbol{\Omega})$$

of the test matrix $\boldsymbol{\Omega}$ defining the Nyström approximation by (1.1) and (1.2). Form *jackknife replicates* $\boldsymbol{X}^{(j)}$ and their average $\boldsymbol{X}^{(\cdot)}$ via

$$\boldsymbol{X}^{(j)} = \boldsymbol{X}\left(\boldsymbol{\Omega}^{(j)}\right) \quad \text{for } j = 1, \dots, s \quad \text{and} \quad \boldsymbol{X}^{(\cdot)} := \frac{1}{s}\sum_{j=1}^{s} \boldsymbol{X}^{(j)},$$

where $\boldsymbol{\Omega}^{(j)}$ again denotes $\boldsymbol{\Omega}$ without its jth column. The jackknife estimate for $\mathrm{Var}(\boldsymbol{X})$ is

$$\mathrm{Jack}^2(\boldsymbol{X}) := \sum_{j=1}^{s} \left\|\boldsymbol{X}^{(j)} - \boldsymbol{X}^{(\cdot)}\right\|_{\mathrm{F}}^2.$$

Guarantees for this estimator are provided in Theorem 3.2. Algorithm SM1.1 and Program SM4 provide pseudocode and a MATLAB implementation of Nyström-accelerated spectral clustering with the jackknife estimate $\mathrm{Jack}(\boldsymbol{X})$. The cost of forming the jackknife estimate $\mathrm{Jack}(\boldsymbol{X})$ is $\mathcal{O}(s^2 d)$, much faster than the $\mathcal{O}(qsd^2)$ total cost of Nyström-accelerated spectral clustering.

1.2.4. Numerical example. To demonstrate the effectiveness of jackknife variance estimation for spectral clustering, we use the following experimental setup. Consider the task of separating the four letters `JACK` from a point cloud $\boldsymbol{c}_1, \dots, \boldsymbol{c}_{9426} \in \mathbb{R}^2$. For spectral clustering, use the square-exponential kernel (1.6) and parameters $n_{\mathrm{dim}} = n_{\mathrm{cen}} = 4$. For Nyström, use $q = 3$ steps of subspace iteration and test a range of approximation ranks $50 \le s \le 150$. For each value of s, we run 1000 trials and report the natural Monte Carlo estimate of the standard deviation

$$\mathrm{Std}(\boldsymbol{X}) = \left(\mathbb{E}\,\|\boldsymbol{X} - \mathbb{E}\,\boldsymbol{X}\|_{\mathrm{F}}^2\right)^{1/2}, \tag{1.8}$$

the mean and standard deviation for the *standard deviation* estimate $\mathrm{Jack}(\boldsymbol{X})$, the relative error $\mathrm{Err}(\widehat{\boldsymbol{A}}, \boldsymbol{A})/\|\boldsymbol{A}\|_{\mathrm{F}}$ for the Nyström approximation $\widehat{\boldsymbol{A}}$, and the empirical success probability for spectral clustering.

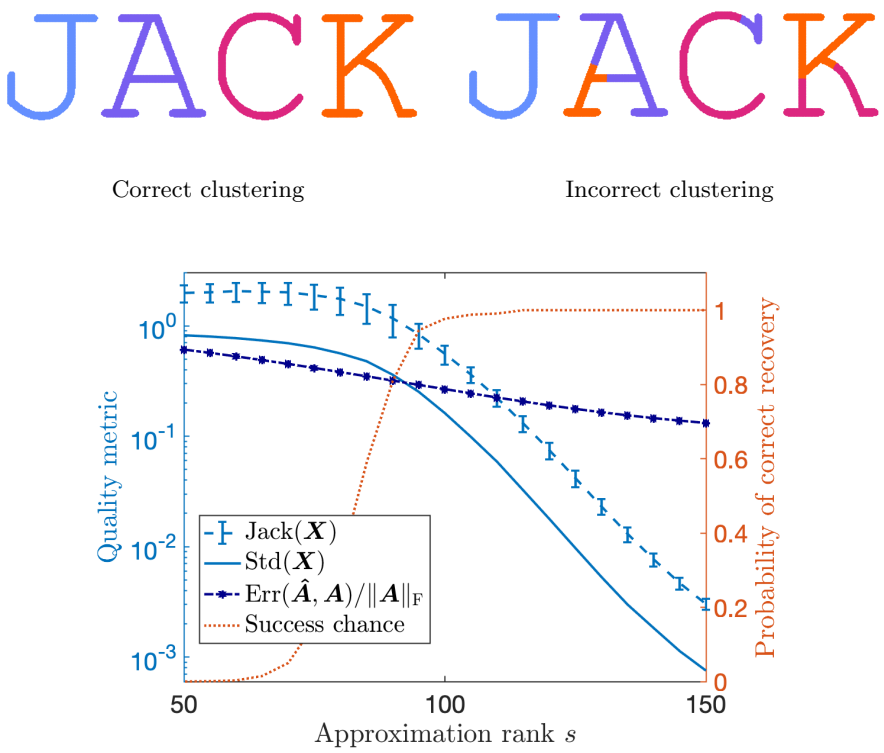


FIG. 2. ***Matrix jackknife for spectral clustering.*** *Top: Example of correct (left) and incorrect (right) clusterings. Bottom: Standard deviation* Std($\boldsymbol{X}$) *and its jackknife estimate* Jack($\boldsymbol{X}$) *(left axis) and success probability of spectral clustering (right axis) versus Nyström approximation rank* $50 \le s \le 150$.

Figure 2 shows the results. The estimate Jack($\boldsymbol{X}$) overestimates the standard deviation by a modest amount (a factor of five at most). The jackknife is not quantitatively sharp, but it is a reliable indicator of whether the variance is high or low.

The jackknife estimate Jack($\boldsymbol{X}$) can be used to determine whether the Nyström-based approximations to the eigenvectors are accurate enough for the clustering task. At rank $s = 50$, clustering is never performed correctly, and the jackknife estimate Jack($\boldsymbol{X}$) ≈ 2 is high. As the approximation rank s is increased, clustering begins to succeed with higher and higher probability and the jackknife estimate Jack($\boldsymbol{X}$) decreases, indicating reduced variability.

The success of the jackknife estimate Jack($\boldsymbol{X}$) should be contrasted with the failure of the Nyström error Err($\widehat{\boldsymbol{A}}, \boldsymbol{A}$) as a useful diagnostic for spectral clustering correctness. The Nyström error Err($\widehat{\boldsymbol{A}}, \boldsymbol{A}$) decreases at a steady rate as s is increased—unlike the jackknife estimate, the plot of Err($\widehat{\boldsymbol{A}}, \boldsymbol{A}$) does not show an inflection point indicating the transition between clustering failure and success.

On the basis of these experiments, we propose two possible uses for the jackknife estimate in a spectral clustering workflow:

- **User warning.** If the jackknife variance estimate is high, provide a warning to the user. This allows the user to determine and fix the problem for themselves by changing the Nyström parameters s, q or the spectral clustering parameters $n_{\mathrm{cen}}, n_{\mathrm{dim}}$.

- **Adaptive stopping.** Choose parameters s or q for the Nyström approximation adaptively at runtime by increasing these parameters until $\mathrm{Jack}(\boldsymbol{X})$ falls below a tolerance (e.g., 0.1).

These uses demonstrate the potential for jackknife variance estimation to be helpful in incorporating randomized low-rank approximation into general-purpose software.

1.2.5. Benefits of matrix jackknife variance estimation. The spectral clustering example demonstrates a number of virtues for matrix jackknife variance estimation:

- **Flexibility.** Matrix jackknife variance estimation can be applied to a very general target function $\boldsymbol{X}(\boldsymbol{\Omega})$ depending on a random test matrix $\boldsymbol{\Omega}$. This allows the jackknife to be applied to a wide array of randomized low-rank approximation algorithms and allows the user to design the variance estimation target for their application.
- **Efficiency.** By using optimized algorithms (section 4 and subsection SM1.2), computation of the jackknife variance estimate can be very fast. For instance, for the clustering problem, the $\mathcal{O}(s^2 d)$ cost of the jackknife estimate is dwarfed by the $\mathcal{O}(sd^2)$ cost of the clustering procedure. For $s = 150$, computing the jackknife variance estimate amounts to less than 3% of the total runtime.

1.3. Outline. Having introduced our two diagnostics, we present each in more generality; section 2 discusses leave-one-out error estimation and section 3 discusses the matrix jackknife. Section 4 discusses efficient computations for both of these diagnostics applied to the randomized SVD and Nyström approximation. Section 5 contains numerical experiments, and section 6 extends the matrix jackknife to higher Schatten norms ($p > 2$).

1.4. Notation. We work over the field $\mathbb{K} = \mathbb{R}$ or $\mathbb{K} = \mathbb{C}$. Notations *, †, and $\|\cdot\|_{\mathrm{F}}$ denote the conjugate transpose, Moore–Penrose pseudoinverse, and Frobenius norm. The expectation of a random variable X is denoted $\mathbb{E}\, X$, and its variance is defined as $\mathrm{Var}(X) := \mathbb{E}\,|X - \mathbb{E}\, X|^2$. We adopt the convention that nonlinear operators bind before the expectation; for example, $\mathbb{E}\, X^2 := \mathbb{E}\,(X^2)$. The variance of a random matrix is given by (1.7).

2. Leave-one-out error estimation for low-rank approximation. In this section, we present the leave-one-out error estimation technique introduced in subsection 1.1 for more general randomized matrix approximations.

2.1. The estimator. Let $\boldsymbol{A} \in \mathbb{K}^{d_1 \times d_2}$ be a matrix we seek to approximate by a randomized approximation $\boldsymbol{X}$. We are interested in a general class of algorithms which collect information about the matrix $\boldsymbol{A}$ by matrix-vector products

$$\boldsymbol{A}\boldsymbol{\omega}_1, \dots, \boldsymbol{A}\boldsymbol{\omega}_s$$

with random test vectors $\boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_s$. Many algorithms are defined for an arbitrary number of test vectors s, allowing us to construct error estimates by leaving out a test vector, resulting in an approximation $\boldsymbol{X}_{s-1}$ defined using only $s-1$ vectors. This motivates the following abstract setup:

- Let $\boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_s$ be independent and identically distributed (iid) random vectors in $\mathbb{K}^{d_2}$ that are *isotropic*: $\mathbb{E}\left[\boldsymbol{\omega}_j \boldsymbol{\omega}_j^*\right] = \mathbf{I}$.
- Let $\boldsymbol{X}$ denote one of two matrix estimators defined for s or $s-1$ inputs:

$$\boldsymbol{X} : (\mathbb{K}^{d_2})^s \to \mathbb{K}^{d_1 \times d_2} \quad \text{or} \quad \boldsymbol{X} : (\mathbb{K}^{d_2})^{s-1} \to \mathbb{K}^{d_1 \times d_2}.$$

- Define estimates $\boldsymbol{X}_s := \boldsymbol{X}(\boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_s)$ and $\boldsymbol{X}_{s-1} := \boldsymbol{X}(\boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_{s-1})$.

Examples of estimators which fit this description include randomized Nyström approximation, the randomized SVD [7], and randomized block Krylov iteration [14, 23].

We seek to approximate the mean-square error

$$\operatorname{MSE}(\boldsymbol{X}_{s-1}, \boldsymbol{A}) = \mathbb{E}\, \|\boldsymbol{A} - \boldsymbol{X}_{s-1}\|_{\mathrm{F}}^2$$

of the $(s-1)$-sample approximation $\boldsymbol{X}_{s-1}$ as a proxy for the mean-square error $\operatorname{MSE}(\boldsymbol{X}_s)$ of the s-sample approximation $\boldsymbol{X}_s$. Define the *leave-one-out mean-square error estimate*

$$\widehat{\operatorname{Err}}^2(\boldsymbol{X}_{s-1}, \boldsymbol{A}) := \frac{1}{s} \sum_{j=1}^{s} \left\| (\boldsymbol{A} - \boldsymbol{X}^{(j)}) \boldsymbol{\omega}_j \right\|^2, \tag{2.1}$$

where the replicates $\boldsymbol{X}^{(j)}$ are

$$\boldsymbol{X}^{(j)} = \boldsymbol{X}(\boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_{j-1}, \boldsymbol{\omega}_{j+1}, \dots, \boldsymbol{\omega}_s) \quad \text{for } j = 1, \dots, s.$$

This estimator is an unbiased estimator for $\operatorname{MSE}(\boldsymbol{X}_{s-1}, \boldsymbol{A})$.

THEOREM 2.1 (leave-one-out error estimator). *With the prevailing notation,*

$$\operatorname{MSE}(\boldsymbol{X}_{s-1}, \boldsymbol{A}) = \mathbb{E}\, \widehat{\operatorname{Err}}^2(\boldsymbol{X}_{s-1}, \boldsymbol{A}).$$

Proof. For each j, $\boldsymbol{X}^{(j)}$ and $\boldsymbol{\omega}_j$ are independent. Consequently, letting $\mathbb{E}_j$ denote an expectation over the randomness in $\boldsymbol{\omega}_j$ alone, we compute

$$\begin{aligned}
\mathbb{E}_j \left\| (\boldsymbol{A} - \boldsymbol{X}^{(j)}) \boldsymbol{\omega}_j \right\|^2 &= \mathbb{E}_j \left[\boldsymbol{\omega}_j^* (\boldsymbol{A} - \boldsymbol{X}^{(j)})^* (\boldsymbol{A} - \boldsymbol{X}^{(j)}) \boldsymbol{\omega}_j \right] \\
&= \mathbb{E}_j \operatorname{tr} \left[(\boldsymbol{A} - \boldsymbol{X}^{(j)})^* (\boldsymbol{A} - \boldsymbol{X}^{(j)}) \boldsymbol{\omega}_j \boldsymbol{\omega}_j^* \right] \\
&= \operatorname{tr} \left((\boldsymbol{A} - \boldsymbol{X}^{(j)})^* (\boldsymbol{A} - \boldsymbol{X}^{(j)}) \mathbb{E} \left[\boldsymbol{\omega}_j \boldsymbol{\omega}_j^* \right] \right) \\
&= \operatorname{tr} \left((\boldsymbol{A} - \boldsymbol{X}^{(j)})^* (\boldsymbol{A} - \boldsymbol{X}^{(j)}) \right) = \left\| \boldsymbol{A} - \boldsymbol{X}^{(j)} \right\|_{\mathrm{F}}^2.
\end{aligned}$$

The first line is an identity for the Frobenius norm, the second line is the cyclic property of the trace, the third line is the independence of $\boldsymbol{X}^{(j)}$ and $\boldsymbol{\omega}_j$, and the fourth line is the isotropic property of $\boldsymbol{\omega}_j$. Thus, by the tower property of conditional expectation, we conclude

$$\begin{aligned}
\mathbb{E}\, \widehat{\operatorname{Err}}^2(\boldsymbol{X}_{s-1}, \boldsymbol{A}) &= \frac{1}{s} \sum_{j=1}^{s} \mathbb{E} \left[\mathbb{E}_j \left\| (\boldsymbol{A} - \boldsymbol{X}^{(j)}) \boldsymbol{\omega}_j \right\|^2 \right] \\
&= \frac{1}{s} \sum_{j=1}^{s} \mathbb{E} \left\| (\boldsymbol{A} - \boldsymbol{X}^{(j)}) \boldsymbol{\omega}_j \right\|_{\mathrm{F}}^2 = \operatorname{MSE}(\boldsymbol{X}_{s-1}, \boldsymbol{A}).
\end{aligned}$$

This confirms the theorem. ∎

Numerical evidence for the quality of this error estimate is provided in Figures 1 and 3. With efficient algorithms (section 4), the leave-one-out error estimator is rapid to compute for the randomized SVD and Nyström approximation.

2.2. Alternatives. Two alternatives to the leave-one-out error estimator are worth mentioning. First, in many situations, it may be possible and computationally

cheap to simply compute the error $\|\boldsymbol{A} - \boldsymbol{X}\|_{\mathrm{F}}$ directly. For instance, if $\boldsymbol{X}$ is the approximation produced by the randomized SVD [7], then

$$\|\boldsymbol{A} - \boldsymbol{X}\|_{\mathrm{F}}^2 = \|\boldsymbol{A}\|_{\mathrm{F}}^2 - \|\boldsymbol{X}\|_{\mathrm{F}}^2,$$

which facilitates fast computation of the error if $\boldsymbol{A}$ is a dense or sparse matrix stored in memory. The leave-one-out error estimator should only be used if direct computation of the error is not possible or is too expensive. This is the case, for example, in the black-box setting where one has access to $\boldsymbol{A}$ only through the matrix-vector product $\boldsymbol{\omega} \mapsto \boldsymbol{A}\boldsymbol{\omega}$ and adjoint-vector product $\boldsymbol{\omega} \mapsto \boldsymbol{A}^*\boldsymbol{\omega}$ operations.

A second alternative is the Girard–Hutchinson norm estimator (1.4), discussed in subsection 1.1. The leave-one-out estimator improves on the Girard–Hutchinson estimator as the leave-one-out estimate does not require any additional matrix-vector products and automatically improves in quality as the number of vectors s is increased.

3. Matrix jackknife variance estimation. This section outlines our proposal for *matrix jackknife variance estimation* for more general randomized matrix algorithms. Subsection 3.1 reviews jackknife variance estimation for scalar quantities. We introduce and analyze the matrix jackknife variance estimator in subsection 3.2. Subsections 3.3 to 3.5 discuss potential applications of the matrix jackknife and complementary topics.

3.1. Tukey's jackknife variance estimator and the Efron–Stein–Steele inequality. To motivate our matrix jackknife proposal, we begin by presenting the jackknife variance estimator [26] for scalar estimators due to Tukey in subsection 3.1.1. In subsection 3.1.2, we discuss the Efron–Stein–Steele inequality used in its analysis.

3.1.1. Tukey's jackknife variance estimator. Consider the problem of estimating the variance of a statistical estimator computed from s random samples. We assume it makes sense to evaluate the estimator with fewer than s samples, as is the case for many classical estimators like the sample mean and variance. This motivates the following setup:

- Let $\omega_1, \dots, \omega_s$ be iid random elements taking values in a measurable space Ω.
- Let f denote either one of two estimators, defined for s or $s-1$ arguments:

$$f : \Omega^s \to \mathbb{K} \quad \text{or} \quad f : \Omega^{s-1} \to \mathbb{K}.$$

- Assume that f is invariant to a reordering of its inputs:

$$f(\omega_1, \dots, \omega_s) = f(\omega_{\pi(1)}, \dots, \omega_{\pi(s)}) \quad \text{for any permutation } \pi.$$

- Define estimates $E_{s-1} := f(\omega_1, \dots, \omega_{s-1})$ and $E_s := f(\omega_1, \dots, \omega_s)$.

We think of E_s as a statistic computed from a collection of samples $\omega_1, \dots, \omega_s$. We can also evaluate the statistic with only $s-1$ samples, resulting in E_{s-1}.

Tukey's jackknife variance estimator provides an estimate for $\mathrm{Var}(E_{s-1})$, which serves as a proxy for the variance of the s-sample estimator E_s. Define jackknife replicates and mean

$$E^{(j)} := f(\omega_1, \dots, \omega_{j-1}, \omega_{j+1}, \dots, \omega_s) \quad \text{for each } j = 1, 2, \dots, s; \quad E^{(\cdot)} := \frac{1}{s}\sum_{j=1}^{s} E^{(j)}.$$

The quantities $E^{(1)}, \dots, E^{(s)}$ represent the statistic recomputed with each of the samples $\omega_1, \dots, \omega_s$ left out in turn.

Tukey's estimator for $\mathrm{Var}(E_{s-1})$ is given by

$$\widehat{\mathrm{Var}}(E_{s-1}) := \sum_{j=1}^{s} \left| E^{(j)} - E^{(\cdot)} \right|^2. \tag{3.1}$$

Observe that Tukey's estimator (3.1) is the sample variance of the jackknife replicates $E^{(j)}$ up to a normalizing constant. The form of Tukey's estimator suggests that the distribution of the jackknife replicates somehow approximates the distribution of the estimator. This intuition can be formalized using the Efron–Stein–Steele inequality.

3.1.2. Efron–Stein–Steele inequality. To analyze Tukey's estimator, we rely on an inequality of Efron and Stein [4], which was improved by Steele [21].

FACT 3.1 (Efron–Stein–Steele inequality). *Let $\omega_1, \dots, \omega_s \in \Omega$ be independent elements in a measurable space Ω, and let $f : \Omega^s \to \mathbb{K}$ be measurable. Let $(\omega_j' : j = 1, \dots, s)$ be an independent copy of $(\omega_j : j = 1, \dots, s)$. Then*

$$\mathrm{Var}\left(f(\omega_1, \dots, \omega_s)\right) \le \frac{1}{2} \sum_{i=1}^{s} \mathbb{E} \left| f(\omega_1, \dots, \omega_s) - f(\omega_1, \dots, \omega_{j-1}, \omega_j', \omega_{j+1}, \dots, \omega_s) \right|^2. \tag{3.2}$$

The complex-valued version of the inequality presented here follows from the more standard version for real values by treating the real and imaginary parts separately.

In the setting of Tukey's estimator (3.1), the samples $\omega_1, \dots, \omega_{s-1}$ are identically distributed and the function f depends symmetrically on its arguments. Therefore, the last sample ω_s can be used to fill the role of each ω_j' in the right-hand side of (3.2). As a consequence, the Efron–Stein–Steele inequality shows that

$$\begin{aligned} \mathrm{Var}(E_{s-1}) &\le \frac{1}{2} \sum_{i=1}^{s-1} \mathbb{E} \left(E^{(j)} - E^{(s)} \right)^2 = \frac{1}{2s} \sum_{i,j=1}^{s} \mathbb{E} \left(E^{(i)} - E^{(j)} \right)^2 \\ &= \sum_{j=1}^{s} \mathbb{E} \left(E^{(j)} - E^{(\cdot)} \right)^2 = \mathbb{E}\, \widehat{\mathrm{Var}}(E_{s-1}). \end{aligned} \tag{3.3}$$

To move from the first line to the second, we expand the square, use the definition of the mean $E^{(\cdot)} = s^{-1} \sum_{j=1}^{s} E^{(j)}$, and regroup terms. This computation shows that Tukey's variance estimator (3.1) *overestimates* the true variance on average.

3.2. The matrix jackknife estimator of variance. Suppose we are interested in the variance of the output $\boldsymbol{X} \in \mathbb{K}^{d_1 \times d_2}$ to a randomized matrix algorithm. Similar to the scalar setting, we assume that $\boldsymbol{X}$ is a function of independent samples $\omega_1, \dots, \omega_s$ and that it makes sense to evaluate $\boldsymbol{X}$ with fewer than s samples. The formal setup is as follows:

- Let $\omega_1, \dots, \omega_s$ be iid random elements in a measurable space Ω.
- Let $\boldsymbol{X}$ denote one of two matrix estimators defined for s or $s-1$ inputs:

$$\boldsymbol{X} : \Omega^s \to \mathbb{K}^{d_1 \times d_2} \quad \text{or} \quad \boldsymbol{X} : \Omega^{s-1} \to \mathbb{K}^{d_1 \times d_2}.$$

- Assume $\boldsymbol{X}$ is invariant to reordering of its inputs:

$$\boldsymbol{X}(\omega_1, \dots, \omega_s) = \boldsymbol{X}(\omega_{\pi(1)}, \dots, \omega_{\pi(s)}) \quad \text{for any permutation } \pi.$$

- Define estimates $\boldsymbol{X}_s := \boldsymbol{X}(\omega_1, \dots, \omega_s)$ and $\boldsymbol{X}_{s-1} := \boldsymbol{X}(\omega_1, \dots, \omega_{s-1})$.

For a randomized low-rank approximation algorithm, the samples $\omega_1, \dots, \omega_s$ might represent the columns of a test matrix $\boldsymbol{\Omega}$, as they did in subsection 1.2.

We are interested in estimating the variance of $\boldsymbol{X}_{s-1}$ as a proxy for the variance of $\boldsymbol{X}_s$. We expect that adding additional samples will refine the approximation and thus reduce its variance. Define jackknife replicates $\boldsymbol{X}^{(j)}$ and their average $\boldsymbol{X}^{(\cdot)}$,

$$\boldsymbol{X}^{(j)} = \boldsymbol{X}(\omega_1, \dots, \omega_{j-1}, \omega_{j+1}, \dots, \omega_s) \quad \text{for each } j = 1, \dots, s; \quad \boldsymbol{X}^{(\cdot)} := \frac{1}{s} \sum_{j=1}^{s} \boldsymbol{X}^{(j)}.$$

We propose the matrix jackknife estimate

$$\operatorname{Jack}^2 \left(\boldsymbol{X}_{s-1} \right) := \sum_{j=1}^{s} \left\| \boldsymbol{X}^{(j)} - \boldsymbol{X}^{(\cdot)} \right\|_{\mathrm{F}}^2 \tag{3.4}$$

for the variance $\operatorname{Var}(\boldsymbol{X}_{s-1})$. The estimator $\operatorname{Jack}(\boldsymbol{X}_{s-1})$ can be efficiently computed for several randomized low-rank approximations, as we shall demonstrate in section 4. Similar to the classic jackknife variance estimator, we can use the Efron–Stein–Steele inequality to show that this variance estimate is an overestimate on average.

THEOREM 3.2 (matrix jackknife). *With the prevailing notation,*

$$\operatorname{Var}(\boldsymbol{X}_{s-1}) \leq \mathbb{E} \operatorname{Jack}^2 \left(\boldsymbol{X}_{s-1} \right). \tag{3.5}$$

Proof. Fix a pair of indices $1 \leq m \leq d_1$ and $1 \leq n \leq d_2$. Applying (3.3) to the (m,n)-matrix entry $(\boldsymbol{X}_{s-1})_{mn}$, we observe

$$\mathbb{E} \left| (\boldsymbol{X}_{s-1})_{mn} - \mathbb{E} \, (\boldsymbol{X}_{s-1})_{mn} \right|^2 \leq \mathbb{E} \sum_{j=1}^{s} \left| \boldsymbol{X}^{(j)}_{mn} - \boldsymbol{X}^{(\cdot)}_{mn} \right|^2.$$

Summing this equation over all $1 \leq m \leq d_1$ and $1 \leq n \leq d_2$ yields the stated result. ∎

When the jackknife variance estimate is small, Theorem 3.2 shows the variance of the approximation is also small. Empirical evidence (section 5) suggests that $\operatorname{Var}(\boldsymbol{X}_{s-1})$ and $\mathbb{E} \operatorname{Jack}^2(\boldsymbol{X}_{s-1})$ tend to be within an order of magnitude for the algorithms we considered.

It is natural to ask whether we can develop and theoretically jackknife estimates of the bias of randomized matrix algorithms to complement our variance estimate, perhaps using the natural analogue of Quenouille's scalar jackknife bias estimate [17]. This is an interesting question for future work. As we have demonstrated in subsection 1.2 and will further demonstrate in section 5, our variance estimate already provides useful and actionable information for randomized matrix algorithms.

3.3. Uses for matrix jackknife variance estimator. Variance is a useful diagnostic for randomized matrix approximations. When the variance is large, it suggests that one of two issues has arisen:

1. More samples are needed to refine the approximation.
2. The underlying approximation problem is badly conditioned.

In either case, the jackknife variance estimate can provide evidence that the computed output should not be trusted.

We anticipate the primary use case for matrix jackknife variance estimation will be for computations using eigenvectors or singular vectors computed by randomized low-rank approximation algorithms such as the randomized SVD and Nyström approximation. Spectral computations with randomized algorithms currently lack effective posterior estimates, making jackknife variance estimation one of the few available

tools to assess the quality of the outputs of such computations at runtime. In the context of spectral computations, matrix jackknife variance estimation can be used to adaptively determine the approximation rank s needed to achieve outputs of sufficiently high quality. This was demonstrated in subsection 1.2, where we used jackknife variance estimation to determine how large to pick s in a spectral clustering context. As we will later demonstrate in section 5, we can also use the jackknife variance estimation to detect ill-disposed eigenvectors and to get *coordinatewise* variance estimates for singular vector computations.

3.4. Matrix jackknife versus scalar jackknife. Sometimes, we are only interested in scalar outputs of a randomized matrix computation, such as eigenvalues or singular values, entries of eigenvectors or singular vectors, or the trace. In these cases, it might be more efficient to directly apply Tukey's variance estimator (3.1) to assess the variance of these scalars. The matrix jackknife may still be a useful tool because it gives *simultaneous* variance estimates over many scalar quantities. As examples, the matrix jackknife estimates the maximum variance over all linear functionals:

$$\mathbb{E} \max_{\|\boldsymbol{C}\|_{\mathrm{F}} \leq 1} |\mathrm{tr}(\boldsymbol{C}\boldsymbol{X}_{s-1}) - \mathrm{tr}(\boldsymbol{C}\,\mathbb{E}\,\boldsymbol{X}_{s-1})|^2 = \mathbb{E}\,\|\boldsymbol{X}_{s-1} - \mathbb{E}\,\boldsymbol{X}_{s-1}\|_{\mathrm{F}}^2 \leq \mathbb{E}\,\mathrm{Jack}^2(\boldsymbol{X}_{s-1}).$$

The matrix jackknife gives the following variance estimate for the singular values:

$$\mathbb{E} \sum_{j=1}^{\min(d_1,d_2)} |\sigma_j(\boldsymbol{X}_{s-1}) - \sigma_j(\mathbb{E}\,\boldsymbol{X}_{s-1})|^2 = \mathbb{E}\,\|\boldsymbol{X}_{s-1} - \mathbb{E}\,\boldsymbol{X}_{s-1}\|_{\mathrm{F}}^2 \leq \mathbb{E}\,\mathrm{Jack}^2(\boldsymbol{X}_{s-1}).$$

Thus, the matrix jackknife is appealing even when one is interested in multiple scalar-valued functions of the matrix approximation $\boldsymbol{X}_{s-1}$. In addition, efficient algorithms for matrix jackknife variance estimation, as detailed in section 4, are useful for scalar jackknife variance estimation of functionals of a randomized matrix approximation.

3.5. Related work: Bootstrap for randomized matrix algorithms. Bootstrap resampling [3, section 5], a close relative of the jackknife, has seen several applications to matrix computations. An early use case was to provide confidence intervals for eigenvalues and eigenvectors of sample covariance matrices [22, section 7.2].

A more recent line of work, led by Lopes and collaborators, applies bootstrap resampling to randomized matrix algorithms [9, 10, 11, 30]. The work closest to ours [9] uses the bootstrap to provide asymptotically sharp estimates of the error quantiles with regards to general error metrics for each singular value and singular vector computed by a *Monte Carlo-type* sketched SVD algorithm. In this special case, the bootstrap provides more fine-grained information than the matrix jackknife. Unfortunately, this sketched SVD is a poor computational method because its error decays at the Monte Carlo rate.

The main benefit of our matrix jackknife approach is that it is effective for very general matrix algorithms, such as the randomized SVD and Nyström approximation. These algorithms are used widely in practice because they achieve spectral accuracy, producing errors comparable with the best low-rank approximation [7]. As we demonstrate in section SM2, a straightforward application of the bootstrap variance of Efron [3, section 5.1] to the randomized SVD can produce standard deviation estimates which are incorrect by over four orders of magnitude.

4. Case studies in low-rank approximation. In this section, we develop *efficient* computational procedures to compute the jackknife variance estimate and leave-one-out error estimate for two randomized low-rank approximations, randomized Nyström approximation (subsection 4.1) and the randomized SVD (subsection 4.2).

4.1. Nyström approximation. Given a test matrix $\boldsymbol{\Omega} \in \mathbb{K}^{d\times s}$, consider again the Nyström approximation (1.1) with q steps of subspace iteration (1.2) applied to a psd matrix $\boldsymbol{A} \in \mathbb{K}^{d\times d}$:

$$\boldsymbol{X} = \boldsymbol{X}(\boldsymbol{\Omega}) := \boldsymbol{A}\langle\boldsymbol{\Phi}\rangle = \boldsymbol{A}\boldsymbol{\Phi}(\boldsymbol{\Phi}^*\boldsymbol{A}\boldsymbol{\Phi})^\dagger(\boldsymbol{A}\boldsymbol{\Phi})^*, \quad \text{where} \quad \boldsymbol{\Phi} = \boldsymbol{A}^q\boldsymbol{\Phi}.$$

The Nyström approximation $\boldsymbol{X}$ is the best psd approximation to $\boldsymbol{A}$ spanned by $\boldsymbol{A}\boldsymbol{\Phi}$ with a psd residual. We focus on the case where $\boldsymbol{\Omega}$ is populated with iid, isotropic columns, such as when $\boldsymbol{\Omega}$ is a standard Gaussian matrix.

We work with the Nyström approximation in eigenvalue decomposition form:

$$\boldsymbol{X} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^*,$$

where $\boldsymbol{V} \in \mathbb{K}^{d\times s}$ has orthonormal columns and $\boldsymbol{\Lambda} \in \mathbb{R}_+^{s\times s}$ is diagonal. To facilitate efficient computations of our diagnostics, we can compute the Nyström approximation in eigendecomposition form as follows:

1. Draw a test matrix $\boldsymbol{\Omega} \in \mathbb{K}^{n\times s}$ with iid isotropic columns.
2. Apply subspace iteration $\boldsymbol{\Phi} = \boldsymbol{A}^q\boldsymbol{\Omega}$.
3. Compute the product $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{\Omega}$.
4. Orthonormalize $\boldsymbol{Q} = \operatorname{orth}(\boldsymbol{Y})$ using economy QR factorization $\boldsymbol{Y} = \boldsymbol{Q}\boldsymbol{R}$.
5. Compute $\boldsymbol{H} = \boldsymbol{\Omega}^*\boldsymbol{Y}$ and Cholesky factorize $\boldsymbol{H} = \boldsymbol{C}^*\boldsymbol{C}$.
6. Obtain a singular value decomposition $\boldsymbol{R}\boldsymbol{C}^{-1} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{Z}^*$.
7. Set $\boldsymbol{\Lambda} := \boldsymbol{\Sigma}^2$ and $\boldsymbol{V} := \boldsymbol{Q}\boldsymbol{U}$.

Algorithm 4.1 provides an implementation with $q = 0$ with tricks to improve its numerical stability adapted from [8, 24]. For $q > 0$, it may be necessary to introduce additional orthogonalization steps for reasons of numerical stability [20, Algorithm 5.2].

Treating the Nyström approximation $\boldsymbol{X}$ as a symmetric function of the iid, isotropic columns $\boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_s$ of the test matrix $\boldsymbol{\Omega}$,

$$\boldsymbol{X} = \boldsymbol{X}(\boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_s),$$

we can apply both the leave-one-out error estimator and jackknife variance estimation to $\boldsymbol{X}$. Define replicates

$$\boldsymbol{X}^{(j)} = \boldsymbol{X}(\boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_{j-1}, \boldsymbol{\omega}_{j+1}, \dots, \boldsymbol{\omega}_s).$$

To compute the replicates efficiently, we use the update formula [5, equation (2.4)]

$$\boldsymbol{X}^{(j)} = \boldsymbol{V}\left(\boldsymbol{\Lambda} - \boldsymbol{t}_j\boldsymbol{t}_j^*\right)\boldsymbol{V}^*, \tag{4.1a}$$

where $\boldsymbol{t}_1, \dots, \boldsymbol{t}_s$ are the columns of the matrix

$$\boldsymbol{T} = \boldsymbol{U}^*\boldsymbol{R}\boldsymbol{H}^{-1} \cdot \operatorname{diag}\left((\boldsymbol{H}^{-1})_{ii}^{-1/2} : i = 1, 2, \dots, s\right). \tag{4.1b}$$

A derivation of this formula is provided in subsection A.1.

The update formula facilitates efficient algorithms for the leave-one-out error estimator and jackknife variance estimates for the Nyström approximation and derived quantities like spectral projectors and truncation of $\boldsymbol{X}$ to rank $r < s$. To not belabor the point by presenting all possible variations, Algorithm 4.1 presents an implementation of Nyström approximation without subspace iteration (i.e., $q = 0$) with the leave-one-out error estimate $\widehat{\operatorname{Err}}(\boldsymbol{X}, \boldsymbol{A})$. The computation of the error estimate is a simple addition to the algorithm, requiring just a single line and taking only $\mathcal{O}(s^3)$ operations, independent of the size d of the input matrix. Further variants are discussed in subsection SM1.1 and a MATLAB implementation is provided in Program SM1.

Algorithm 4.1. Nyström approximation ($q = 0$) with leave-one-out error estimate.

Input: $\boldsymbol{A} \in \mathbb{K}^{d\times d}$ to be approximated and approximation rank s
Output: Factors $\boldsymbol{V} \in \mathbb{K}^{d\times s}$ and $\boldsymbol{\Lambda} \in \mathbb{R}^{s\times s}$ defining a rank-s approximation $\boldsymbol{X} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^*$ and leave-one-out error estimate $\widehat{\mathrm{Err}}$

$\boldsymbol{\Omega} \leftarrow \mathtt{randn}(d, s)$
$\boldsymbol{Y} \leftarrow \boldsymbol{A}\boldsymbol{\Omega}$
$\nu \leftarrow \epsilon_{\mathrm{mach}} \|\boldsymbol{Y}\|$ and $\boldsymbol{Y} \leftarrow \boldsymbol{Y} + \nu\boldsymbol{\Omega}$ ▷ Shift for numerical stability
$(\boldsymbol{Q}, \boldsymbol{R}) \leftarrow \mathtt{qr}(\boldsymbol{Y}, \mathtt{'econ'})$ ▷ Economy QR factorization
$\boldsymbol{H} \leftarrow \boldsymbol{\Omega}^*\boldsymbol{Y}$
$\boldsymbol{C} \leftarrow \mathtt{chol}((\boldsymbol{H} + \boldsymbol{H}^*)/2)$ ▷ Upper triangular Cholesky decomposition $\boldsymbol{H} = \boldsymbol{C}^*\boldsymbol{C}$
$(\boldsymbol{U}, \boldsymbol{\Sigma}, \sim) \leftarrow \mathtt{svd}(\boldsymbol{R}\boldsymbol{C}^{-1})$ ▷ Triangular solve
$\boldsymbol{\Lambda} \leftarrow \max(\boldsymbol{\Sigma}^2 - \nu\mathbf{I}, 0)$ ▷ Entrywise maximum, shift back for numerical stability
$\boldsymbol{V} \leftarrow \boldsymbol{Q}\boldsymbol{U}$
$\widehat{\mathrm{Err}} \leftarrow \left\|(\boldsymbol{R}\boldsymbol{C}^{-1})\boldsymbol{C}^{-*} \cdot \mathrm{diag}\{(\boldsymbol{H}_{ii}^{-1})^{-1} : i = 1, \ldots, s)\}\right\|_{\mathrm{F}} / \sqrt{s}$

4.2. Randomized SVD. The randomized SVD computes a rank-s approximation to $\boldsymbol{A} \in \mathbb{K}^{d_1\times d_2}$ formed as an economy SVD $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$, where $\boldsymbol{U} \in \mathbb{K}^{d_1\times s}$ and $\boldsymbol{V} \in \mathbb{K}^{d_2\times s}$ have orthonormal columns and $\boldsymbol{\Sigma} \in \mathbb{R}_+^{s\times s}$ is diagonal. With $q \geq 0$ steps of subspace iteration, the algorithm proceeds as follows:

1. Draw a *test matrix* $\boldsymbol{\Omega} \in \mathbb{C}^{d_2\times s}$ with iid isotropic columns.
2. Compute the product $\boldsymbol{Y} = (\boldsymbol{A}\boldsymbol{A}^*)^q\boldsymbol{A}\boldsymbol{\Omega}$.
3. Orthonormalize $\boldsymbol{Q} = \mathrm{orth}(\boldsymbol{Y})$ using economy QR factorization $\boldsymbol{Y} = \boldsymbol{Q}\boldsymbol{R}$.
4. Form the matrix $\boldsymbol{C} = \boldsymbol{Q}^*\boldsymbol{A}$.
5. Compute an economy SVD $\boldsymbol{C} = \boldsymbol{W}\boldsymbol{\Sigma}\boldsymbol{V}^*$.
6. Set $\boldsymbol{U} = \boldsymbol{Q}\boldsymbol{W}$.

The output $\boldsymbol{X}$ is a symmetric function of the iid, isotropic columns $\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_s$ of $\boldsymbol{\Omega}$,

$$\boldsymbol{X} = \boldsymbol{X}(\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_s),$$

making it a candidate for leave-one-out error estimation and jackknife variance estimation.

To compute the replicates efficiently, we will use the following update formula for the $\boldsymbol{Q}$ matrix in the randomized SVD [5, equation (2.1)]:

$$\boldsymbol{Q}^{(j)}\left(\boldsymbol{Q}^{(j)}\right)^* = \boldsymbol{Q}\left(\mathbf{I} - \boldsymbol{t}_j\boldsymbol{t}_j^*\right)\boldsymbol{Q}^*, \tag{4.2}$$

where $\boldsymbol{Q}^{(j)}$ denotes the $\boldsymbol{Q}$ matrix produced by the randomized SVD algorithm executed without the jth column of $\boldsymbol{\Omega}$ and the vectors $\boldsymbol{t}_1, \ldots, \boldsymbol{t}_s$ are the normalized columns of $(\boldsymbol{R}^*)^{-1}$. See Appendix A.2 for a derivation. With this formula, the replicates are easily computed,

$$\boldsymbol{X}^{(j)} = \boldsymbol{Q}^{(j)}\left(\boldsymbol{Q}^{(j)}\right)^*\boldsymbol{A} = \boldsymbol{Q}\left(\mathbf{I} - \boldsymbol{t}_j\boldsymbol{t}_j^*\right)\boldsymbol{Q}^*\boldsymbol{A} = \boldsymbol{U}(\mathbf{I} - \boldsymbol{W}^*\boldsymbol{t}_j\boldsymbol{t}_j^*\boldsymbol{W})\boldsymbol{\Sigma}\boldsymbol{V}^*. \tag{4.3}$$

The update formula enables efficient algorithms for the leave-one-out error estimator and jackknife variance estimator for the approximation $\boldsymbol{X}$ and derived quantities like projectors onto singular subspaces and truncation of $\boldsymbol{X}$ to rank $r < s$. As one example, Algorithm 4.2 gives an implementation the randomized SVD with no subspace iteration ($q = 0$) with the leave-one-out error estimator $\widehat{\mathrm{Err}}(\boldsymbol{X}, \boldsymbol{A})$. The leave-one-out error estimator requires just two lines and runs in $\mathcal{O}(s^3)$ operations. Further variants

Algorithm 4.2. Randomized SVD ($q = 0$) with jackknife variance estimate.

Input: $\boldsymbol{A} \in \mathbb{K}^{d_1 \times d_2}$ to be approximated and approximation rank s
Output: Factors $\boldsymbol{U} \in \mathbb{K}^{d_1 \times s}$, $\boldsymbol{\Sigma} \in \mathbb{R}^{s \times s}$, and $\boldsymbol{V} \in \mathbb{K}^{d_2 \times s}$ defining a rank-s approximation $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$, leave-one-out error estimate $\widehat{\mathrm{Err}} = \widehat{\mathrm{Err}}(\boldsymbol{X}, \boldsymbol{A})$

1: $\boldsymbol{\Omega} \leftarrow \mathtt{randn}(d_2, s)$
2: $\boldsymbol{Y} \leftarrow \boldsymbol{A}\boldsymbol{\Omega}$
3: $(\boldsymbol{Q}, \boldsymbol{R}) \leftarrow \mathtt{qr}(\boldsymbol{Y}, \mathtt{'econ'})$ ▷ Economy QR factorization
4: $\boldsymbol{C} \leftarrow \boldsymbol{Q}^*\boldsymbol{A}$
5: $(\boldsymbol{W}, \boldsymbol{\Sigma}, \boldsymbol{V}) \leftarrow \mathtt{svd}(\boldsymbol{C}, \mathtt{'econ'})$
6: $\boldsymbol{U} \leftarrow \boldsymbol{Q}\boldsymbol{W}$
7: $\boldsymbol{G} \leftarrow (\boldsymbol{R}^*)^{-1}$
8: $\widehat{\mathrm{Err}} \leftarrow \left(s^{-1} \sum_{j=1}^{s} \|\boldsymbol{G}(:, j)\|^{-2} \right)^{1/2}$

are discussed in subsection SM1.3 and a MATLAB implementation is provided in Program SM5.

5. Numerical experiments. In this section, we showcase numerical examples that demonstrate the effectiveness of the matrix jackknife variance estimate and leave-one-out error estimate for the Nyström approximation and randomized SVD. All numerical experiments work over the real numbers, $\mathbb{K} = \mathbb{R}$.

5.1. Experimental setup. To evaluate our diagnostics for matrices with different spectral characteristics, we consider synthetic test matrices from [24, section 5]:

(NoisyLR) $$\boldsymbol{A} = \mathrm{diag}(\underbrace{1, \dots, 1}_{R \text{ times}}, 0 \dots, 0) + \xi d^{-1} \boldsymbol{G}\boldsymbol{G}^* \in \mathbb{R}^{d \times d}.$$

(ExpDecay) $$\boldsymbol{A} = \mathrm{diag}(\underbrace{1, \dots, 1}_{R \text{ times}}, 10^{-q}, 10^{-2q}, \dots, 10^{-(d-R)q}) \in \mathbb{R}^{d \times d}.$$

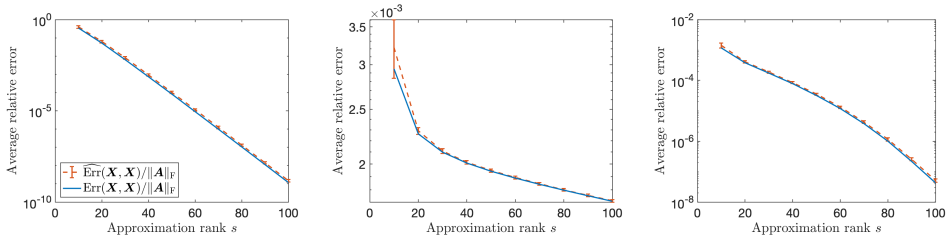
Here, $\xi, q \in \mathbb{R}$ are parameters, and $\boldsymbol{G} \in \mathbb{R}^{d \times d}$ is a standard Gaussian matrix. Using diagonal test matrices is justified by the observation that the randomized SVD, Nyström approximation, and our diagnostics are orthogonally invariant when $\boldsymbol{\Omega}$ is a standard Gaussian matrix, which we use. We also consider matrices from applications:

- **Velocity.** We consider a matrix $\boldsymbol{A} \in \mathbb{R}^{25096 \times 1000}$ whose columns are snapshots of the velocity and pressure from simulations of a fluid flow past a cylinder. We thank Beverley McKeon and Sean Symon for this data.
- **Spectral clustering.** We consider the matrix $\boldsymbol{A}$ from the spectral clustering example in subsection 1.2.4.

We apply the jackknife variance estimate and leave-one-out error estimate to the randomized Nyström approximation and randomized SVD with a standard Gaussian test matrix $\boldsymbol{\Omega}$ for a range of values for the approximation rank s. For each value of s, we estimate the mean error Err (1.5), standard deviation Std (1.8), mean jackknife estimate Jack, or mean leave-one-out error estimator $\widehat{\mathrm{Err}}$ using 1000 independent trials. Error bars on all figures show one standard deviation.

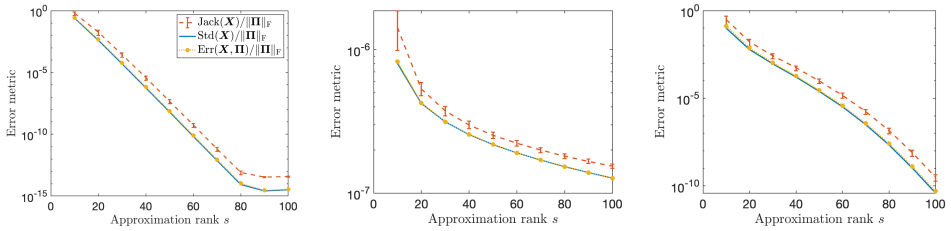
5.2. Leave-one-out error estimator for randomized SVD. First, we apply leave-one-out error estimation to estimate the error for the randomized SVD,

(5.1) $$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*.$$



ExpDecay($q=0.1, R=5$) NoisyLR($\xi=10^{-4}, R=5$) Velocity

FIG. 3. ***Leave-one-out error estimator for randomized SVD.*** *Error and error estimate for randomized SVD ($q=0$) approximation for three different matrices.*



ExpDecay($q=0.1, R=5$) NoisyLR($\xi=10^{-4}, R=5$) Velocity

FIG. 4. ***Matrix jackknife for projectors onto singular subspaces.*** *Error, standard deviation, and jackknife estimate for randomized SVD ($q=0$) approximation $\boldsymbol{X}$ (5.2) to the projector $\boldsymbol{\Pi}$ onto the span of the five dominant right singular vectors for three different matrices.*

We set $q=0$. Figure 3 shows the results for three examples in the previous section. In all cases, error estimate $\widehat{\mathrm{Err}}(\boldsymbol{X},\boldsymbol{A})$ tracks the true error $\mathrm{Err}(\boldsymbol{X},\boldsymbol{A})$ closely. Additional examples and analogous plots for randomized Nyström approximation are provided in section SM3.

5.3. Matrix jackknife for projectors onto singular subspaces. Consider the task of computing the projector $\boldsymbol{\Pi}$ onto the dominant five-dimensional right singular subspace to a matrix $\boldsymbol{A}$. The randomized SVD $\boldsymbol{A}\approx\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$ yields the approximation

$$\boldsymbol{X}=\boldsymbol{V}(:,1:5)\boldsymbol{V}(:,1:5)^*. \tag{5.2}$$

Figure 4 shows the mean error $\mathrm{Err}(\boldsymbol{X},\boldsymbol{\Pi})$, standard deviation $\mathrm{Std}(\boldsymbol{X})$, and jackknife estimate $\mathrm{Jack}(\boldsymbol{X})$ for the same three test matrices as previously. We again set $q=0$. Consistent with Theorem 3.2, the jackknife estimate $\mathrm{Jack}(\boldsymbol{X})$ is an *overestimate* of $\mathrm{Std}(\boldsymbol{X})$ by a factor of $2\times$ to $8\times$. While the jackknife is not quantitatively sharp, it provides an order of magnitude estimate of the standard deviation and is a useful diagnostic for the quality of the computed output. Additional examples and plots for randomized Nyström approximations of projectors onto invariant subspaces are provided in section SM3.

5.4. Application: Diagnosing ill-conditioning for spectral clustering. Jackknife variance estimates can be used to identify situations when a computational task is ill-posed or ill-conditioned. In such cases, refining the approximation (i.e., increasing s or q) may be of little help to improve the quality of the computation.

As an example, consider the spectral clustering application from subsection 1.2. In Nyström-accelerated spectral clustering, we use the dominant n_{dim} eigenvectors of a Nyström approximation $\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^*$ as coordinates for k-means clustering. For spectral clustering to be reliable, we should pick the parameter n_{dim} such that these coordinates are well-conditioned; that is, they should not be highly sensitive to small changes in the normalized kernel matrix $\boldsymbol{A}$. For the example in subsection 1.2.4, the five largest eigenvalues of $\boldsymbol{A}$ are

$$\begin{aligned}\lambda_1 &= 0.999999999999999,\\ \lambda_2 &= 0.999999998639842,\\ \lambda_3 &= 0.999999940523446,\\ \lambda_4 &= 0.999999931126177,\\ \lambda_5 &= 0.99\mathbf{7}867975285136.\end{aligned}$$

The first four eigenvalues agree up to eight digits of accuracy, with the fifth eigenvalue separated by $\lambda_4 - \lambda_5 \approx 2 \times 10^{-3}$. Based on these values, the natural parameter setting would be $n_{\mathrm{dim}} = 4$, as the first four eigenvalues are nearly indistinguishable but are well-separated from the fifth.

When we use Nyström-accelerated spectral clustering, we do not have access to the true eigenvalues of the matrix $\boldsymbol{A}$ and thus can have difficulties selecting the parameter n_{dim} appropriately. Fortunately, the jackknife variance estimate can help warn the user of a poor choice for n_{dim}. Let

$$\boldsymbol{X} = \boldsymbol{V}(:, 1 : n_{\mathrm{dim}})\boldsymbol{V}(:, 1 : n_{\mathrm{dim}})^* \tag{5.3}$$

be the orthoprojector onto the dominant n_{dim}-dimensional invariant subspace of the Nyström approximation $\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^*$. Figure 5 shows the standard deviation $\mathrm{Std}(\boldsymbol{X})$ and its jackknife estimate $\mathrm{Jack}(\boldsymbol{X})$ for both $n_{\mathrm{dim}} = 3$ and $n_{\mathrm{dim}} = 4$. As in subsection 1.2.4,

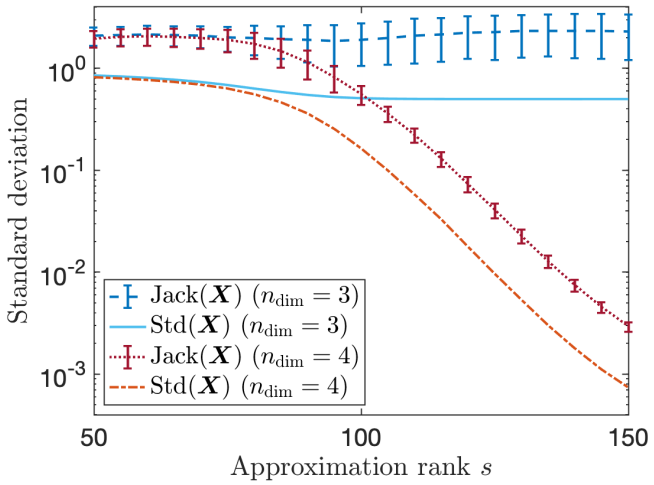
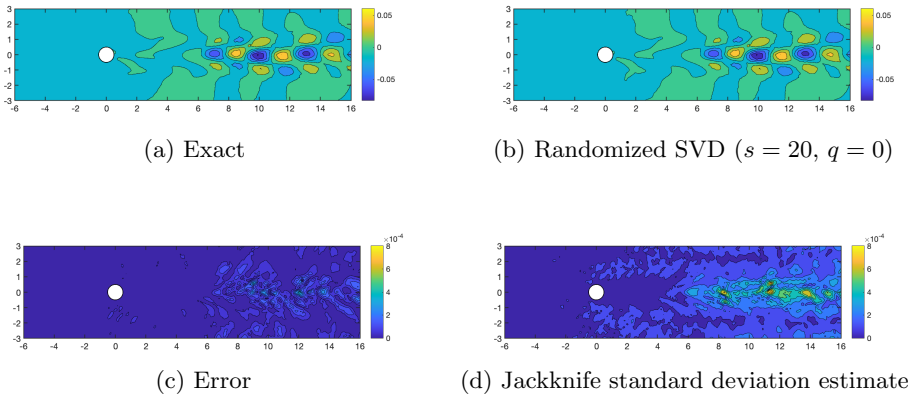


FIG. 5. ***Detecting ill-conditioning.*** *Standard deviation and jackknife estimate for the Nyström spectral projector* (5.3) *associated with the dominant* n_{dim}*-dimensional invariant subspace for* $n_{\mathrm{dim}} \in \{3, 4\}$ *for the spectral clustering matrix.*



(a) Exact (b) Randomized SVD ($s = 20$, $q = 0$)

(c) Error (d) Jackknife standard deviation estimate

FIG. 6. ***Jackknife for singular vector.*** *These panels assess the entrywise errors in the streamwise velocity from the fifth left singular vector of the velocity test matrix. Panel* (a) *shows the exact answer, and panel* (b) *shows the estimate produced by the randomized SVD. Panels* (c) *and* (d) *display the error and the Tukey jackknife standard deviation estimate. The jackknife estimate presents a descriptive portrait of where the error is localized.*

use $q = 3$ and $50 \leq s \leq 150$. For the good parameter setting $n_{\text{dim}} = 4$, the variance decreases sharply as s is increased. For the bad choice $n_{\text{dim}} = 3$, the variance remains persistently high, even as the approximation is refined. This provides evidence to the user that $\boldsymbol{X}$ is poorly conditioned and allows the user to fix this by changing the parameter n_{dim}.

5.5. Application: POD modes. Jackknife variance estimation can be used to give more fine-grained information about a randomized matrix computation. In Figure 6, we compute a (scalar) jackknife variance estimate for the absolute value of each entry of the fifth left singular vector of the velocity matrix computed using the randomized SVD with $s = 20$ and $q = 0$. (The absolute value is introduced to avoid sign ambiguities.)

Left singular vectors of a matrix of simulation data are known as POD modes and are useful for data visualization and model reduction. Since variance is a lower bound on the mean-square error, the coordinatewise jackknife variance estimates can be used as a diagnostic to help identify regions of a POD mode that have high error. This is demonstrated in Figure 6; the jackknife estimate is not quantitatively sharp but paints a descriptive portrait of where the errors are localized.

6. Extension: Variance estimates for higher Schatten norms. The variance estimate $\text{Jack}(\boldsymbol{X})$ serves as an estimate for the *Frobenius-norm* variance

$$\text{Jack}^2(\boldsymbol{X}) \approx \text{Var}(\boldsymbol{X}) = \mathbb{E}\, \|\boldsymbol{X} - \mathbb{E}\, \boldsymbol{X}\|_{\text{F}}^2 .$$

Often, it is more desirable to have error or variance estimates for Schatten norms $|||\cdot|||_p$ with $p > 2$, defined as the ℓ_p norm of the singular values:

$$|||\boldsymbol{B}|||_p^p := \sum_{j=1}^{\min(d_1,d_2)} \sigma_j^p(\boldsymbol{B}).$$

One can also construct jackknife estimates for the variance in higher Schatten norms, although the estimates take more intricate forms. For this section, fix an even number

$p \geq 2$ and assume the same setup as subsection 3.2 with the additional stipulation that the samples $\omega_1, \dots, \omega_s$ take values in a Polish space Ω.

The jackknife variance estimate is defined as follows. Consider matrix-valued jackknife *variance proxies*:

$$\widehat{\mathbf{Var}}_1(\boldsymbol{X}_{s-1}) := \frac{1}{2}\sum_{j=1}^{s-1}\left|\boldsymbol{X}^{(s)} - \boldsymbol{X}^{(j)}\right|^2 \quad \text{and} \quad \widehat{\mathbf{Var}}_2(\boldsymbol{X}_{s-1}) := \frac{1}{2}\sum_{j=1}^{s-1}\left|\left(\boldsymbol{X}^{(s)} - \boldsymbol{X}^{(j)}\right)^*\right|^2.$$

Here, $|\boldsymbol{B}| = (\boldsymbol{B}^*\boldsymbol{B})^{1/2}$ denotes the matrix modulus. Define the Schatten p-norm variance estimate

$$\operatorname{Jack}_p(\boldsymbol{X}_{s-1}) := 2^{-\frac{1}{p}}\sqrt{2(p-1)}\left(\left|\!\left|\!\left|\widehat{\mathbf{Var}}_1\right|\!\right|\!\right|_{p/2}^{p/2} + \left|\!\left|\!\left|\widehat{\mathbf{Var}}_2\right|\!\right|\!\right|_{p/2}^{p/2}\right)^{1/p}.$$

This quantity seeks to approximate

$$\operatorname{Jack}_p^p(\boldsymbol{X}_{s-1}) \approx \mathbb{E}\,|\!|\!|\boldsymbol{X}_{s-1} - \mathbb{E}\,\boldsymbol{X}_{s-1}|\!|\!|_p^p.$$

A matrix generalization of the Efron–Stein–Steele inequality [15, Theorem 4.2] shows this jackknife variance estimate overestimates the Schatten p-norm variance in the sense

$$\mathbb{E}\,|\!|\!|\boldsymbol{X}_{s-1} - \mathbb{E}\,\boldsymbol{X}_{s-1}|\!|\!|_p^p \leq \mathbb{E}\,\operatorname{Jack}_p^p(\boldsymbol{X}_{s-1}).$$

The techniques we introduce in section 4 can be extended in a natural way to compute $\operatorname{Jack}_p(\boldsymbol{X}_{s-1})$ efficiently for the randomized SVD and Nyström approximation.

Appendix A. Derivation of update formulas. In this section, we provide proofs of the update formulas (4.1) and (4.2).

A.1. Proof of (4.1). Assume without loss of generality that $j = s$, instate the notation of subsection 4.1, and assume $\boldsymbol{H}$ is invertible. The key ingredient is the following consequence of the Banachiewicz inversion formula [16, equation (0.7.2)]:

$$\begin{bmatrix}\left(\boldsymbol{H}^{(s)}\right)^{-1} & \mathbf{0}\\ \mathbf{0}^* & 0\end{bmatrix} = \boldsymbol{H}^{-1} - \frac{\boldsymbol{H}^{-1}\mathbf{e}_s\mathbf{e}_s^*\boldsymbol{H}^{-1}}{\mathbf{e}_s^*\boldsymbol{H}^{-1}\mathbf{e}_s}.$$

Using this formula and letting $\boldsymbol{R}_{-s}$ denote $\boldsymbol{R}$ without its sth column, we compute

$$\boldsymbol{X}^{(s)} = \boldsymbol{Q}\boldsymbol{R}_{-s}\left(\boldsymbol{H}^{(s)}\right)^{-1}\boldsymbol{R}_{-s}^*\boldsymbol{Q}^* = \boldsymbol{X} - \frac{\boldsymbol{Q}\boldsymbol{R}\boldsymbol{H}^{-1}\mathbf{e}_s\mathbf{e}_s^*\boldsymbol{H}^{-1}\boldsymbol{R}^*\boldsymbol{Q}^*}{\mathbf{e}_s^*\boldsymbol{H}^{-1}\mathbf{e}_s}.$$

Since $\boldsymbol{Q} = \boldsymbol{V}\boldsymbol{U}^*$ and $\boldsymbol{X} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^*$, we thus have

$$\boldsymbol{X}^{(s)} = \boldsymbol{V}\left(\boldsymbol{\Lambda} - \frac{\boldsymbol{U}^*\boldsymbol{R}\boldsymbol{H}^{-1}\mathbf{e}_s\mathbf{e}_s^*\boldsymbol{H}^{-1}\boldsymbol{R}^*\boldsymbol{Q}^*\boldsymbol{U}}{\mathbf{e}_s^*\boldsymbol{H}^{-1}\mathbf{e}_s}\right)\boldsymbol{V}^* = \boldsymbol{V}\left(\boldsymbol{\Lambda} - \boldsymbol{t}_s\boldsymbol{t}_s^*\right)\boldsymbol{V}^*,$$

where $\boldsymbol{t}_s$ is defined in (4.1b). The formula is established.

A.2. Proof of (4.2). Fix $j \in \{1, \dots, s\}$, instate the notation of subsection 4.2, and assume $\boldsymbol{R}$ is invertible. First observe that

$$\boldsymbol{Y}^{(j)} = \boldsymbol{Q}\boldsymbol{R}_{-j},$$

where $\boldsymbol{R}_{-j}$ is $\boldsymbol{R}$ without its jth column. To compute $\boldsymbol{Q}^{(j)}$, we need an economy QR factorization of $\boldsymbol{Y}^{(j)} = \boldsymbol{Q}\boldsymbol{R}_{-j}$. To this end, compute a (full) QR decomposition of $\boldsymbol{R}_{-j}$:

$$\boldsymbol{R}_{-j} = \begin{bmatrix} \boldsymbol{Q}' & \boldsymbol{t}_j \end{bmatrix} \begin{bmatrix} \boldsymbol{R}^{(j)} \\ \boldsymbol{0}^* \end{bmatrix}, \tag{A.1}$$

where $\boldsymbol{Q}' \in \mathbb{K}^{s\times(s-1)}$, $\boldsymbol{R}^{(j)} \in \mathbb{K}^{(s-1)\times(s-1)}$, and $\boldsymbol{t}_j \in \mathbb{K}^s$. Then

$$\boldsymbol{Y}^{(j)} = \boldsymbol{Q}^{(j)}\boldsymbol{R}^{(j)} \quad \text{for} \quad \boldsymbol{Q}^{(j)} = \boldsymbol{Q}\boldsymbol{Q}'$$

is an economy QR decomposition of $\boldsymbol{Y}^{(j)}$. Since $\begin{bmatrix} \boldsymbol{Q}' & \boldsymbol{t}_j \end{bmatrix}$ is orthogonal, we have

$$\mathbf{I} = \begin{bmatrix} \boldsymbol{Q}' & \boldsymbol{t}_j \end{bmatrix} \begin{bmatrix} \boldsymbol{Q}' & \boldsymbol{t}_j \end{bmatrix}^* = \boldsymbol{Q}'(\boldsymbol{Q}')^* + \boldsymbol{t}_j\boldsymbol{t}_j^* \implies \boldsymbol{Q}'(\boldsymbol{Q}')^* = \mathbf{I} - \boldsymbol{t}_j\boldsymbol{t}_j^*.$$

Thus,

$$\boldsymbol{Q}^{(j)}\left(\boldsymbol{Q}^{(j)}\right)^* = \boldsymbol{Q}\boldsymbol{Q}'(\boldsymbol{Q}')^*\boldsymbol{Q}^* = \boldsymbol{Q}\left(\mathbf{I} - \boldsymbol{t}_j\boldsymbol{t}_j^*\right)\boldsymbol{Q}^*.$$

Finally, observe that (A.1) implies that $\boldsymbol{t}_j$ is orthogonal to the columns of $\boldsymbol{R}_{-j}$ so

$$\boldsymbol{t}_j^*\boldsymbol{R} \quad \text{is a nonzero multiple of } \mathbf{e}_j^*.$$

Therefore, $\boldsymbol{t}_j$ is proportional to the jth column of $(\boldsymbol{R}^*)^{-1}$. Since $\boldsymbol{t}_j$ is a column of an orthogonal matrix, it is equal to a normalized version of the jth column of $(\boldsymbol{R}^*)^{-1}$.

Acknowledgment. We thank Robert Webber for his advice and feedback.

REFERENCES

[1] D. Arthur and S. Vassilvitskii, *k-means++: The advantages of careful seeding*, in Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, Philadelphia, 2007, pp. 1027–1035.

[2] P. Drineas and M. W. Mahoney, *RandNLA: Randomized numerical linear algebra*, Commun. ACM, 59 (2016), pp. 80–90.

[3] B. Efron, *The Jackknife, the Bootstrap and Other Resampling Plans*, SIAM, Philadelphia, 1982.

[4] B. Efron and C. Stein, *The jackknife estimate of variance*, Ann. Statist., 9 (1981), pp. 586–596.

[5] E. N. Epperly, J. A. Tropp, and R. J. Webber, *XTrace: Making the most out of every sample in stochastic trace estimation*, SIAM J. Matrix Anal. Appl., 45 (2024), pp. 1–23, https://doi.org/10.1137/23M1548323.

[6] A. Gittens and M. Mahoney, *Revisiting the Nyström method for improved large-scale machine learning*, in Proceedings of the 30th International Conference on Machine Learning, PMLR, 2013, pp. 567–575.

[7] 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.
[8] H. Li, G. C. Linderman, A. Szlam, K. P. Stanton, Y. Kluger, and M. Tygert, *Algorithm 971: An implementation of a randomized algorithm for principal component analysis*, ACM Trans. Math. Software, 43 (2017), 28.
[9] M. Lopes, N. B. Erichson, and M. Mahoney, *Error estimation for sketched SVD via the bootstrap*, in Proceedings of the 37th International Conference on Machine Learning, Proc. Mach. Learn. Res. 119, PMLR, 2020, pp. 6382–6392.
[10] M. E. Lopes, N. B. Erichson, and M. W. Mahoney, *Bootstrapping the operator norm in high dimensions: Error estimation for covariance matrices and sketching*, Bernoulli, 29 (2023), pp. 428–450.
[11] M. E. Lopes, S. Wang, and M. W. Mahoney, *A bootstrap method for error estimation in randomized matrix multiplication*, J. Mach. Learn. Res., 20 (2019), 39.
[12] P.-G. Martinsson and J. A. Tropp, *Randomized numerical linear algebra: Foundations and algorithms*, Acta Numer., 29 (2020), pp. 403–572.
[13] R. Murray, J. Demmel, M. W. Mahoney, N. B. Erichson, M. Melnichenko, O. A. Malik, L. Grigori, P. Luszczek, M. Derezinski, M. E. Lopes, T. Liang, H. Luo, and J. Dongarra, *Randomized Numerical Linear Algebra: A Perspective on the Field with an Eye to Software*, http://arxiv.org/abs/2302.11474, 2023.
[14] C. Musco and C. Musco, *Randomized block Krylov methods for stronger and faster approximate singular value decomposition*, in Proceedings of the 28th International Conference on Neural Information Processing Systems, Vol. 1, MIT Press, Cambridge, MA, 2015, pp. 1396–1404.
[15] D. Paulin, L. Mackey, and J. A. Tropp, *Efron–Stein inequalities for random matrices*, Ann. Probab., 44 (2016), pp. 3431–3473.
[16] S. Puntanen and G. P. H. Styan, *Historical introduction: Issai Schur and the early development of the Schur complement*, in The Schur Complement and Its Applications, Numer. Methods Algorithms 4, F. Zhang, ed., Springer, New York, 2005, pp. 1–16.
[17] M. H. Quenouille, *Approximate tests of correlation in time-series*, J. R. Stat. Soc. Ser. B. Methodol., 11 (1949), pp. 68–84.
[18] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. von Lilienfeld, *Quantum chemistry structures and properties of 134 kilo molecules*, Scientific Data, 1 (2014), 140022.
[19] L. Ruddigkeit, R. van Deursen, L. C. Blum, and J.-L. Reymond, *Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17*, J. Chem. Inf. Model., 52 (2012), pp. 2864–2875.
[20] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Manchester University Press, Manchester, UK, 1992.
[21] J. M. Steele, *An Efron-Stein inequality for nonsymmetric statistics*, Ann. Statist., 14 (1986), pp. 753–758.
[22] R. J. Tibshirani and B. Efron, *An Introduction to the Bootstrap*, Monogr. Statist. Appl. Probab. 57, CRC Press, Boca Raton, FL, 1993.
[23] J. A. Tropp and R. J. Webber, *Randomized Algorithms for Low-Rank Matrix Approximation: Design, Analysis, and Applictions*, 2023, https://doi.org/10.48550/arXiv.2306.12418.
[24] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher, *Fixed-rank approximation of a positive-semidefinite matrix from streaming data*, in Advances in Neural Information Processing Systems, Vol. 30, 2017, pp. 1225–1234.
[25] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher, *Streaming low-rank matrix approximation with an application to scientific simulation*, SIAM J. Sci. Comput., 41 (2019), pp. A2430–A2463.
[26] J. Tukey, *Bias and confidence in not quite large samples*, Ann. Math. Statist., 29 (1958), p. 614.
[27] U. Von Luxburg, *A tutorial on spectral clustering*, Stat. Comput., 17 (2007), pp. 395–416.
[28] C. K. I. Williams and M. Seeger, *Using the Nyström method to speed up kernel machines*, in Proceedings of the 13th International Conference on Neural Information Processing Systems, MIT Press, Cambridge, MA, 2000, pp. 661–667.
[29] D. P. Woodruff, *Sketching as a tool for numerical linear algebra*, Found. Trends Theoret. Comput. Sci., 10 (2014), pp. 1–157.
[30] J. Yao, N. B. Erichson, and M. E. Lopes, *Error estimation for random Fourier features*, in Proceedings of the 26th International Conference on Artificial Intelligence and Statistics, Proc. Mach. Learn. Res. 206, PMLR, 2023, pp. 2348–2364.