JACKKNIFE VARIABILITY ESTIMATION FOR RANDOMIZED MATRIX COMPUTATIONS*

ETHAN N. EPPERLY† AND JOEL A. TROPP†

Abstract. Randomized algorithms based on sketching have become a workhorse tool in low-rank matrix approximation. To use these algorithms safely in applications, they should be coupled with diagnostics to assess the quality of approximation. To meet this need, this paper proposes a jackknife resampling method to estimate the variability of the output of a randomized matrix computation. The variability estimate can recognize that a computation requires additional data or that the computation is intrinsically unstable. As examples, the paper studies jackknife estimates for two randomized low-rank matrix approximation algorithms. In each case, the operation count for the jackknife estimate is independent of the dimensions of the target matrix. In numerical experiments, the estimator accurately assesses variability and also provides an order-of-magnitude estimate of the mean-square error.

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

AMS subject classifications. 62F40, 65F55, 68W20

1. Introduction. In recent years, randomness has become an essential tool in the design of matrix algorithms [6, 12, 18, 29], with randomized algorithms proving especially effective for low-rank matrix approximation. To use the outputs of randomized algorithms in applications, one should deploy them in conjunction with posterior error estimates or other diagnostics that measure the quality of the computed output.

For certain problems, direct error estimates can be hard to obtain. Consider the problem of computing an approximation $\widetilde{\mathbf{\Pi}}$ to the othoprojector $\mathbf{\Pi}$ onto the leading eigenvector of a symmetric matrix. Estimating the norm of the error $\widetilde{\mathbf{\Pi}} - \mathbf{\Pi}$ appears difficult since we do not have access to $\mathbf{\Pi}$. Indeed, if we could efficiently compute $\mathbf{\Pi}$, there would be no reason to use the randomized approximation $\widetilde{\mathbf{\Pi}}$!

In settings where it is difficult to obtain error estimates, the *variance* can serve as a valuable alternative. High variability of a random approximation usually indicates that more sampling data is needed to produce an accurate result. Variability may also signal that the quantity being approximated is ill-posed or ill-conditioned. In either case, a highly variable approximation should be treated with suspicion.

The variance can also serve as a computable lower bound on the mean-square error. The Frobenius-norm error of a random approximation $\boldsymbol{X}$ to a matrix $\boldsymbol{A}$ admits the bias–

*Submitted to the editors DATE.

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 Number DE-SC0021110. JAT 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).

variance decomposition

$$\mathbb{E}\|\boldsymbol{A}-\boldsymbol{X}\|_{\mathrm{F}}^2 = \|\boldsymbol{A}-\mathbb{E}\boldsymbol{X}\|_{\mathrm{F}}^2 + \mathbb{E}\|\boldsymbol{X}-\mathbb{E}\boldsymbol{X}\|_{\mathrm{F}}^2 =: \|\boldsymbol{A}-\mathbb{E}\boldsymbol{X}\|_{\mathrm{F}}^2 + \mathrm{Var}(\boldsymbol{X}). \tag{1.1}$$

This decomposition shows the variance $\mathrm{Var}(\boldsymbol{X})$ gives a lower bound on the mean-square error $\mathbb{E}\|\boldsymbol{A}-\boldsymbol{X}\|_{\mathrm{F}}^2$. If the squared bias is comparable with the variance, then the variance can be appropriate as an order-of-magnitude estimate for the error, which is often sufficient for applications.

The variance $\mathrm{Var}(\boldsymbol{X})$ depends only on the approximation $\boldsymbol{X}$ itself, so we can estimate the variance even when the underlying matrix $\boldsymbol{A}$ (or $\boldsymbol{\Pi}$ in our earlier example) is inaccessible. In this paper, we propose using a matrix extension of the Tukey jackknife variance estimator [26] as an estimate for the variance. For many randomized low-rank approximation algorithms, this jackknife variance estimate is rapid to compute, and it requires no information beyond that generated by the algorithm. In our experiments, the matrix jackknife proves highly effective at assessing the quality of computed approximations.

1.1. The matrix jackknife idea. To motivate matrix jackknife variance estimation, let us consider a typical algorithm framework. We seek a low-rank approximation to a matrix $\boldsymbol{A} \in \mathbb{R}^{d_1 \times d_2}$. Proceed in two steps:

1. Collect data about the matrix $\boldsymbol{A}$ through matrix–vector products

$$\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{A}\boldsymbol{\omega}_1 & \cdots & \boldsymbol{A}\boldsymbol{\omega}_s \end{bmatrix}$$

   where $\boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_s$ are independent and identically distributed random vectors.
2. Use this data to form an approximation to the matrix

$$\boldsymbol{X} = \boldsymbol{F}(\boldsymbol{Y}; \boldsymbol{A}) \approx \boldsymbol{A}.$$

Several randomized low-rank approximation algorithms fit this template. For instance, the basic randomized SVD algorithm of Halko, Martinsson, and Tropp [12] has this form with

$$\boldsymbol{F}(\boldsymbol{Y}; \boldsymbol{A}) = \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A} \quad \text{where} \quad \boldsymbol{Q} = \mathrm{orth}(\boldsymbol{Y}).$$

As usual, $\mathrm{orth}(\boldsymbol{Y})$ returns a matrix with orthonormal columns that span the range of $\boldsymbol{Y}$.

The matrix jackknife we propose seeks to answer the question:

What is the *variance* of the approximation $\boldsymbol{X}$?

One way we could estimate the variance of the algorithm is to simply run the algorithm repeatedly, each time with a new set $\{\boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_s\}$ of random vectors, and then directly compute the sample variance of the output. This approach is grossly inefficient.

The insight of jackknife methodology [7], particularized to our setting, is that we can get a variance estimate by recomputing the estimate from *subsamples* of the matrix–vector products $\boldsymbol{A}\boldsymbol{\omega}_1, \dots, \boldsymbol{A}\boldsymbol{\omega}_k$. Letting $\boldsymbol{Y}^{(j)}$ denote the data matrix $\boldsymbol{Y}$ with its jth column deleted, form *jackknife replicates* $\boldsymbol{X}^{(j)}$ and their average $\boldsymbol{X}^{(\cdot)}$ via

$$\boldsymbol{X}^{(j)} := \boldsymbol{F}\left(\boldsymbol{Y}^{(j)}; \boldsymbol{A}\right) \quad \text{for each } j = 1, \dots, s \quad \text{and} \quad \boldsymbol{X}^{(\cdot)} := \frac{1}{s}\sum_{j=1}^{s} \boldsymbol{X}^{(j)}.$$

We define the jackknife variance estimate

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

This quantity serves as a proxy for the variance $\mathrm{Var}(\boldsymbol{X}) := \mathbb{E}\|\boldsymbol{X} - \mathbb{E}\boldsymbol{X}\|_{\mathrm{F}}^2$. Guarantees for this jackknife estimator are provided in Theorem 2.2.

In the context of randomized matrix computattions, the jackknife approach has several desirable properties:

- **Efficiency.** The replicates $\boldsymbol{X}^{(j)} = \boldsymbol{F}(\boldsymbol{Y}^{(j)}; \boldsymbol{A})$ are obtained by *downdating* the approximation $\boldsymbol{X} = \boldsymbol{F}(\boldsymbol{Y}; \boldsymbol{A})$ after deleting a column from $\boldsymbol{Y}$. This observation helps us design efficient algorithms for computing the replicates. In sections 3 and 4, we discuss two low-rank approximation algorithms where $\mathrm{Jack}(\boldsymbol{X})$ can be obtained in $\mathcal{O}(s^3)$ operations, independent of *both dimensions* of the matrix $\boldsymbol{A}$.
- **Flexibility.** The jackknife estimate can estimate the variance of general quantities of the form $\boldsymbol{X} = \boldsymbol{F}(\boldsymbol{Y}; \boldsymbol{A})$. In particular, the jackknife can be used to obtain variance estimates for randomized approximations to singular projectors or spectral projectors. Direct error estimates for these objects can be hard to obtain by other means.

1.2. Outline. Section 2 begins with our proposal for jackknife variance estimation for matrix computations. Sections 3 and 4 then instantiate this propsal for two algorithms, the randomized SVD and a single-view Nyström approximation. Both of these sections include an algorithm for computing the variance estimate efficiently, with operation count independent of *both dimensions* of the input matrix. Section 5 contains numerical results demonstating that the jackknife variance estimate closely tracks the true variance of the algorithmic output. Sections 6, 7, and 8 conclude with an extension to variance estimation in other Schatten norms ($p > 2$), related work, and closing remarks.

1.3. Notation. Matrices are denoted by boldface capital letters, vectors by boldface lowercase letters, and scalars by italic letters. We work over the field $\mathbb{K} = \mathbb{R}$ or $\mathbb{K} = \mathbb{C}$. For much of the text, $\boldsymbol{A}$ will be a $d_1 \times d_2$ input matrix while $\boldsymbol{X}$ is a random approximation to $\boldsymbol{A}$.

The symbols * and † denote the conjugate transpose and Moore–Penrose pseudoinverse. For a matrix $\boldsymbol{B} \in \mathbb{K}^{d_1 \times d_2}$ with decreasingly ordered singular values σ_j and with left and right singular vectors $\boldsymbol{u}_j$ and $\boldsymbol{v}_j$ for $1 \le j \le \min(d_1, d_2)$, we use the following notations. The Frobenius norm $\|\boldsymbol{B}\|_{\mathrm{F}} := (\sum_j \sigma_j^2)^{1/2}$. Construct an optimal rank-r approximation

$$[\![\boldsymbol{B}]\!]_r := \sum_{j=1}^{r} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^*.$$

This approximation is unique if and only if $\sigma_r \neq \sigma_{r+1}$, but the meaning will always be unambiguous in context. When $\sigma_{j-1} > \sigma_r > \sigma_{j+1}$, we can introduce the jth left and right singular projectors

$$\boldsymbol{\Pi}_j^{\mathrm{L}}(\boldsymbol{B}) = \boldsymbol{u}_j \boldsymbol{u}_j^* \quad \text{and} \quad \boldsymbol{\Pi}_j^{\mathrm{R}}(\boldsymbol{B}) = \boldsymbol{v}_j \boldsymbol{v}_j^*.$$

If $\boldsymbol{B}$ is positive semidefinite, then the left and right singular projectors coincide, and they both equal the jth spectral projector, which we write $\boldsymbol{\Pi}_j(\boldsymbol{B})$.

The expectation of a $\mathbb{K}$-valued 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, $\mathbb{E}X^2 := \mathbb{E}(X^2)$. Expectations extend entrywise to vector- and matrix-valued quantities. The variance of a random matrix $\boldsymbol{X}$ is defined as

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

2. The matrix jackknife. This section outlines our proposal for a jackknife estimate of the variance of a matrix approximation. Section 2.1 reviews jackknife variance estima-

tion for scalar quantities and the Efron–Stein–Steele inequality, which is used in its analysis. We introduce the matrix jackknife variance estimator in section 2.2. Sections 2.3, 2.4, and 2.5 discuss potential applications of the matrix jackknife and complementary topics.

2.1. Tukey's jackknife variance estimator and the Efron–Stein–Steele inequality. To motivate our matrix jackknife proposal, we begin by presenting Tukey's jackknife estimator [26] for the variance of a scalar statistic in section 2.1.1. In section 2.1.2, we discuss the Efron–Stein–Steele inequality used in its analysis.

2.1.1. 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 independent and identically distributed random elements taking values in a measurable space Ω.
- Let f denote either one of two estimators, defined for either 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 $E_{s-1} \coloneqq f(\omega_1, \dots, \omega_{s-1})$ and $E_s \coloneqq f(\omega_1, \dots, \omega_s)$.

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 the variance of E_{s-1}, which serves as a proxy for the variance of the s-sample estimator E_s. Define jackknife replicates

$$E^{(j)} \coloneqq f(\omega_1, \dots, \omega_{j-1}, \omega_{j+1}, \dots, \omega_s) \quad \text{for each } j = 1, 2, \dots, s. \tag{2.1}$$

The mean of the jackknife replicates is

$$E^{(\cdot)} \coloneqq \frac{1}{s} \sum_{j=1}^{s} E^{(j)}. \tag{2.2}$$

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

$$\widehat{\mathrm{Var}}(E_{s-1}) \coloneqq \sum_{j=1}^{s} \left| E^{(j)} - E^{(\cdot)} \right|^2. \tag{2.3}$$

Observe that Tukey's estimator (2.3) 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.

2.1.2. The Efron–Stein–Steele inequality. To analyze Tukey's variance estimator, we rely on a famous inequality of Efron and Stein [8], which was improved by Steele [23]:

FACT 2.1 (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*

$$\operatorname{Var}\big(f(\omega_1, \dots, \omega_s)\big) \le \frac{1}{2} \sum_{i=1}^{s} \mathbb{E}\left| f(\omega_1, \dots, \omega_s) - f(\omega_1, \dots, \omega_{j-1}, \omega'_j, \omega_{j+1}, \dots, \omega_s) \right|^2. \tag{2.4}$$

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 (2.3), 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 (2.4). As a consequence, the Efron–Stein–Steele inequality shows that

$$\begin{aligned} \operatorname{Var}(E_{s-1}) \le \frac{1}{2} \sum_{i=1}^{s-1} \mathbb{E}\left(E^{(j)} - E^{(s)}\right)^2 &= \frac{1}{2k} \sum_{i,j=1}^{s} \mathbb{E}\left(E^{(i)} - E^{(j)}\right)^2 \\ &= \sum_{j=1}^{s} \mathbb{E}\left(E^{(j)} - E^{(\cdot)}\right)^2 = \mathbb{E}\,\widehat{\operatorname{Var}}(E_{s-1}). \end{aligned} \tag{2.5}$$

That is, Tukey's variance estimator (2.3) *overestimates* the true variance on average.

2.2. The matrix jackknife estimator of variance. Now, suppose we seek to approximate a matrix $\boldsymbol{A} \in \mathbb{K}^{d_1 \times d_2}$ by a random matrix $\boldsymbol{X} \in \mathbb{K}^{d_1 \times d_2}$. 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 independent and identically distributed 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: $\boldsymbol{X}(\omega_1, \dots, \omega_s) = \boldsymbol{X}(\omega_{\pi(1)}, \dots, \omega_{\pi(s)})$ for any permutation π.
- Define estimates $\boldsymbol{X}_s := \boldsymbol{X}(\omega_1, \dots, \omega_s)$ and $\boldsymbol{X}_{s-1} := \boldsymbol{X}(\omega_1, \dots, \omega_{s-1})$.

In the setting of a randomized low-rank approximation algorithm, the samples $\omega_1, \dots, \omega_s$ might represent the columns of a sketching matrix.

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 $\boldsymbol{X}^{(\cdot)}$

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

These quantities are analogous to the replicates (2.1) and average (2.2) from the scalar setting. We propose the matrix jackknife estimate

$$\operatorname{Jack}^2(\boldsymbol{X}_{s-1}) := \sum_{j=1}^{s} \left\| \boldsymbol{X}^{(j)} - \boldsymbol{X}^{(\cdot)} \right\|_{\mathrm{F}}^2 \tag{2.6}$$

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 approximation, as we shall demonstrate in sections 3 and 4. Similar to the classic jackknife variance estimator, we can use the Efron–Stein–Steele inequality to show that this variance estimate is, in an appropriate sense, an overestimate.

THEOREM 2.2 (Matrix jackknife). *With the prevailing notation,*

$$\operatorname{Var}(\boldsymbol{X}_{s-1}) \le \mathbb{E}\operatorname{Jack}^2\left(\boldsymbol{X}_{s-1}\right). \tag{2.7}$$

Proof. Fix a pair of indices $1 \le m \le d_1$ and $1 \le n \le d_2$. Applying (2.5) to the (m,n)-matrix entry $(\boldsymbol{X}_{s-1})_{mn}$, we observe

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

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

When the jackknife variance estimate is small, Theorem 2.2 shows the variance of the approximation is also small. Unfortunately, Theorem 2.2 does not show the converse: It remains possible that $\operatorname{Jack}^2(\boldsymbol{X}_{s-1})$ is large while $\operatorname{Var}(\boldsymbol{X}_{s-1})$ is 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.

2.3. Uses for the jackknife variance estimator. The matrix jackknife estimator is designed to show whether the output of a matrix computation is highly variable. When the variability is large as compared with the scale of the matrix being computed, it suggests that one of two issues has arisen. Either more samples are needed to refine the approximation, or else the underlying approximation problem is badly conditioned. In either case, the jackknife variance estimate can provide a warning that the computed output should not be trusted.

In some settings, the jackknife variance estimate can also provide a reliable order-of-magnitude estimate of the average Frobenius-norm error. We will illustrate this phenomenon in our experiments with two randomized matrix approximation algorithms (section 5), We anticipate that this type of error estimate could have several uses:

- In contrast to randomized norm estimates (section 7.1), the matrix jackknife does not require additional matrix–vector products with the matrix $\boldsymbol{A}$. This makes the matrix jackknife useful in settings where acquiring or storing extra matrix–vector products is computationally expensive.
- The jackknife can be combined with a more accurate and costly error estimate to adaptively determine the number of samples needed to meet an error tolerance. One may use the jackknife as an inexpensive approximation to the error and draw more samples if the jackknife fails to meet the tolerance; the expensive error estimate only needs to be evaluated in marginal cases when the computed approximation is near the threshold.
- The matrix jackknife provides a direct way to assess the accuracy of quantities like singular projectors and spectral projectors or the truncated singular value decomposition, which may be hard to approach by other means.

2.4. What about the bias?. If our ultimate goal is to estimate the error $\|\boldsymbol{A} - \boldsymbol{X}_s\|_{\mathrm{F}}$, then the jackknife variance estimate accounts for the variance term in the bias–variance decomposition (1.1). This naturally raises a question: What do we do about the bias?

Empirically, in the two algorithms we study, we often observe that the squared bias is smaller than the variance (section 5). This evidence gives some justification for using a modest multiple of the jackknife variance estimate as a proxy for the full error.

For particular matrix computations, it may be possible to prove that the bias and variance are always comparable. As a simple case where this relation can be explored, consider the approximation of the identity matrix $\mathbf{I}$ by a uniformly random rank-s orthoprojector $\boldsymbol{X}$.

In this case, we have

$$\mathrm{Bias}^2(\boldsymbol{X}) = \|\mathbf{I} - \mathbb{E}\,\boldsymbol{X}\|_{\mathrm{F}}^2 = \frac{d-s}{\sqrt{d}} \quad \text{and} \quad \mathrm{Var}(\boldsymbol{X}) = \frac{\sqrt{s(d-s)}}{\sqrt{d}}.$$

The ratio of squared bias to variance is roughly $\sqrt{d/s}$ for small s, which is large if $d \gg s$. This example is likely represents something of a worst-case for the bias–variance split, as most inputs $\boldsymbol{A}$ in applications possess a decaying spectrum. We leave further investigation for future work.

One could also attempt to estimate the bias using a matrix version of the Quenouille jackknife bias estimator [21]. The natural matrix analog of the Quenouille bias estimator is

$$\widehat{\mathbf{Bias}}(\boldsymbol{X}_s) := (s-1)\big(\boldsymbol{X}^{(\cdot)} - \boldsymbol{X}_s\big). \tag{2.8}$$

If $\boldsymbol{X}(\cdot)$ is a functional statistic [7, Ch. 2] and the bias $\boldsymbol{X}_s - \mathbb{E}\,\boldsymbol{X}_s$ can be expanded in reciprocal powers of s, then the classic analysis of the jackknife bias estimator [7, §2.1] shows that the bias-corrected estimator $\boldsymbol{X}_s + \widehat{\mathbf{Bias}}(\boldsymbol{X}_s)$ has error $\mathcal{O}(s^{-2})$, reduced from $\mathcal{O}(s^{-1})$. Unfortunately, $\boldsymbol{X}(\cdot)$ is rarely a functional statistic in the context of matrix computations, making it unclear what, if anything, $\widehat{\mathbf{Bias}}(\boldsymbol{X}_s)$ can tell us about the true bias.

2.5. Matrix jackknife versus scalar jackknife. Sometimes, we are only interested in scalar outputs of a randomized matrix computation, such as eigenvalues, singular values, or the trace. In these cases, it might be more efficient to directly apply Tukey's variance estimator (2.3) to assess their variance. 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:

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

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

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

Thus, the matrix jackknife is appealing even in settings where one is interested in multiple scalar-valued functions of the matrix approximation $\boldsymbol{X}_{s-1}$. In addition, efficient methods for computing matrix jackknife estimates for randomized algorithms (sections 3.2 and 4.2) may be useful for accelerating the computation of variance estimates for individual functionals.

3. Case study: Randomized singular value decomposition. In this section, we consider the application of the matrix jackknife to the randomized SVD of Halko, Martinsson, and Tropp [12]. Section 3.1 introduces the randomized SVD with subspace iteration. Section 3.2 shows how the randomized SVD can be interpreted in the framework of the matrix jackknife, and it presents an *efficient* algorithm for computing the variance estimate. Section 3.3 discusses extensions.

3.1. Randomized SVD with subspace iteration. Let $\boldsymbol{A} \in \mathbb{K}^{d_1 \times d_2}$ be a matrix for which we seek an approximate truncated singular value decomposition $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$, 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 a nonnegative diagonal matrix. One may interested in the factors $\boldsymbol{\Sigma}$, $\boldsymbol{U}$, and $\boldsymbol{V}$ as approximations to the singular

values and left and right singular vectors of $\boldsymbol{A}$. Alternatively, one might use $\boldsymbol{X}$ as a low-rank approximation to $\boldsymbol{A}$.

The randomized SVD algorithm with q steps of subspace iteration operates as follows.

1. Draw a random *sketching matrix* $\boldsymbol{\Omega} \in \mathbb{C}^{d_2 \times s}$ with independent standard Gaussian entries.
2. Compute the product $\boldsymbol{Y} = (\boldsymbol{A}\boldsymbol{A}^*)^q \boldsymbol{A}\boldsymbol{\Omega}$.
3. Extract an orthonormal basis $\boldsymbol{Q} = \operatorname{orth}(\boldsymbol{Y})$ by economy QR factorization $\boldsymbol{Y} = \boldsymbol{Q}\boldsymbol{R}$.
4. Form the matrix $\boldsymbol{C} = \boldsymbol{Q}^* \boldsymbol{A}$.
5. Compute an economy SVD $\boldsymbol{C} = \widetilde{\boldsymbol{U}}\boldsymbol{\Sigma}\boldsymbol{V}^*$.
6. Set $\boldsymbol{U} = \boldsymbol{Q}\widetilde{\boldsymbol{U}}$.
7. Report $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$.

Informative *a priori* bounds on the spectral-norm error $\mathbb{E}\|\boldsymbol{A} - \boldsymbol{X}\|$ for the randomized SVD with subspace iteration are well-established; see [12, Thm. 9.3 and Cor. 10.10].

3.2. Matrix jackknife for the randomized SVD. To put the randomized SVD into the framework of the matrix jackknife, we consider the randomized SVD as a function of the columns of the sketching matrix $\boldsymbol{\Omega} = \begin{bmatrix} \boldsymbol{\omega}_1 & \cdots & \boldsymbol{\omega}_s \end{bmatrix}$:

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

Note that $\boldsymbol{X}$ depends symmetrically on the independent and identically distributed samples $\boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_s \in \mathbb{K}^{d_2}$, making the randomized SVD a candidate for the matrix jackknife. To compute the jackknife, we must form the replicates

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

Throughout this section, we use a superscript $^{(j)}$ to denote a quantity computed *without* the jth sample.

To facilitate *efficient* computation of the matrix jackknife, the core observation is that $\operatorname{range}(\boldsymbol{U})$ contains $\operatorname{range}(\boldsymbol{X}^{(j)})$ for all j. To see why this is the case, observe that the columns of $\boldsymbol{Y}^{(j)}$ are a subset of the columns of $\boldsymbol{Y}$; cf. step 2 in section 3.1. Thus,

$$\operatorname{range}\left(\boldsymbol{X}^{(j)}\right) = \operatorname{range}\left(\boldsymbol{U}^{(j)}\right) = \operatorname{range}\left(\boldsymbol{Q}^{(j)}\right) \subseteq \operatorname{range}(\boldsymbol{Q}) = \operatorname{range}(\boldsymbol{U}).$$

A similar statement holds for the right singular vectors. Indeed, since $\boldsymbol{C}^{(j)} = \left(\boldsymbol{Q}^{(j)}\right)^* \boldsymbol{A}$,

$$\operatorname{range}\left(\left(\boldsymbol{X}^{(j)}\right)^*\right) = \operatorname{range}(\boldsymbol{V}^{(j)}) = \operatorname{range}\left(\left(\boldsymbol{C}^{(j)}\right)^*\right) \subseteq \operatorname{range}\left(\boldsymbol{C}^*\right) = \operatorname{range}(\boldsymbol{V}).$$

Because $\operatorname{range}\left(\boldsymbol{X}^{(j)}\right) \subseteq \operatorname{range}(\boldsymbol{U})$ and $\operatorname{range}\left(\left(\boldsymbol{X}^{(j)}\right)^*\right) \subseteq \operatorname{range}(\boldsymbol{V})$, each replicate $\boldsymbol{X}^{(j)}$ can be factorized as

$$\boldsymbol{X}^{(j)} = \boldsymbol{U}\boldsymbol{T}_j\boldsymbol{V}^* \quad \text{for some core matrix } \boldsymbol{T}_j \in \mathbb{K}^{s \times s}. \tag{3.1}$$

This observation yields immense savings: rather than manipulating the $d_1 \times d_2$ matrix $\boldsymbol{X}^{(j)}$, we can work with the much smaller $s \times s$ matrix $\boldsymbol{T}_j$. With these preparations in place, we arrive at a computationally efficient expression for the jackknife variance estimate:

$$\operatorname{Jack}^2(\boldsymbol{X}) = \sum_{j=1}^{s} \left\| \boldsymbol{X}^{(j)} - \boldsymbol{X}^{(\cdot)} \right\|_{\mathrm{F}}^2 = \sum_{j=1}^{s} \left\| \boldsymbol{T}_j - \overline{\boldsymbol{T}} \right\|_{\mathrm{F}}^2 \quad \text{where} \quad \overline{\boldsymbol{T}} := \frac{1}{s} \sum_{j=1}^{s} \boldsymbol{T}_j. \tag{3.2}$$

We have used the unitary invariance of the Frobenius norm $\|\cdot\|_{\mathrm{F}}$.

Algorithm 3.1 Randomized SVD with jackknife variance estimate

Input: $\boldsymbol{A} \in \mathbb{K}^{d_1 \times d_2}$ to be approximated, approximation rank s, subspace iteration steps q
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 $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$, variance estimate Jack = Jack($\boldsymbol{X}$)

procedure SVDWITHJACKKNIFE($\boldsymbol{A}$, s, q)
  $\boldsymbol{\Omega} \leftarrow$ `randn`(d_2, s)
  $\boldsymbol{Y} \leftarrow \boldsymbol{A}(\boldsymbol{A}^*\boldsymbol{A})^q\boldsymbol{\Omega}$ ▷ Repeated matrix multiplication
  $(\boldsymbol{Q}, \boldsymbol{R}) \leftarrow$ `qr`$(\boldsymbol{Y}$, `'econ'`$)$ ▷ Economy QR Factorization
  $\boldsymbol{C} \leftarrow \boldsymbol{Q}^*\boldsymbol{A}$
  $(\widetilde{\boldsymbol{U}}, \boldsymbol{\Sigma}, \boldsymbol{V}) \leftarrow$ `svd`$(\boldsymbol{C}$, `'econ'`$)$
  $\boldsymbol{U} \leftarrow \boldsymbol{Q}\widetilde{\boldsymbol{U}}$
  for $j = 1$ to s **do**
    $\boldsymbol{R}' \leftarrow \boldsymbol{R}(:, [1:(j-1),\ (j+1):s])$ ▷ $\boldsymbol{R}'$ is $\boldsymbol{R}$ without its jth column
    $(\widetilde{\boldsymbol{Q}}, \widetilde{\boldsymbol{R}}) \leftarrow$ `qr`$(\boldsymbol{R}'$, `'econ'`$)$
    $\boldsymbol{T}_j \leftarrow \widetilde{\boldsymbol{U}}^*\widetilde{\boldsymbol{Q}}\widetilde{\boldsymbol{Q}}^*\widetilde{\boldsymbol{U}}\boldsymbol{\Sigma}$
  end for
  $\overline{\boldsymbol{T}} \leftarrow s^{-1}\sum_{j=1}^{s} \boldsymbol{T}_j$
  Jack $\leftarrow \left(\sum_{j=1}^{s} \left\|\boldsymbol{T}_j - \overline{\boldsymbol{T}}\right\|_{\mathrm{F}}^2\right)^{1/2}$
  return $\boldsymbol{U}$, $\boldsymbol{\Sigma}$, $\boldsymbol{V}$, Jack
end procedure

Fortunately, the core matrices $\boldsymbol{T}_j$ can also be computed efficiently. Algorithm 3.1 gives an implementation that forms each $\boldsymbol{T}_j$ in $\mathcal{O}(s^3)$ operations. The jackknife variance estimate Jack($\boldsymbol{X}$) is obtained in $\mathcal{O}(s^4)$ operations in total, *independent of either of the dimensions d_1 and d_2 of the original problem.* In appendix A.1, we describe how to use downdating to reduce the total cost to $\mathcal{O}(s^3)$ operations.

3.3. Extensions. We can also apply the matrix jackknife to quantities derived from the randomized SVD computation. Indeed, Algorithm 3.1 provides a representation of each of the replicates: $\boldsymbol{X}^{(j)} = \boldsymbol{U}\boldsymbol{T}_j\boldsymbol{V}^*$. If we further extract an SVD of $\boldsymbol{T}_j$, say $\boldsymbol{T}_j = \boldsymbol{U}_j\boldsymbol{\Sigma}_j\boldsymbol{V}_j^*$, then we obtain an implicit representation of a truncated SVD of each jackknife replicate

$$\boldsymbol{X}^{(j)} = \big(\boldsymbol{U}\boldsymbol{U}_j\big)\boldsymbol{\Sigma}_j\big(\boldsymbol{V}\boldsymbol{V}_j\big)^*.$$

This formula allows us to obtain variance estimates for matrix- and scalar-valued quantities that can be determined from the core matrix $\boldsymbol{T}_j$ or its singular value decomposition. We focus on variance estimates for two examples: projectors onto leading singular subspaces and entries of the output approximation. In a similar fashion, we can develop variance estimates for other quantities such as projectors onto other singular subspaces, truncations of the low-rank approximation to smaller rank, and entries of the eigenvectors.

3.3.1. Projectors onto leading singular subspaces. As an example, we consider how to estimate the variance of the ith left singular projector $\boldsymbol{\Pi}_i^{\mathrm{L}}(\boldsymbol{X})$. Note the identity

$$\big(\boldsymbol{\Pi}_i^{\mathrm{L}}(\boldsymbol{X})\big)^{(j)} = \boldsymbol{\Pi}_i^{\mathrm{L}}\Big(\boldsymbol{X}^{(j)}\Big) = \boldsymbol{\Pi}_i^{\mathrm{L}}\big(\boldsymbol{U}\boldsymbol{T}_j\boldsymbol{V}^*\big) = \boldsymbol{U}\boldsymbol{\Pi}_i^{\mathrm{L}}(\boldsymbol{T}_j)\boldsymbol{U}^*.$$

In view of this fact, the jackknife variance estimate for $\boldsymbol{\Pi}_i^{\mathrm{L}}(\boldsymbol{T}_j)$ can be computed in $\mathcal{O}(s^4)$ operations by replacing $\boldsymbol{T}_j$ with $\boldsymbol{\Pi}_i^{\mathrm{L}}(\boldsymbol{T}_j) = \boldsymbol{U}_j(:,i)\boldsymbol{U}_j^*(:,i)$ in Algorithm 3.1.

3.3.2. Entries of the approximation. We can extract the (m,n)-entry $E_j := \left(\boldsymbol{X}^{(j)}\right)_{mn}$ is directly from the core matrix $\boldsymbol{T}_j$ via the expression $E_j = \boldsymbol{U}(m,:)\boldsymbol{T}_j\boldsymbol{V}(n,:)^*$. This formula allows us to form Tukey's variance estimator (2.3) to be computed for any given entry of the matrix $\boldsymbol{X}$ in $\mathcal{O}(s^3)$ time from the core matrices $\boldsymbol{T}_1, \dots, \boldsymbol{T}_s$.

4. Case study: Single-view Nyström approximation. In this section, we apply the matrix jackknife variance estimate to another randomized low-rank approximation algorithm, the single-view Nyström approximation [10, 24] of a positive-semidefinite matrix $\boldsymbol{A} \in \mathbb{K}^{d\times d}$. We introduce the single-view Nyström in section 4.1 before discussing *efficient* computation of the jackknife variance estimate in section 4.2. Extensions are briefly discussed in section 4.3.

4.1. Single-view Nyström approximation. Given an arbitrary test matrix $\boldsymbol{\Omega} \in \mathbb{K}^{d\times s}$, the Nyström approximation of a positive-semidefinite matrix $\boldsymbol{A}$ takes the form

$$\boldsymbol{X} = \boldsymbol{A\Omega}\left(\boldsymbol{\Omega A\Omega}\right)^{\dagger}\left(\boldsymbol{A\Omega}\right)^*. \tag{4.1}$$

The Nyström approximation $\boldsymbol{X}$ is the best positive semidefinite approximation to $\boldsymbol{A}$ in the span of the columns of the sketch $\boldsymbol{A\Omega}$.

Early machine learning applications of the Nyström method [5, 28] exclusively used randomized column sampling; that is, the columns of $\boldsymbol{\Omega}$ were required to be columns of the identity matrix. The method is now used with more general random test matrices [10, 24]. We shall restrict attention to the case when $\boldsymbol{\Omega}$ is populated with statistically independent standard Gaussian entries. *A priori* error bounds for the spectral norm error $\mathbb{E}\|\boldsymbol{A}-\boldsymbol{X}\|$ in this setting are given by [9, Prop. 2.2].

An appealing feature of Nyström approximation is that it only accesses the matrix $\boldsymbol{A}$ via the single matrix product $\boldsymbol{A\Omega}$. The product $\boldsymbol{A\Omega}$ can be evaluated using only a single pass over the matrix $\boldsymbol{A}$. This observation means the Nyström approximation can be computed in a streaming setting where $\boldsymbol{A}$ is presented as a sum of updates which must be processed on the spot and discarded [24].

We can represent the Nyström approximation $\boldsymbol{X}$ using an economy eigenvalue decomposition:

$$\boldsymbol{X} = \boldsymbol{V\Lambda V}^* \quad \text{with} \quad \boldsymbol{V} \in \mathbb{K}^{d\times s} \quad \text{and} \quad \boldsymbol{\Lambda} \in \mathbb{R}^{s\times s}. \tag{4.2}$$

Here, $\boldsymbol{\Lambda}$ is a diagonal matrix containing the nonzero eigenvalues of $\boldsymbol{X}$. The matrix $\boldsymbol{V}$ contains the associated eigenvectors. We construct the single-view Nyström approximation in the following way to facilitate efficient computation of the jackknife variance estimate:

1. Draw a random sketching matrix $\boldsymbol{\Omega} \in \mathbb{K}^{n\times s}$ with independent standard Gaussian entries.
2. Compute the product $\boldsymbol{Y} = \boldsymbol{A\Omega}$.
3. Extract an orthonormal basis $\boldsymbol{Q} = \mathrm{orth}(\boldsymbol{Y})$ by economy QR factorization $\boldsymbol{Y} = \boldsymbol{QR}$.
4. Compute $\boldsymbol{B} = \boldsymbol{\Omega}^*\boldsymbol{Y}$ and Cholesky factorize $\boldsymbol{B} = \boldsymbol{C}^*\boldsymbol{C}$.
5. Obtain a singular value decomposition $\boldsymbol{RT}^{-1} = \boldsymbol{U\Sigma Z}^*$.
6. Set $\boldsymbol{\Lambda} := \boldsymbol{\Sigma}^2$ and $\boldsymbol{V} := \boldsymbol{QU}$.

Algorithm 4.1 provides a numerically robust implementation of this template.

4.2. Matrix jackknife for the single-view Nyström approximation. The setup for the single-value Nyström approximation is similar to the randomized SVD in section 3.2. We regard the Nyström approximation $\boldsymbol{X}$ defined in (4.1) as a function

$$\boldsymbol{X} = \boldsymbol{X}(\boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_s) \quad \text{where} \quad \boldsymbol{\Omega} = \begin{bmatrix}\boldsymbol{\omega}_1 & \cdots & \boldsymbol{\omega}_s\end{bmatrix}.$$

Algorithm 4.1 Single-view Nyström approximation with jackknife variance 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 variance estimate Jack = Jack($\boldsymbol{X}$)

procedure NYSTRÖMWITHJACKKNIFE($\boldsymbol{A}$, s) ▷ $\boldsymbol{A} \in \mathbb{K}^{d\times d}$ positive semidefinite
  $\boldsymbol{\Omega} \leftarrow$ `randn`(d, s)
  $\boldsymbol{Y} \leftarrow \boldsymbol{A}\boldsymbol{\Omega}$
  $\nu \leftarrow \epsilon_{\text{mach}} \|\boldsymbol{Y}\|$ and $\boldsymbol{Y} \leftarrow \boldsymbol{Y} + \nu\boldsymbol{\Omega}$ ▷ Shift for numerical stability
  $(\boldsymbol{Q}, \boldsymbol{R}) \leftarrow$ `qr`$(\boldsymbol{Y}$, `'econ'`$)$ ▷ Economy QR Factorization
  $\boldsymbol{B} \leftarrow \boldsymbol{\Omega}^*\boldsymbol{Y}$
  $\boldsymbol{C} \leftarrow$ `chol`$((\boldsymbol{B} + \boldsymbol{B}^*)/2)$
  $(\boldsymbol{U}, \boldsymbol{\Sigma}, \sim) \leftarrow$ `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
  $\boldsymbol{V} \leftarrow \boldsymbol{Q}\boldsymbol{U}$
  for $j = 1$ to s **do**
    $\boldsymbol{R}_j \leftarrow \boldsymbol{R}(:, [1:(j-1),\ (j+1):s])$ ▷ $\boldsymbol{R}'$ is $\boldsymbol{R}$ without its jth column
    $\boldsymbol{B}_j \leftarrow \boldsymbol{B}([1:(j-1),\ (j+1):s], [1:(j-1),\ (j+1):s])$
    $\boldsymbol{T}_j \leftarrow \boldsymbol{U}^*\boldsymbol{R}_j\boldsymbol{B}_j^{-1}\boldsymbol{R}_j^*\boldsymbol{U}$
  end for
  $\overline{\boldsymbol{T}} \leftarrow s^{-1}\sum_{j=1}^{s}\boldsymbol{T}_j$
  Jack $\leftarrow \sqrt{\sum_{j=1}^{s}\left\|\boldsymbol{T}_j - \overline{\boldsymbol{T}}\right\|_{\mathrm{F}}^2}$
  return $\boldsymbol{V}$, $\boldsymbol{\Lambda}$, Jack
end procedure

Introduce jackknife replicates

$$\boldsymbol{X}^{(j)} = \boldsymbol{X}(\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_{j-1}, \boldsymbol{\omega}_{j+1}, \ldots, \boldsymbol{\omega}_s) \quad \text{for each } j = 1, 2, \ldots, s.$$

As in section 4.3, the critical observation is that the range of each of the replicates is contained in the range of the matrix $\boldsymbol{V}$ defining $\boldsymbol{X}$ in (4.2). Because of this, each of the replicates takes the form

$$\boldsymbol{X}^{(j)} = \boldsymbol{V}\boldsymbol{T}_j\boldsymbol{V}^* \quad \text{where } \boldsymbol{T}_j \in \mathbb{K}^{s\times s}.$$

It follows that jackknife variance estimate simplifies to

$$\mathrm{Jack}^2(\boldsymbol{X}) = \sum_{j=1}^{s}\left\|\boldsymbol{X}^{(j)} - \boldsymbol{X}^{(\cdot)}\right\|_{\mathrm{F}}^2 = \sum_{j=1}^{s}\left\|\boldsymbol{T}_j - \overline{\boldsymbol{T}}\right\|_{\mathrm{F}}^2 \quad \text{where} \quad \overline{\boldsymbol{T}} := \frac{1}{s}\sum_{j=1}^{s}\boldsymbol{T}_j.$$

Algorithm 4.1 constructs the random Nyström approximation $\boldsymbol{X}$ in factored form (4.2) along with the reduced jackknife variance estimate $\boldsymbol{W}$. This algorithm uses several tricks, drawn from [14], to improve its numerical stability. Just like Algorithm 3.1 for the randomized SVD, Algorithm 4.1 requires $\mathcal{O}(s^4)$ operations to compute the variance estimate, independent of the dimension d of the matrix $\boldsymbol{A}$. Appendix A discusses how to use downdating to reduce the total cost to $\mathcal{O}(s^3)$ operations.

4.3. Extensions. One can easily extend Algorithm 4.1 to obtain variance estimates for derived quantities. Examples of such derived quantities include truncations of the Nyström approximation to rank $r < s$, estimates of spectral projectors, and entries of the ap-

proximation. We omit the details because they are similar to the case of the randomized SVD (section 3.3).

5. Numerical experiments. In this section, we showcase numerical examples that demonstrate the effectiveness of the matrix jacknife variance estimate for the randomized SVD and the single-view Nyström approximation. All numerical results will use real numbers ($\mathbb{K} = \mathbb{R}$).

5.1. Test matrices. To evaluate the performance of the jackknife for matrices with different spectral characteristics, we consider the following synthetic test matrices from [24, Sec. 5]:

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

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

Here, $\xi, s \in \mathbb{R}$ are parameters, and $\boldsymbol{G} \in \mathbb{R}^{d\times d}$ is a standard Gaussian matrix. Our focus on diagonal test matrices is justified by the observation that the randomized SVD, the single-view Nyström approximation, and their jackknife variance estimates depend only on the singular values. This statement relies on the fact that the Gaussian test matrix $\boldsymbol{\Omega}$ is rotationally invariant. We also consider matrices from application domains:

- **Velocity.** We consider a matrix $\boldsymbol{A} \in \mathbb{R}^{25096\times 1000}$ whose columns are snapshots of the streamwise and transverse velocity and pressure from simulations of a fluid flow past a cylinder. We thank Beverley McKeon and Sean Symon for this data.
- **Kernel matrix.** We study an RBF kernel matrix $\boldsymbol{A} \in \mathbb{R}^{1599\times 1599}$ assembled from the red wine data in the wine quality dataset [2] from the UCI Machine Learning Repository. The data is standardized, and the bandwidth is $\sigma = 10$.

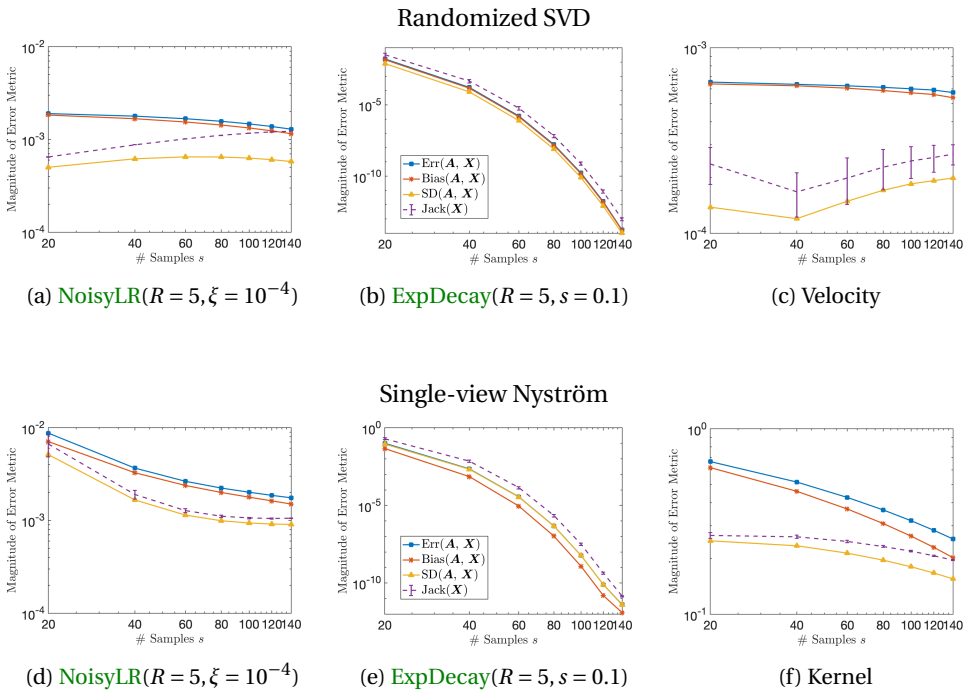
5.2. Evaluation. We test the randomized SVD and the single-view Nyström approximation with variety of sketching dimensions $s \in \{20, 40, \dots, 140\}$. We use $q = 2$ steps of subspace iteration for the randomized SVD.

We wish to compare the jackknife standard deviation estimate $\operatorname{Jack}(\boldsymbol{X})$ with the bias $\|\boldsymbol{A} - \mathbb{E}\boldsymbol{X}\|_{\mathrm{F}}$, standard deviation $(\operatorname{Var}(\boldsymbol{X}))^{1/2}$, and root-mean-square error $(\mathbb{E}\|\boldsymbol{A}-\boldsymbol{X}\|_{\mathrm{F}}^2)^{1/2}$. Since these quantities are not readily available, we estimate them by a Monte Carlo procedure. Letting $\boldsymbol{X}[1],\dots,\boldsymbol{X}[m]$ denote the outputs of m independent executions randomized SVD or single-view Nyström approximation, we define estimates

$$\begin{aligned}
\operatorname{Err}(\boldsymbol{A},\boldsymbol{X}) &\coloneqq \frac{1}{\|\boldsymbol{A}\|_{\mathrm{F}}}\left(\frac{1}{m}\sum_{i=1}^{m}\|\boldsymbol{A}-\boldsymbol{X}[i]\|_{\mathrm{F}}^2\right)^{1/2} &&\approx \frac{\left(\mathbb{E}\|\boldsymbol{A}-\boldsymbol{X}\|_{\mathrm{F}}^2\right)1/2}{\|\boldsymbol{A}\|_{\mathrm{F}}},\\
\operatorname{Bias}(\boldsymbol{A},\boldsymbol{X}) &\coloneqq \frac{1}{\|\boldsymbol{A}\|_{\mathrm{F}}}\left\|\boldsymbol{A}-\frac{1}{m}\sum_{i=1}^{m}\boldsymbol{X}[i]\right\|_{\mathrm{F}} &&\approx \frac{\|\boldsymbol{A}-\mathbb{E}\boldsymbol{X}\|_{\mathrm{F}}}{\|\boldsymbol{A}\|_{\mathrm{F}}},\\
\operatorname{SD}(\boldsymbol{A},\boldsymbol{X}) &\coloneqq \frac{1}{\|\boldsymbol{A}\|_{\mathrm{F}}}\left(\frac{1}{m}\sum_{i=1}^{m}\left\|\boldsymbol{X}[i]-\frac{1}{m}\sum_{j=1}^{m}\boldsymbol{X}[j]\right\|_{\mathrm{F}}^2\right)^{1/2} &&\approx \frac{(\operatorname{Var}(\boldsymbol{X}))^{1/2}}{\|\boldsymbol{A}\|_{\mathrm{F}}}.
\end{aligned} \tag{5.1}$$

We use $m = 1000$ trials to compute these quantities.

We shall also use analogs of the the approximate root-mean-square error, standard deviation, and bias (5.1) to evaluate the accuracy of spectral projectors. For example, $\operatorname{Err}(\boldsymbol{\Pi}_j(\boldsymbol{A}),\boldsymbol{\Pi}_j(\boldsymbol{X}))$ and $\operatorname{Jack}(\boldsymbol{\Pi}_j(\boldsymbol{X}))$ denotes the approximate root-mean-square error and standard deviation estimate for the dominant left singular projector $\boldsymbol{\Pi}_j(\boldsymbol{X})$ as an approximation for $\boldsymbol{\Pi}_j(\boldsymbol{A})$.



(a) NoisyLR($R = 5, \xi = 10^{-4}$) (b) ExpDecay($R = 5, s = 0.1$) (c) Velocity

(d) NoisyLR($R = 5, \xi = 10^{-4}$) (e) ExpDecay($R = 5, s = 0.1$) (f) Kernel

Fig. 1: **Jackknife for randomized SVD and single-view Nyström.** These panels compare the error metrics (5.1) to the jackknife standard deviation estimate Jack($\boldsymbol{X}$) for the randomized SVD (top) and and single-view Nyström approximation (bottom). Error bars show plus-or-minus one standard deviation. The jackknife standard deviation estimate closely tracks the standard deviation estimate, and it lies within an order of magnitude of the error estimate in all cases.

5.3. Summary of experiments. We ran several experiments to test the performance of the matrix jackknife on different examples. These experiments are summarized in the following figures:

- In Figure 1, we apply the jackknife standard deviation estimate, Jack($\boldsymbol{X}$), to the full approximation matrix returned by the randomized SVD and the single-view Nyström approximation.
- In Figure 2, we apply the jackknife to study both a stable and unstable spectral projector estimated via the single-view Nyström approximation.

In all figures, the error bars on the jackknife curves represent plus-or-minus one standard deviation. Additional numerical experiments are included in the supplementary materials (section **??**).

As a final experiment, we apply the techniques described in section 3.3.2 to obtain variance estimates for the absolute values of the entries of a singular vector computed by the randomized SVD. (The absolute value is introduced to avoid sign ambiguities.) We illustrate the results in Figure 3 on data from a singular vector computed from the fluid velocity data. For this experiment alone, we use $q = 0$ steps of subspace iteration so that the error is large enough to visualize.

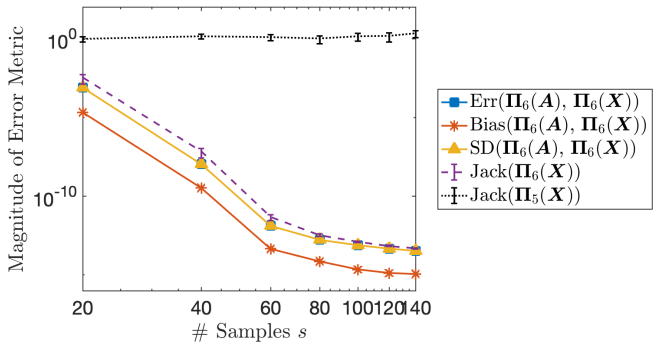


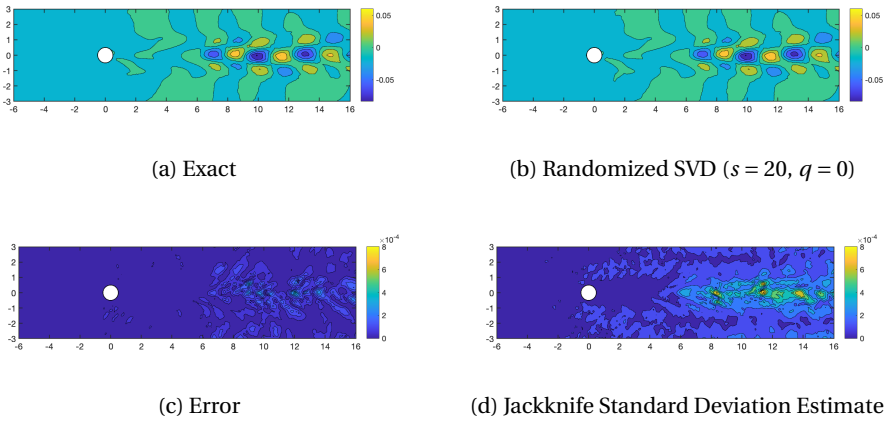
Fig. 2: **Jacknife for spectral projectors.** Demonstration of the jackknife standard deviation estimate as a measure of stability for spectral projectors computed using the single-view Nyström approximation. For the matrix $\boldsymbol{A} = \text{ExpDecay}(R = 5, s = 0.25)$, the largest eigenvalue has multiplicity five, so the spectral projector $\mathbf{\Pi}_5(\boldsymbol{A})$ is ill-defined while the spectral projector $\mathbf{\Pi}_6(\boldsymbol{A})$ is stable. The jackknife standard deviation estimator diagnoses this difference, with Jack($\mathbf{\Pi}_5(\boldsymbol{X})$) remaining large while Jack($\mathbf{\Pi}_6(\boldsymbol{X})$) and Err($\mathbf{\Pi}_6(\boldsymbol{A})$, $\mathbf{\Pi}_6(\boldsymbol{X})$) decrease as the approximation is refined.

5.4. Discussion. The jackknife standard deviation estimate, Jack($\boldsymbol{X}$), does a very good job of tracking the true standard deviation SD($\boldsymbol{A}$, $\boldsymbol{X}$) for low-rank approximations produced by the randomized SVD and the single-view Nyström approximation (Figure 1). The average jackknife standard deviation estimate, Jack($\boldsymbol{X}$), lies within an order of magnitude of the standard deviation SD($\boldsymbol{A}$, $\boldsymbol{X}$) in all of experiments. This suggests that, while the jackknife overestimates the true variance on average, the overestimate is consistently modest.

For the test matrices considered, we observe that the jackknife standard deviation estimate Jack($\boldsymbol{X}$) is also a reasonably good proxy for the root-mean-square error, Err($\boldsymbol{A}$, $\boldsymbol{X}$). For both the randomized SVD and the single-view Nyström approximation (Figure 1), the bias, Bias($\boldsymbol{A}$, $\boldsymbol{X}$), and the root-mean-square error, Err($\boldsymbol{A}$, $\boldsymbol{X}$), sometimes exceed the jackknife standard deviation estimator Jack($\boldsymbol{X}$), but never by more than an order of magnitude. The results for spectral projectors computed by the single-view Nyström approximation (Figure 2) and the singular projectors computed by the randomized SVD (Figure **??**) were even better; the jackknife standard deviation estimate always dominated the error in our tests. This reinforces our assertion that the variance estimator could be used as an order-of-magnitude error estimate. Certainly, our experiments demonstrate that if the standard deviation estimate is higher than a user-specified error tolerance, then it is reasonable to conclude the number of samples needs to be increased.

Figure 2 demonstrates the utility of the variance estimator as a more qualitative diagnostic. We consider the fifth and sixth spectral projectors of a matrix for which the largest eigenvalue has multiplicity five. This degeneracy makes the fifth spectral projector ill-defined. The jackknife properly diagnoses this problem, with the variance estimate Jack($\mathbf{\Pi}_5(\boldsymbol{X})$) remaining high and the variance estimate Jack($\mathbf{\Pi}_6(\boldsymbol{X})$) decreasing rapidly with increasing sketch dimension.

Figure 3 shows the Tukey jackknife standard deviation estimate $\left(\widehat{\text{Var}}\right)^{1/2}$ (where $\widehat{\text{Var}}$ is as in (2.3)) for each of the absolute values of the entries of the fifth left singular vector of the velocity matrix, as computed using the efficient algorithms developed in section 3. The jackknife gave an estimate for the accuracy of the entries of the singular vector computed



(a) Exact

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

(c) Error

(d) Jackknife Standard Deviation Estimate

Fig. 3: **Jackknife for singular vector.** These panels assess the entrywise errors in the absolute the streamwise velocity from the fifth left singular vector of the velocity test matrix computed. Panel (a) shows the exact answer, which is compared against the estimate produced by the randomized SVD in panel (b). Panels (c) and (d) display the error and the the Tukey jacknife standard deviation estimate. The jackknife estimate modestly overestimates the error, but correctly shows where the errors are localized.

by the randomized SVD which was accurate up to a small multiple. Visualizing the error profile produces by the jackknife yields a descriptive picture of where the errors produced by the algorithm are localized.

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

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

Often, it is more desirable to have error or variance estimates for Schatten norms $|||\cdot|||_p$ with $p > 2$. Here, the Schatten p-norm is

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

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 section 2.2 with the additional stipulation that the samples $\omega_1, \ldots, \omega_s$ take values in a Polish space Ω.

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

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

Define the Schatten p-norm variance estimate

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

This quantity seeks to approximate

$$\mathrm{Jack}_p^p(\boldsymbol{X}_{s-1}) \approx \mathbb{E} \left\| \boldsymbol{X}_{s-1} - \mathbb{E} \boldsymbol{X}_{s-1} \right\|_p^p.$$

A matrix generalization of the Efron–Stein–Steele inequality [19, Thm. 4.2] shows this jackknife variance estimate overestimates the Schatten p-norm variance in the sense

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

The techniques we introduce in sections 3 and 4 can be extended in a natural way to compute $\mathrm{Jack}_p(\boldsymbol{X}_{s-1})$ efficiently for the randomized SVD and the single-view Nyström approximation.

7. Related work. We now compare the matrix jackknife to other posterior diagnostics for randomized matrix computations in the literature.

7.1. Norm estimates. In some settings, the approximation error $\|\boldsymbol{A} - \boldsymbol{X}\|$ can be computed in a similar operation count to the randomized matrix algorithm. For instance, this is the case if $\boldsymbol{A}$ is a dense matrix stored in RAM, $\|\cdot\| = \|\cdot\|_{\mathrm{F}}$ is the Frobenius norm, and the algorithm is the randomized SVD.

For settings where the approximation error is expensive to compute, several authors have proposed using randomized norm estimates [18, §§4–5] as posterior error estimates for randomized matrix computations. Rather than assess the error of an approximation $\boldsymbol{X}$ to a matrix $\boldsymbol{A}$ on average, norm estimates seek to assess the norm of $\boldsymbol{A} - \boldsymbol{X}$ for a *specific instantiation* of $\boldsymbol{X}$ by computing matrix–vector products with $\boldsymbol{A} - \boldsymbol{X}$. Randomized norm estimators and the jackknife provide complementary perspectives on the output of a matrix computation: Norm estimates bound the error for a specific instantiation of the random approximation $\boldsymbol{X}$, whereas the jackknife estimates the variance which *can* be a good proxy for the mean-square error. There are several settings where the jackknife possesses a distinct advantage, such as when the fresh matrix–vector products with $\boldsymbol{A}$ are computationally costly to obtain or for assessing the error of derived quantities like spectral projectors.

7.2. Bootstrap resampling. In recent work, Lopes and collaborators have applied bootstrap resampling to randomized matrix algorithms [15, 16, 17]. The work [15] that is closest to ours constructs bootstrap error estimates for a *linear* sketched SVD algorithm. As presented in [15], this approach has several limitations:

1. The linearity of the algorithm is essential for the analysis of the bootsrap; it is not obvious that the bootstrap can be extended to nonlinear algorithms such as the randomized SVD and single-view Nyström approximation considered in our work.
2. Each bootstrap iteration requires $\mathcal{O}(d_2 s^2)$ operations for an s-column sketching matrix. By contrast, the most efficient implementations of our algorithm require time $\mathcal{O}(s^2)$ per jackknife replicate, independent of either of the dimensions of $\boldsymbol{A}$. The techniques from our paper can be used to accelerate each bootstrap iteration to $\mathcal{O}(s^3)$ operations.

Modulo these two significant limitations, the bootstrap does give somewhat more fine-grained information than the matrix jackknife. Indeed, the bootstrap provides quantile estimates for the errors of all the singular values and vectors. These estimates con-

verge in an appropriate asymptotic sense to the true quantiles as the number of samples increases.

8. Conclusion. Jackknife resampling gives a flexible, data-driven methodology for estimating the variance of a randomized matrix approximation. The method has nonasymptotic guarantees (Theorem 2.2), and it can be efficiently computed for several widely used algorithms. In the common situation where the variance makes up a large fraction of the error in a bias–variance decomposition, the matrix jackknife provides a descriptive portrait of the error in the approximation.

We anticipate the matrix jackknife can be efficiently computed for other low-rank approximation algorithms. For some algorithms (e.g., the single-view SVD of [25]), this is straightforward. For other procedures, like CUR factorizations [27], efficiently computing the variance estimate may be more involved.

There also may be applications of this matrix jackknife approach to statistical problems such as low-rank matrix completion. In contrast to linear algebraic applications where the randomness is algorithmically generated, randomness in statistical applications usually comes from problem data that is assumed to be drawn from a distribution. While leave-one-out techniques have been used in the *analysis* of matrix completion techniques [4], we are unaware of work using jackknife resampling in an efficient way as a posterior estimate for the variance of the recovered low-rank matrix. For different methods for low-rank matrix completion (e.g., nuclear norm minimization [22], alternating minimization [13], and projection-based methods [20]), it appears challenging to efficiently compute (or descriptively bound) the difference between jackknife replicates without running the algorithm many times.

Appendix A. Faster algorithms for the jackknife variance estimate. In this appendix, we present improvements to the efficient jackknife procedures presented for the randomized SVD (section A.1) and single-view Nyström approximation (section A.2).

A.1. Randomized singular value decomposition. We begin with an improvement to Algorithm 3.1 for computing Jack($\boldsymbol{X}$) for the randomized SVD. The improved algorithm from this section forms core matrix $\boldsymbol{T}_1, \dots, \boldsymbol{T}_s$ in $\mathcal{O}(s^2)$ operations each. The jackknife estimate Jack($\boldsymbol{X}$) can then be determined in $\mathcal{O}(s^3)$ operations by evaluating (3.2). After this, we discuss how to factor each core matrix $\boldsymbol{T}_j = \boldsymbol{U}_j \boldsymbol{\Sigma}_j \boldsymbol{V}_j^*$ by an SVD in $\mathcal{O}(s^2)$ operations each. This factorization accelerates the computation of variance estimates for singular projectors (section 3.3.1) to $\mathcal{O}(s^3)$ operations.

Instate the notation of Algorithm 3.1. The key insight shall be that each $\boldsymbol{T}_j$ is a rank-one update to $\boldsymbol{\Sigma}$, i.e., $\boldsymbol{T}_j = \boldsymbol{\Sigma} + \boldsymbol{x}_j \boldsymbol{y}_j^*$. To see this, let $\boldsymbol{R}'$ denote $\boldsymbol{R}$ without its jth column. Suppose we compute a (full) QR factorization of $\boldsymbol{R}'$, which we conformally partition as

$$\boldsymbol{R}' = \begin{bmatrix} \widetilde{\boldsymbol{Q}} & \boldsymbol{q} \end{bmatrix} \begin{bmatrix} \widetilde{\boldsymbol{R}} \\ \boldsymbol{0}^* \end{bmatrix}. \tag{A.1}$$

Since the first factor is unitary, we have that $\widetilde{\boldsymbol{Q}}\widetilde{\boldsymbol{Q}}^* = \mathbf{I} - \boldsymbol{q}\boldsymbol{q}^*$. Plugging this into line 11 of Algorithm 3.1, we see that

$$\boldsymbol{T}_j = \widetilde{\boldsymbol{U}}^* \widetilde{\boldsymbol{Q}} \widetilde{\boldsymbol{Q}}^* \widetilde{\boldsymbol{U}} \boldsymbol{\Sigma} = \widetilde{\boldsymbol{U}}^* (\mathbf{I} - \boldsymbol{q}\boldsymbol{q}^*) \widetilde{\boldsymbol{U}} \boldsymbol{\Sigma} = \boldsymbol{\Sigma} + \left(-\widetilde{\boldsymbol{U}}^* \boldsymbol{q}\right)\left(\boldsymbol{\Sigma}\widetilde{\boldsymbol{U}}^* \boldsymbol{q}\right)^* =: \boldsymbol{\Sigma} + \boldsymbol{x}_j \boldsymbol{y}_j^*. \tag{A.2}$$

Each $\boldsymbol{T}_j$ is a rank-one modification of $\boldsymbol{\Sigma}$, as claimed.

We now describe how to form $\boldsymbol{T}_j$ in $\mathcal{O}(s^2)$ operations using this insight. First observe that the matrix $\boldsymbol{R}'$ is upper Hessenberg, so its QR factorization can be computed as the product of at most s Givens rotations. By multiplying these Givens rotations rotations into

the standard basis vector $\mathbf{e}_s$, we obtain the vector $\boldsymbol{q}$. Finally, we form $\boldsymbol{T}_j$ by evaluating (A.2). Accounting for these steps, the total operation count is $\mathcal{O}(s^2)$, as desired. By forming each core matrix $\boldsymbol{T}_j$ in $\mathcal{O}(s^2)$ operations, the total operation count of computing Jack($\boldsymbol{X}$) has been reduced to $\mathcal{O}(s^3)$.

Finally, we comment on how to obtain variance estimators for derived quantities like singular projectors in a reduced $\mathcal{O}(s^3)$ operations. The essential step will be obtaining a singular value decomposition of each $\boldsymbol{T}_j$ matrix. Fortunately, the problem of computing the SVD of a diagonal matrix modified by a rank-one matrix can be achieved in $\mathcal{O}(s^2)$ operations [11, Part I]. Thus, an SVD of each T_j can be obtained by applying this procedure in concert with the rank-one update formula (A.2).

A.2. Single-view Nyström Approximation. We now discuss an improvement to Algorithm 4.1 which allows the variance estimate for the single-view Nyström approximation to be computed in $\mathcal{O}(s^3)$ operations. The first observation is that each $\boldsymbol{T}_j$ can be written as

$$\boldsymbol{T}_j = \boldsymbol{U}^* \left(\boldsymbol{R}_j \boldsymbol{B}_j^{-1} \boldsymbol{R}_j^* \right) \boldsymbol{U} =: \boldsymbol{U}^* \boldsymbol{S}_j \boldsymbol{U}$$

Since $\boldsymbol{U}$ and $\boldsymbol{U}^*$ are unitary, left or right multiplication by them do not effect the Frobenius or other Schatten norms. As such, it is sufficient to develop an efficient algorithm to determine each $\boldsymbol{S}_j$ matrix.

As in section A.1, we shall realize that each $\boldsymbol{S}_j$ is a rank-one modification to a single matrix: In this case,

$$\boldsymbol{S}_j = \boldsymbol{R}\boldsymbol{B}^{-1}\boldsymbol{R}^* - \boldsymbol{x}_j \boldsymbol{x}_j^*. \tag{A.3}$$

To show this, consider $\boldsymbol{R}_j$ and $\boldsymbol{B}_j$ as defined on lines 12 and 13. For notational convenience, we focus on the last loop iteration when $j = s$. Conformally partitioning $\boldsymbol{B}$ as

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{B}_s & \boldsymbol{b} \\ \boldsymbol{b}^* & \beta \end{bmatrix}.$$

The inverse then has representation

$$\boldsymbol{B}^{-1} = \begin{bmatrix} \boldsymbol{B}_s^{-1} & \boldsymbol{0} \\ \boldsymbol{0}^* & 0 \end{bmatrix} + \frac{1}{\beta - \boldsymbol{b}^* \boldsymbol{B}_s^{-1} \boldsymbol{b}} \begin{bmatrix} -\boldsymbol{B}_s^{-1}\boldsymbol{b} \\ 1 \end{bmatrix} \begin{bmatrix} -\boldsymbol{B}_s^{-1}\boldsymbol{b} \\ 1 \end{bmatrix}^*.$$

From this, we obtain an update formula of the desired form (A.3):

$$\boldsymbol{S}_s = \boldsymbol{R}_s \boldsymbol{B}_s^{-1} \boldsymbol{R}_s^* = \boldsymbol{R}\boldsymbol{B}^{-1}\boldsymbol{R}^* - \frac{1}{\beta - \boldsymbol{b}^* \boldsymbol{B}_s^{-1} \boldsymbol{b}} \left(\boldsymbol{R} \begin{bmatrix} -\boldsymbol{B}_s^{-1}\boldsymbol{b} \\ 1 \end{bmatrix} \right) \left(\boldsymbol{R} \begin{bmatrix} -\boldsymbol{B}_s^{-1}\boldsymbol{b} \\ 1 \end{bmatrix}^* \right) =: \boldsymbol{R}\boldsymbol{B}^{-1}\boldsymbol{R}^* - \boldsymbol{x}_s \boldsymbol{x}_s^*. \tag{A.4}$$

Similar formulas hold for $j < s$.

Having realized each $\boldsymbol{S}_j$ as a rank-one update to $\boldsymbol{R}\boldsymbol{B}^{-1}\boldsymbol{R}^*$, we describe how to form $\boldsymbol{S}_j$ in $\mathcal{O}(s^2)$ operations. We again restrict attention to $j = s$. First, obtain a Cholesky factorization of $\boldsymbol{B}_s$ from that of $\boldsymbol{B}$ using $\mathcal{O}(s^2)$ operations. This can be achieved by the approach from [3, §3] to compute the Cholesky factorization of a matrix after row and column deletion together with the Cholesky downdating algorithm of, e.g., [1]. Next, use this Cholesky factorization to form $\boldsymbol{S}_s$ by evaluating (A.4). By using the generalization of (A.4) for arbitrary j, this approach allows us to form each $\boldsymbol{S}_j$ in $\mathcal{O}(s^2)$ operations. The jackknife variance estimate is then computable in $\mathcal{O}(s^3)$ operations, as claimed.

Using an approach similar to the one outlined at the end of section A.1, this approach can be modified to obtain jackknife variance estimates for derived quantities like spectral projectors in $\mathcal{O}(s^3)$ operations.

REFERENCES

[1] A. W. Bojanczyk, R. P. Brent, P. van Dooren, and F. R. de Hoog, *A note on downdating the Cholesky factorization*, SIAM Journal on Scientific and Statistical Computing, 8 (1987), pp. 210–221. 18

[2] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, *Modeling wine preferences by data mining from physicochemical properties*, Decision Support Systems, 47 (2009), pp. 547–553. 12

[3] T. A. Davis and W. W. Hager, *Row modifications of a sparse Cholesky factorization*, SIAM Journal on Matrix Analysis and Applications, 26 (2005), pp. 621–639. 18

[4] L. Ding and Y. Chen, *Leave-one-out approach for matrix completion: Primal and dual analysis*, arXiv:1803.07554 [stat.ML], (2018). 17

[5] P. Drineas and M. W. Mahoney, *On the Nyström method for approximating a Gram matrix for improved kernel-based learning*, Journal of Machine Learning Research, 6 (2005), pp. 2153–2175. 10

[6] P. Drineas and M. W. Mahoney, *RandNLA: Randomized numerical linear algebra*, Communications of the ACM, 59 (2016), pp. 80–90. 1

[7] B. Efron, *The Jackknife, the Bootstrap and Other Resampling Plans*, SIAM, 1982. 2, 7

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

[9] Z. Frangella, J. A. Tropp, and M. Udell, *Randomized Nyström Preconditioning*, arXiv:2110.02820 [math.NA], (2021). 10

[10] 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. 10

[11] M. Gu, *Studies in numerical linear algebra*, PhD thesis, Yale University, 1993. 18

[12] N. Halko, P.-G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Review, 53 (2011), pp. 217–288. 1, 2, 7, 8

[13] P. Jain, P. Netrapalli, and S. Sanghavi, *Low-rank matrix completion using alternating minimization*, in Proceedings of the 45th Annual ACM Symposium on Symposium on Theory of Computing, ACM Press, 2013, pp. 665–674. 17

[14] 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 transactions on mathematical software, 43 (2017). 11

[15] 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, vol. 119 of Proceedings of Machine Learning Research, PMLR, 13–18 Jul 2020, pp. 6382–6392. 16

[16] M. E. Lopes, N. B. Erichson, and M. W. Mahoney, *Bootstrapping the operator norm in high dimensions: Error estimation for covariance matrices and sketching*, arXiv:1909.06120 [math.ST], (2019). 16

[17] M. E. Lopes, S. Wang, and M. W. Mahoney, *A bootstrap method for error estimation in randomized matrix multiplication*, Journal of Machine Learning Research, (2019), p. 40. 16

[18] P.-G. Martinsson and J. A. Tropp, *Randomized numerical linear algebra: Foundations and algorithms*, Acta Numerica, 29 (2020), pp. 403–572. 1, 16

[19] D. Paulin, L. Mackey, and J. A. Tropp, *Efron–Stein inequalities for random matrices*, Annals of Probability, 44 (2016), pp. 3431–3473. 16

[20] Y. Plan, R. Vershynin, and E. Yudovina, *High-dimensional estimation with geometric constraints*, Information and Inference: A Journal of the IMA, 6 (2017), pp. 1–40. 17

[21] M. H. Quenouille, *Approximate tests of correlation in time-series*, Journal of the Royal Statistical Society. Series B (Methodological), 11 (1949), pp. 68–84. 7

[22] N. Srebro, *Learning with matrix factorizations*, PhD thesis, Massachusetts Institute of Technology, 2004. 17

[23] J. M. Steele, *An Efron-Stein inequality for nonsymmetric statistics*, Annals of Statistics, 14 (1986), pp. 753–758. 4

[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. 10, 12

[25] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher, *Streaming low-rank matrix approximation with an application to scientific simulation*, SIAM Journal on Scientific Computing, 41 (2019), pp. A2430–A2463. 17

[26] J. Tukey, *Bias and confidence in not quite large samples*, Ann. Math. Statist., 29 (1958), p. 614. 2, 4

[27] S. Voronin and P.-G. Martinsson, *Efficient algorithms for CUR and interpolative matrix decompositions*, Advances in Computational Mathematics, 43 (2017), pp. 495–516. 17

[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, 2000,

pp. 661–667. 10

[29] D. P. WOODRUFF, *Sketching as a Tool for Numerical Linear Algebra*, Foundations and Trends in Theoretical Computer Science, 10 (2014), pp. 1–157. 1