RANDOMIZED ALGORITHMS FOR LOW-RANK MATRIX APPROXIMATION: DESIGN, ANALYSIS, AND APPLICATIONS*

JOEL A. TROPP† AND ROBERT J. WEBBER†

Abstract. This survey explores modern approaches for computing low-rank approximations of high-dimensional matrices by means of the randomized SVD, randomized subspace iteration, and randomized block Krylov iteration. The paper compares the procedures via theoretical analyses and numerical studies to highlight how the best choice of algorithm depends on spectral properties of the matrix and the computational resources available.

Despite superior performance for many problems, randomized block Krylov iteration has not been widely adopted in computational science. This paper strengthens the case for the method in three ways. First, it presents new pseudocode that can significantly reduce computational costs. Second, it provides a new analysis that yields simple, precise, and informative error bounds. Last, it showcases applications to challenging scientific problems, including principal component analysis for genetic data and spectral clustering for molecular dynamics data.

Key words. Low-rank matrix approximation, randomized numerical linear algebra, Krylov subspace methods

AMS subject classifications. 68W20, 65F10, 65F55

1. Motivation.

A core problem in numerical linear algebra is to produce a low-rank, factorized approximation of a high-dimensional input matrix $\boldsymbol{A} \in \mathbb{R}^{L \times N}$:

$$\underset{L \times N}{\boldsymbol{A}} \approx \underset{L \times k}{\boldsymbol{B}} \underset{k \times N}{\boldsymbol{C}}$$

We think of the inner dimension k as much smaller than the outer dimensions L and N. In this case, the factorized approximation $\boldsymbol{BC}$ has fewer degrees of freedom than the input matrix $\boldsymbol{A}$, so the factorization is easier to store and manipulate. Furthermore, to attain a small error, the approximation must expose structure in the input matrix. Particular examples of "structure" include the range of the input matrix (rank-revealing QR factorization), principal components (truncated singular value decomposition), or salient columns (CUR or interpolative decomposition).

Low-rank approximation serves as a fundamental tool for computational science. Application areas include fluid dynamics [11, 92], uncertainty quantification [22, 29], genetics [41], climate science [106], geophysics [111], astronomical imaging [66], and beyond. Indeed, low-rank approximation is helpful whenever we need to extract dominant patterns in data, remove unwanted noise, or perform compression.

For high-dimensional matrices, the computational difficulty of low-rank approximation depends on the singular value spectrum of the input matrix; see Figure 1 for a schematic. While we can apply simple algorithms to approximate a matrix with a rapidly decaying spectrum, we need more powerful algorithms to approximate a matrix with a slowly decaying spectrum. This challenge arises in many modern applications. For example, consider this quotation from the genetics literature [23]:

> "[I]n large datasets, eigenvalues may be highly significant (reflecting real population structure in the data) but only slightly larger than

*Funding: JAT and RJW acknowledge partial support from the Office of Naval Research through BRC Award N00014-18-1-2363, from the National Science Foundation through FRG Award 1952777, and from Caltech through the Carver Mead New Adventures Fund.

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

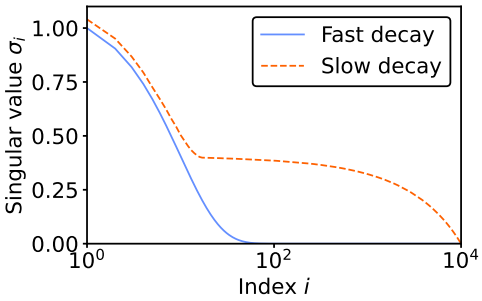

FIG. 1. ***(Singular value decay profiles).*** *Fast versus slow singular value decay; for details of the matrices, see subsection 2.3.*

background noise eigenvalues..."

In this setting, we require scalable techniques that can filter out the noise components, while accurately approximating the signal components. So what algorithms should we use?

This survey explores a family of powerful methods for low-rank approximation: the randomized singular value decomposition (`RSVD`), randomized subspace iteration (`RSI`), and randomized block Krylov iteration (`RBKI`). We will introduce these techniques in subsections 2.1 and 2.2. They are related because each one collects information about the input matrix using matrix–vector products with random vectors. We will develop a systematic comparison of the three approaches by examining their mathematical structure, providing new implementations, refining theoretical analyses, and presenting numerical studies.

As compared with classic methods, the randomized algorithms are more scalable, reliable, and robust. Our investigation shows that the simplest method, `RSVD`, returns accurate approximations for matrices with rapid singular value decay, but we must use the more sophisticated `RBKI` method for challenging problems. In case storage is the limiting factor, `RSI` may offer a reasonable compromise. These conclusions will be familiar to experts.

Nevertheless, the `RBKI` method still has not reached a wide audience in computational science. (For example, see the recent genetics review [103].) Therefore, the second goal of this survey is to establish firm foundations for the `RBKI` method to speed its adoption. We develop a new formulation of the algorithm that is significantly faster than previous implementations. We prove new theoretical guarantees that demonstrate exactly how `RBKI` improves over `RSVD` and `RSI`. Finally, we show applications to benchmark problems in computational science that confirm the benefits of `RBKI` in these settings.

Remark 1.1 (Finding structure with randomness). The paper [54] of Halko et al. made a comprehensive case for `RSVD` and `RSI`, which led to broader usage of these methods. The goal of this survey is to perform the same mission for `RBKI`.

2. Randomized matrix approximation: Overview.

This section offers an introduction to the randomized algorithms that we study in this paper. It provides a first look at the numerical behavior of these methods, and it presents simple theoretical bounds that allow us to compare their performance.

2.1. Framework. In 2009, Halko et al. [54] developed a framework for designing randomized algorithms for low-rank matrix approximation. In this section, we outline the key concepts from their approach.

Consider the problem of approximating a high-dimensional matrix $\boldsymbol{A} \in \mathbb{R}^{L \times N}$. The matrix can be an array of data stored on a computer, or it can be an abstract linear operator defined by its action on test vectors. Regardless, we only access $\boldsymbol{A}$ by performing multiplications $\boldsymbol{x} \mapsto \boldsymbol{A}\boldsymbol{x}$ and $\boldsymbol{y} \mapsto \boldsymbol{A}^*\boldsymbol{y}$ with input vectors $\boldsymbol{x} \in \mathbb{R}^N$ and $\boldsymbol{y} \in \mathbb{R}^L$. As usual, $\boldsymbol{A}^*$ is the (conjugate) transpose.

The first idea underlying low-rank approximation is that we can find important directions in the range of $\boldsymbol{A}$ by applying the matrix to a *random* vector. Suppose that $\boldsymbol{A}$ has the singular value decomposition (SVD)

$$\boldsymbol{A} = \sum\nolimits_{i=1}^{N} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^* \quad \text{with } \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_N \geq 0.$$

The "leading" left singular vectors $\boldsymbol{u}_i$, associated with the largest singular values σ_i, are significant directions in the range. Draw a random vector $\boldsymbol{\omega} \in \mathbb{R}^N$ from the standard normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I}_N)$. By rotational invariance, we can express $\boldsymbol{\omega}$ in the basis of right singular vectors:

$$\boldsymbol{\omega} = \sum\nolimits_{i=1}^{N} Z_i \, \boldsymbol{v}_i \quad \text{where } Z_i \overset{\text{iid}}{\sim} \mathcal{N}(0,1).$$

With probability one, all of the random coefficients Z_i are nonzero, and they are on the same scale because each coefficient Z_i has mean zero and variance one. The matrix–vector product takes the form

$$\boldsymbol{A}\boldsymbol{\omega} = \sum\nolimits_{i=1}^{N} Z_i \sigma_i \, \boldsymbol{u}_i.$$

We see that the image $\boldsymbol{A}\boldsymbol{\omega}$ is strongly correlated with the leading left singular vectors, while it is weakly correlated with the trailing left singular vectors.

Now, observe that *repeated* multiplication with the input matrix $\boldsymbol{A}$ and its transpose $\boldsymbol{A}^*$ amplify the coefficients associated with large singular values, while attenuating coefficients with small singular values. More precisely,

$$(\boldsymbol{A}\boldsymbol{A}^*)^{q-1}\boldsymbol{A}\boldsymbol{\omega} = \sum\nolimits_{i=1}^{N} Z_i \sigma_i^{2q-1} \, \boldsymbol{u}_i.$$

As we increase the depth $q = 1, 2, 3, \ldots$, the small singular values are suppressed exponentially fast. Therefore, the output vector is more and more likely to be aligned with the leading left singular vectors. In fact, we can achieve this goal even when the depth is a moderate constant, say, $q \leq 5$. There is no need to take a limit.

Next, we double down on this insight by multiplying the input matrix with a *block* of $k \gg 1$ vectors to find many important directions in the range. To express this operation, we introduce a standard normal matrix $\boldsymbol{\Omega} = \begin{bmatrix} \boldsymbol{\omega}_1 & \ldots & \boldsymbol{\omega}_k \end{bmatrix} \in \mathbb{R}^{N \times k}$ with independent columns. By sequential multiplication, we obtain the matrices

$$\boldsymbol{A}\boldsymbol{\Omega}, \quad (\boldsymbol{A}\boldsymbol{A}^*)\boldsymbol{A}\boldsymbol{\Omega}, \quad (\boldsymbol{A}\boldsymbol{A}^*)^2\boldsymbol{A}\boldsymbol{\Omega}, \quad \ldots, \quad (\boldsymbol{A}\boldsymbol{A}^*)^{q-1}\boldsymbol{A}\boldsymbol{\Omega}.$$

As the powers increase, the ranges of these matrices align better with the k leading left singular vectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k$ of the input matrix. We will quantify the extent of the overlap in sections 8 and 9.

Finally, suppose we have constructed a matrix $\boldsymbol{M} \in \mathbb{R}^{L \times k}$ whose range aligns with the leading left singular vectors of the input matrix $\boldsymbol{A}$. We can build the desired

low-rank approximation of $\boldsymbol{A}$ by *compressing* $\boldsymbol{A}$ into the range of $\boldsymbol{M}$. To that end, we orthogonalize the columns of $\boldsymbol{M}$ to obtain a matrix $\boldsymbol{X} = \texttt{orth}(\boldsymbol{M})$, and we construct the orthogonal projection $\boldsymbol{\Pi}_{\boldsymbol{M}} = \boldsymbol{X}\boldsymbol{X}^*$ onto the range of $\boldsymbol{M}$. Then we form the approximation

$$\hat{\boldsymbol{A}} = \boldsymbol{\Pi}_{\boldsymbol{M}}\boldsymbol{A} = \boldsymbol{X}\boldsymbol{X}^*\boldsymbol{A} = \boldsymbol{X}\boldsymbol{Y}^* \qquad \text{where } \boldsymbol{Y} = \boldsymbol{A}^*\boldsymbol{X}. \tag{2.1}$$

Since $\boldsymbol{X}$ has at most k columns, so does the matrix $\boldsymbol{Y}$. We have produced a factorized rank-k approximation $\hat{\boldsymbol{A}} = \boldsymbol{X}\boldsymbol{Y}^*$ of the input matrix $\boldsymbol{A}$.

At this point, the approximation $\hat{\boldsymbol{A}}$ can be manipulated into alternative forms by standard transformations. For example, we can easily produce an SVD, a QR factorization, or a CUR decomposition. See [54, Sec. 5] for details.

2.2. Matrix approximations and matrix computations. Combining these basic ideas, we obtain several fundamental approaches to low-rank matrix approximation.

In the first approach, called "randomized singular value decomposition" or `RSVD`, we generate the low-rank approximation

$$\hat{\boldsymbol{A}} = \boldsymbol{\Pi}_{\boldsymbol{A}\boldsymbol{\Omega}}\boldsymbol{A}. \tag{RSVD}$$

This procedure requires a first multiplication with $\boldsymbol{A}$ and a second multiplication with $\boldsymbol{A}^*$, while storage is limited to two blocks of k vectors. See Algorithm 5.1 for pseudocode.

In the second approach, called "randomized subspace iteration" or `RSI`, we generate

$$\hat{\boldsymbol{A}} = \boldsymbol{\Pi}_{(\boldsymbol{A}\boldsymbol{A}^*)^{q-1}\boldsymbol{A}\boldsymbol{\Omega}}\boldsymbol{A}. \tag{RSI}$$

This procedure requires $2q$ multiplications, alternating between $\boldsymbol{A}$ and $\boldsymbol{A}^*$, while storage is limited to two blocks of k vectors. We think about the depth parameter q as a *fixed* number, rather than treating the algorithm as a limiting process. See Algorithm 5.2 for pseudocode.

The third approach, called "randomized block Krylov iteration" or `RBKI`, forms the approximation

$$\hat{\boldsymbol{A}} = \boldsymbol{\Pi}_{[\,\boldsymbol{A}\boldsymbol{\Omega}\;\cdots\;(\boldsymbol{A}\boldsymbol{A}^*)^{q-1}\boldsymbol{A}\boldsymbol{\Omega}\,]}\boldsymbol{A}. \tag{RBKI}$$

In this case, we project onto the range of the block matrix $\big[\boldsymbol{A}\boldsymbol{\Omega} \quad \cdots \quad (\boldsymbol{A}\boldsymbol{A}^*)^{q-1}\boldsymbol{A}\boldsymbol{\Omega}\big]$. Subsection 5.3 shows how to carry out this construction using $2q$ multiplications. The storage cost is higher than the other algorithms, typically $2q$ blocks of k vectors. As before, the depth parameter q is viewed as a fixed number. See Algorithm 5.4 for pseudocode, which reduces the number of matrix–vector products by 33% compared to previous `RBKI` implementations.

Our primary objective in this work is to address the following question:

> What is the most appropriate algorithm for low-rank approximation of a given matrix: `RSVD`, `RSI`, or `RBKI`?

The answer depends on the singular value spectrum of the matrix, the required accuracy, and limitations on computational resources (such as storage).

We focus on the high-dimensional setting in which L or N is large (say, $\geq 10^5$). In this case, the predominant operating cost for each approximation method is the

repeated application of $\boldsymbol{A}$ and $\boldsymbol{A}^*$ to a block of k vectors. Therefore, we measure the operating cost in terms of the number m of matrix–matrix multiplications, together with the block size k. The number of matrix–vector products is km for all the algorithms we consider.

Our goal is to reduce the parameters m and k as much as possible, while ensuring a robust and accurate low-rank approximation. We accept certain tradeoffs between m and k, but we focus mainly on decreasing the number m of multiplications. Reducing m gives a linear improvement in the runtime because the repeated matrix multiplications are inherently serial.

Remark 2.1 (Matrix multiplication). On modern computers, matrix multiplications with a large block size k are optimized to take advantage of architectural features, including multithreading [93], caching [49, 69], parallelization [22], and single-instruction multiple data processing [104]. By designing multiplication-rich algorithms, we harness the full power of modern computing platforms. This is one of the main advantages of block Krylov methods over Krylov methods with a single starting vector ($k = 1$). See subsection 6.1 for runtime comparisons.

2.3. Illustrative comparison. In this section, we present an illustrative comparison of `RSVD`, `RSI`, and `RBKI`. Section 7 presents a more comprehensive series of experiments, including real-world applications.

We consider approximating two matrices with dimensions $L = N = 10^4$:

$$\boldsymbol{A} = \operatorname{diag}(1, \mathrm{e}^{-.1}, \mathrm{e}^{-.2}, \dots, \mathrm{e}^{-999.9}), \qquad \boldsymbol{B} = \boldsymbol{A} + \boldsymbol{Z}, \quad \text{where } Z_{ij} \overset{\text{iid}}{\sim} \mathcal{N}(0, .002^2).$$

The first matrix $\boldsymbol{A}$ has exponentially decaying singular values; see the "fast decay" profile in Figure 1. The second matrix $\boldsymbol{B}$ is a noisy version of $\boldsymbol{A}$ that has been perturbed by adding an independent Gaussian random variable to each entry. For a particular realization of the noise, the upper left submatrices are

$$\boldsymbol{A} = \begin{bmatrix} 1.000 & 0.000 & 0.000 & 0.000 & \\ 0.000 & 0.905 & 0.000 & 0.000 & \\ 0.000 & 0.000 & 0.819 & 0.000 & \\ 0.000 & 0.000 & 0.000 & 0.741 & \\ & & & & \ddots \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} 1.001 & -0.002 & 0.002 & 0.002 & \\ 0.000 & 0.907 & -0.003 & -0.001 & \\ -0.003 & 0.002 & 0.819 & -0.001 & \\ -0.002 & 0.000 & -0.003 & 0.739 & \\ & & & & \ddots \end{bmatrix}.$$

The matrix entries are similar, up to variations on the scale ± 0.002. However, the Gaussian noise transforms the structure of the singular values. The matrix $\boldsymbol{B}$ has a long tail of singular values with magnitudes 0.0 to 0.4; see the "slow decay" profile in Figure 1.

First, we exhibit the output of `RSVD` (Algorithm 5.1) with a block size $k = 100$ when applied to the original matrix $\boldsymbol{A}$:

$$(\texttt{RSVD}) \qquad \hat{\boldsymbol{A}}_{k=100} = \begin{bmatrix} 1.000 & 0.000 & 0.000 & 0.000 & \\ 0.000 & 0.905 & 0.000 & 0.000 & \\ 0.000 & 0.000 & 0.819 & 0.000 & \\ 0.000 & 0.000 & 0.000 & 0.741 & \\ & & & & \ddots \end{bmatrix}.$$

The upper left submatrix coincides with $\boldsymbol{A}$ up to three decimal places. `RSVD` performs ideally for this example, and there is no reason to increase the block size or use a more elaborate approximation.

In contrast, RSVD with a block size $k = 100$ produces an appalling approximation of the noisy matrix $\boldsymbol{B}$, with major discrepancies in the first four rows and columns:

$$\text{(RSVD)} \qquad \hat{\boldsymbol{B}}_{k=100} = \begin{bmatrix} 0.270 & -0.018 & 0.024 & -0.006 & \\ -0.019 & 0.159 & 0.021 & -0.007 & \\ 0.029 & 0.024 & 0.106 & -0.015 & \\ -0.008 & -0.008 & -0.018 & 0.086 & \\ & & & & \ddots \end{bmatrix}.$$

We can improve the approximation quality by using a larger block size $k = 1{,}000$, but the errors remains large:

$$\text{(RSVD)} \qquad \hat{\boldsymbol{B}}_{k=1000} = \begin{bmatrix} 0.779 & -0.001 & 0.004 & 0.006 & \\ 0.002 & 0.655 & 0.000 & -0.004 & \\ 0.003 & 0.005 & 0.569 & -0.006 & \\ 0.004 & -0.005 & -0.009 & 0.487 & \\ & & & & \ddots \end{bmatrix}.$$

The off-diagonal entries are too large in magnitude, and the diagonal entries are too small.

As an alternative to RSVD, we can approximate the noisy matrix $\boldsymbol{B}$ using RSI (Algorithm 5.3) or RBKI (Algorithm 5.5) with a block size $k = 100$. After $m = 5$ multiplications, RSI improves on RSVD, but it still produces an underestimate of the diagonal:

$$\text{(RSI)} \qquad \hat{\boldsymbol{B}}_{\text{RSI}} = \begin{bmatrix} 0.997 & -0.002 & 0.001 & 0.001 & \\ 0.001 & 0.899 & -0.002 & 0.000 & \\ -0.003 & 0.002 & 0.806 & -0.001 & \\ -0.003 & 0.000 & -0.002 & 0.723 & \\ & & & & \ddots \end{bmatrix}.$$

In contrast, after $m = 5$ multiplications, RBKI yields the superior approximation

$$\text{(RBKI)} \qquad \hat{\boldsymbol{B}}_{\text{RBKI}} = \begin{bmatrix} 0.999 & -0.002 & 0.001 & 0.002 & \\ 0.000 & 0.905 & -0.003 & -0.001 & \\ -0.003 & 0.002 & 0.816 & -0.001 & \\ -0.002 & 0.000 & -0.003 & 0.736 & \\ & & & & \ddots \end{bmatrix}.$$

In fact, the approximation $\hat{\boldsymbol{B}}_{\text{RBKI}}$ agrees with the best rank-100 approximation of $\boldsymbol{B}$ up to three decimal places. The remaining error is a consequence of the low-rank approximation, rather than a deficiency in the RBKI algorithm.

The takeaway from these simple experiments is that it can be difficult to approximate a matrix such as $\boldsymbol{B}$ that has been corrupted by noise. For approximation problems with noisy matrices, RBKI typically offers the best combination of speed and accuracy, and it outperforms the other randomized low-rank approximation algorithms.

2.4. Theoretical analysis. In this section, we introduce our main error bounds for randomized low-rank matrix approximation, which help to explain the outcomes of the experiments in subsection 2.3. Detailed derivations and additional bounds can be found in sections 8 and 9.

Return to the setting where $\boldsymbol{A} \in \mathbb{R}^{L \times N}$ is an arbitrary matrix. We measure the quality of a computed approximation by comparison with an optimal rank-r approximation. For a parameter $r \geq 1$, the minimal rank-r approximation error in the spectral norm is given by the $(r+1)$st singular value:

$$\min\{\|\boldsymbol{B} - \boldsymbol{A}\| : \operatorname{rank}(\boldsymbol{B}) = r\} = \sigma_{r+1}(\boldsymbol{A}).$$

Here, $\|\cdot\|$ is the spectral norm, and $\sigma_j(\cdot)$ returns the jth largest singular value. Equality holds for any r-truncated SVD of the matrix $\boldsymbol{A}$.

A randomized low-rank approximation algorithm produces a random approximation $\hat{\boldsymbol{A}}$ of the input matrix $\boldsymbol{A}$. We evaluate the computed approximation $\hat{\boldsymbol{A}}$ using the mean-square relative error:

$$\mathbb{E}\left[\frac{\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|^2}{\sigma_{r+1}(\boldsymbol{A})^2}\right].$$

The expectation $\mathbb{E}$ averages over the randomness in the algorithm, which comes from the choice of the random initialization matrix $\boldsymbol{\Omega}$. When we compare an approximation with rank k to a best approximation with rank $r < k$, the relative error could be smaller than one.

Here is the core technical question:

> For a given target rank r, what block size k and number m of matrix multiplications suffice to make the mean-square relative error small?

Our theoretical analysis elucidates how the parameters k, m and the choice of approximation subspace affect the quality of the approximation. Let us emphasize that the user of the algorithm specifies k and m, while the comparison rank r only appears in the theoretical analysis.

As a first result, we present an error bound for `RSVD` with k Gaussian vectors. For each block size $k \geq r + 2$, we have the estimate

$$\frac{\mathbb{E}\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|^2}{\sigma_{r+1}(\boldsymbol{A})^2} \leq 1 + \frac{r}{k - r - 1} \sum_{i > r} \frac{\sigma_i(\boldsymbol{A})^2}{\sigma_{r+1}(\boldsymbol{A})^2}. \tag{RSVD}$$

As the block size k increases, the polynomial factor $r/(k - r - 1)$ decreases. Typical choices for the block size are $k = r + 2$ and $k = 2r + 1$. Once k and r are fixed, the bound depends only on the tail singular values $\sigma_{r+1}(\boldsymbol{A}), \sigma_{r+2}(\boldsymbol{A}), \ldots$. When these tail singular values decay quickly, the error is proportional to the optimal rank-r error. For example, when the singular values of $\boldsymbol{A}$ decay exponentially fast, the error satisfies $\mathbb{E}\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|^2 \leq (1 + \varepsilon) \cdot \sigma_{r+1}(\boldsymbol{A})^2$ for $k \geq r + \text{const} \cdot \log(1 + r/\varepsilon)$.

The next result shows that randomized subspace iteration, `RSI`, is a better alternative for matrices with slowly decaying singular values. Consider the approximation $\hat{\boldsymbol{A}}$ produced by the `RSI` algorithm with k Gaussian vectors and m matrix multiplications. For each block size $k \geq r + 2$,

$$\log\left(\frac{\mathbb{E}\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|^2}{\sigma_{r+1}(\boldsymbol{A})^2}\right) \leq \frac{1}{m - 1} \log\left(1 + \frac{r}{k - r - 1} \sum_{i > r} \frac{\sigma_i(\boldsymbol{A})^2}{\sigma_{r+1}(\boldsymbol{A})^2}\right). \tag{RSI}$$

The *logarithm* of the mean-square relative error decreases in proportion to the number m of multiplications. The logarithm ensures that the error cannot depend too strongly

on the singular value decay. For example, the error always satisfies $\mathbb{E}\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|^2 \leq \mathrm{e} \cdot \sigma_{\mathrm{r}+1}(\mathrm{A})^2$ for $k = 2r + 1$ and $m \geq \log\bigl(1 + \min\{L, N\}\bigr) + 1$.

Now, consider the approximation $\hat{\boldsymbol{A}}$ produced by the RBKI algorithm with k Gaussian initialization vectors and m matrix multiplications. For each block size $k \geq r+2$,

$$\text{(RBKI)} \qquad \log\left(\frac{\mathbb{E}\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|^2}{\sigma_{r+1}(\boldsymbol{A})^2}\right) \leq \frac{1}{4(m-2)^2}\left[\log\left(4 + \frac{4r}{k-r-1}\sum_{i>r}\frac{\sigma_i(\boldsymbol{A})^2}{\sigma_{r+1}(\boldsymbol{A})^2}\right)\right]^2.$$

This bound shows that RBKI reduces the logarithm of the relative error proportionally to the *square* m^2 of the number m of multiplications, which suggests that RBKI can provide a far more accurate approximation than RSI. For example, we might require $m = 100$ multiplications to obtain an accurate approximation using RSI but only $m = 10$ multiplications using RBKI.

As stated, these error bounds are new, but they build on prior work. The results for RSVD and RSI are patterned on arguments from Halko et al. [54, Thm. 10.6, Cor. 10.10]. Musco and Musco [74, Thm. 1] obtained a qualitative result for RBKI that identifies the $\mathcal{O}(m^{-2})$ scaling, but our detailed analysis of RBKI does not have a precedent in the literature. For mathematical derivations and a more comprehensive discussion, see sections 8 and 9.

2.5. Positive-semidefinite matrices. A special priority of this survey is to develop methods for approximating *positive-semidefinite* (psd) matrices. Of course, we can approximate a psd matrix using the general-purpose algorithms RSVD, RSI, and RBKI, but this strategy is wasteful. A more accurate alternative that also preserves the psd property is to employ algorithms based on Nyström approximation (discussed in subsection 5.4).

We discuss three methods: NysSVD, NysSI, and NysBKI, which are the Nyström-based extensions of RSVD, RSI, or RBKI. We address the following question:

> What is the most appropriate algorithm for low-rank approximation of a given psd matrix: NysSVD, NysSI, or NysBKI?

Our results show that NysSVD is the most efficient algorithm for rapidly decaying eigenvalues, while NysBKI is the most efficient algorithm for slowly decaying eigenvalues. Additionally, our theory and experiments suggest that NysBKI improves on RBKI by a factor of $\sqrt{2}$ matrix–matrix multiplications, demonstrating the potential for speedups in the psd setting.

The NysRBKI algorithm is new, as is most of the theory for the Nyström methods. For a real-world application, see the spectral clustering problem in subsection 7.3.

2.6. Plan for the paper. The rest of the paper is organized as follows. Section 3 presents historical background, section 4 defines goals of low-rank matrix approximation, section 5 presents pseudocode, section 6 discusses parameter choices, section 7 showcases numerical experiments, sections 8 and 9 derive error bounds, and section 10 offers conclusions.

2.7. Notation. For simplicity, we work with real-valued matrices only. There is an analogous theory for complex-valued matrices, although the constants may differ. We use the term *Gaussian* random matrix (or vector) as a shorthand for a matrix with independent Gaussian entries with mean zero and variance one.

We adopt standard notation for various matrix operations. The transpose of a matrix $\boldsymbol{A} \in \mathbb{R}^{L\times N}$ is represented by $\boldsymbol{A}^*$. The Moore–Penrose pseudoinverse of

$\boldsymbol{A}$ is denoted by $\boldsymbol{A}^\dagger$, and the orthogonal projection onto the range space of $\boldsymbol{A}$ is written as $\boldsymbol{\Pi}_{\boldsymbol{A}} = \boldsymbol{A}(\boldsymbol{A}^*\boldsymbol{A})^\dagger\boldsymbol{A}^*$. The eigenvalues of a positive-semidefinite matrix $\boldsymbol{A}$ are ordered from largest to smallest as $\lambda_1(\boldsymbol{A}) \geq \lambda_2(\boldsymbol{A}) \geq \cdots$, as are the singular values $\sigma_1(\boldsymbol{A}) \geq \sigma_2(\boldsymbol{A}) \geq \ldots$ of a general matrix $\boldsymbol{A}$. The symbol $\preccurlyeq$ denotes the positive-semidefinite (psd) order on symmetric matrices: $\boldsymbol{A} \preccurlyeq \boldsymbol{B}$ if and only if $\boldsymbol{B} - \boldsymbol{A}$ is psd.

We measure the size of a matrix $\boldsymbol{M} \in \mathbb{R}^{L\times N}$ using the Schatten p-norm [12, Sec. 4.2], defined by

$$\|\boldsymbol{M}\|_p = \left(\sum\nolimits_{i=1}^{N} \sigma_i(\boldsymbol{M})^p\right)^{1/p}$$

for $1 \leq p < \infty$ and $\|\boldsymbol{M}\|_\infty = \sigma_1(\boldsymbol{M})$. The most commonly used Schatten p-norms are the trace norm $\|\cdot\|_1 = \|\cdot\|_*$, the Frobenius norm $\|\cdot\|_2 = \|\cdot\|_{\mathrm{F}}$, and the spectral norm $\|\cdot\|_\infty = \|\cdot\|$, each with its distinctive notation. Schatten p-norms are unitarily invariant, invariant under (conjugate) transposition, and satisfy the matrix inequality $\|\boldsymbol{AMB}\|_p \leq \|\boldsymbol{A}\|\|\boldsymbol{M}\|_p\|\boldsymbol{B}\|$.

In addition to the standard conventions, we introduce the following special definitions. A matrix is said to be rank-r when its rank does not exceed r. We use $\lfloor \boldsymbol{A} \rfloor_r$ to denote any optimal rank-r approximation of the matrix $\boldsymbol{A} \in \mathbb{R}^{L\times N}$, derived from an r-truncated singular value decomposition:

$$\boldsymbol{A} = \sum\nolimits_{i=1}^{N} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^* \quad \text{yields} \quad \lfloor \boldsymbol{A} \rfloor_r = \sum\nolimits_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^*.$$

This low-rank approximation may not be unique, so we employ the notation only in contexts where it leads to an unambiguous statement. We represent the Nyström approximation of a psd matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ with respect to a test matrix $\boldsymbol{X} \in \mathbb{R}^{N\times k}$ as $\boldsymbol{A}\langle \boldsymbol{X} \rangle = \boldsymbol{AX}(\boldsymbol{X}^*\boldsymbol{AX})^\dagger\boldsymbol{X}^*\boldsymbol{A}$.

3. History.

In this section, we provide a concise history of randomized low-rank matrix approximation.

3.1. Numerical methods for spectral computation.

Computational tools for low-rank approximation have their roots in numerical methods for spectral computation. We focus on methods that extract a few eigenvalues and eigenvectors, rather than returning a full spectral decomposition.

In the early 20th century, the power method and variants, such as inverse iteration, emerged as popular techniques for computing a few eigenvalues and eigenvectors of a square matrix (see [96, Sec. 2.1.3]). Around 1950, Lanczos [62] introduced an improvement of the power method for Hermitian matrices that fully exploits Krylov information (matrix powers applied to vectors).

Both the power method and the Lanczos algorithm employ just a single starting vector (i.e., block size $k = 1$), making them inadequate for resolving eigenvalues with multiplicity higher than one. To address this shortcoming, between the 1950s and 1970s, computational mathematicians developed versions of the power method and the Lanczos algorithm that use multiple starting vectors, called "subspace iteration" and "block Krylov iteration" (see [96, Secs. 5.3.4, 6.1.4]).

These classic iterative methods have been extensively discussed in the literature; see Golub and van der Vorst [48] for an historical overview. For modern accounts of the power method and the Lanczos algorithm, refer to the textbooks of Partlett [83] and Saad [90]. For analyses of subspace iteration and block Krylov iteration, see Stewart [94] and Li and Zhang [68].

All the traditional iterative methods for calculating the dominant eigenvalues and eigenvectors of a Hermitian matrix can be adapted to calculate the dominant singular values and singular vectors of a rectangular matrix $\boldsymbol{A} \in \mathbb{R}^{L \times N}$ by applying them to the Jordan–Wielandt matrix $\left[\begin{smallmatrix} \mathbf{0} & \boldsymbol{A} \\ \boldsymbol{A}^* & \mathbf{0} \end{smallmatrix}\right]$; see [45, 47]. When initialized with $\boldsymbol{A\Omega}$ where $\boldsymbol{\Omega}$ is a random matrix, this approach matches the approximate singular value decomposition $\hat{\boldsymbol{A}} = \boldsymbol{U\Sigma V}^*$ from `RSI` or `RBKI`. However, the iterative approach to singular value decomposition was historically used to calculate a few (say, 1–4) leading singular values and singular vectors [47]. The potential for converting these schemes into algorithms for low-rank approximation apparently went unrecognized until the 21st century.

Early iterative algorithms for eigenvalue or singular vector computations used relatively few starting vectors (say, 3 or 4), which were often generated randomly, e.g., by random Gaussians. The random initialization was viewed as undesirable but necessary to ensure nonzero overlap with the leading singular vectors [79, 27]. To obtain accurate results, the algorithms employed a large number of matrix multiplications (10s or 100s). Numerical analysts emphasized the benefit of these repeated multiplications, since they viewed these computations as limiting processes, rather than finite algorithms [83, Sec. 13.2].

3.2. The benefits of random sampling. The idea to calculate a singular value decomposition using just a few matrix multiplications was not seriously considered in the literature on classical iterative algorithm. Computational scientists needed a new probabilistic perspective to bring this computational strategy to light.

Dixon (1983) [31] and Kuczyński and Woźniakowski (1992) [60] introduced a fresh analysis by applying probability theory to analyze the power method and the Lanczos algorithm. When the starting vector is Gaussian, these authors proved that the iterative algorithms can approximate the largest eigenvalue of a psd matrix after a *fixed* number of steps that depends logarithmically on the matrix dimension. They also demonstrated that the algorithms succeed even when the maximum eigenvalue is not separated from the remaining eigenvalues.

In the late 1990s and early 2000s, theoretical computer scientists began to analyze a different type of randomized low-rank approximation, through an approach called column sampling [40, 32, 30, 35]. The simplest column sampling approximation takes the form $\hat{\boldsymbol{A}} = \boldsymbol{\Pi}_{\boldsymbol{C}}\boldsymbol{A}$, where $\boldsymbol{C}$ is a random subset of the columns of $\boldsymbol{A}$ [35], chosen from an appropriate probability distribution. The probabilities can be defined proportionally to the square column norms [40, 32], or using a more complicated distribution based on adaptive sampling [30] or subspace sampling [35]. Additionally as a post-processing procedure, many column sampling algorithms apply an r-truncated singular value decomposition to produce the rank-r approximation $\hat{\boldsymbol{A}} = \lfloor \boldsymbol{\Pi}_{\boldsymbol{C}}\boldsymbol{A} \rfloor_r$. Various analyses [34, 89, 30, 21] demonstrate that column sampling can lead to a more accurate low-rank approximation than deterministic algorithms based on rank-revealing QR [44, 51]. For a more detailed history of column-sampling approaches, see the survey [54, Sec. 2.1.2].

Around the same time, researchers began investigating randomized low-rank approximations similar to `RSVD` and `RSI`. Papadimitriou et al. (1998) [82, 81] introduced a low-rank approximation $\hat{\boldsymbol{A}} = \boldsymbol{\Pi}_{\lfloor \boldsymbol{A\Omega} \rfloor_r}\boldsymbol{A}$, where $\boldsymbol{\Omega} \in \mathbb{R}^{N \times k}$ is a random orthogonal matrix and $\lfloor \boldsymbol{A\Omega} \rfloor_r$ is a r-truncated singular value decomposition of $\boldsymbol{A\Omega}$. Sarlós (2006) [91] proposed a similar approximation $\hat{\boldsymbol{A}} = \lfloor \boldsymbol{\Pi}_{\boldsymbol{A\Omega}}\boldsymbol{A} \rfloor_r$, where $\boldsymbol{\Omega} \in \mathbb{R}^{N \times k}$ is a subsampled trigonometric transform. Rokhlin et al. (2010) [87] acknowledged the difficulty of approximating a matrix with slowly decaying singular values, and they proposed

a more accurate low-rank approximation $\boldsymbol{\Pi}_{\lfloor (\boldsymbol{A}\boldsymbol{A}^*)^{q-1}\boldsymbol{A}\boldsymbol{\Omega}\rfloor_r}\boldsymbol{A}$, where $\boldsymbol{\Omega} \in \mathbb{R}^{N\times k}$ is a Gaussian matrix and $q \geq 1$ is a depth parameter; see also the 1997 paper of Roweis [88]. These early works argue that a small number of multiplications with a large block of random vectors (typically, $10 \leq k \leq 1{,}000$) is sufficient to approximate a matrix with rapidly decaying singular values, and they provide preliminary theory to support their argument.

In 2008 and 2009, Halko, Martinsson, and Tropp [54] developed a framework for designing randomized low-rank approximation algorithms, which leads to the standard versions of `RSVD` and `RSI` that treat the the rank-r truncation as an optional last step. Halko et al. also obtained the first theoretical analysis for these algorithms that is precise enough to predict their empirical behavior. They showed that `RSI` yields provable guarantees after a fixed number of matrix multiplications, even when the input matrix does not have gaps between the singular values. Following the publication of [54], researchers from various fields have applied randomized low-rank approximation to large-scale matrix computations in spectral clustering [20], genetics [1, 41], uncertainty quantification [22, 58], and image processing [25, 61, 111], among other areas.

3.3. Proposals for more efficient methods. In the next stage of inquiry, researchers proposed ways to improve the efficiency of randomized low-rank approximation, going beyond the basic `RSVD` and `RSI`. First, they introduced new approaches for the low-rank approximation of psd matrices based on *Nyström approximation* [54, 67, 100]. For a mathematical description of Nyström approximation, see subsection 5.4 or [70, Sec. 14]. Nyström approximation can be applied with any choice of test vectors, including random Gaussian vectors [100] or random coordinate vectors [109]. The approximation is always psd, and it always yields more accurate results than the simpler approximation based on orthogonal projection (see Lemma 5.2).

In another development, researchers improved the theoretical understanding of structured random matrices $\boldsymbol{\Omega} \in \mathbb{R}^{N\times k}$. Random matrices constructed using sparse sign vectors [18, 28] or subsampled trigonometric transforms [97, 39, 85] can be used for initializing randomized low-rank approximation, and they ensure a high quality of approximation. Due to fast matrix multiplications, these structured initialization matrices significantly improve runtimes for algorithms that require just a single matrix–matrix multiplication, such as `NysSVD` (Algorithm 5.6) or one-pass `RSVD` algorithms [110, 26, 101, 102]. However, when low-rank matrix approximation requires repeatedly multiplying the iterates with $\boldsymbol{A}$ and $\boldsymbol{A}^*$, as in `RSVD` or `RBKI`, the computational benefits of a structured random initialization are limited due to loss of structure after the first multiplication.

Last, one of the most significant advances in low-rank matrix approximation has been the introduction [87, 53] and analysis [74, 107, 33, 98, 99, 7] of randomized block Krylov methods. In 2015, Musco and Musco [74] proved that `RBKI` can be substantially more accurate than `RSI`, and their insight has spurred a variety of follow-up works [107, 33, 98, 99, 7, 73, 8]. In spite of this progress, the literature still lacks a crisp treatment of `RBKI` that parallels the earlier work on `RSVD` and `RBKI`. As a consequence, while `RBKI` is well-known among experts in randomized numerical linear algebra, it remains under-utilized in computational science (for instance, see the recent survey article [103] in genetics). Therefore, our objective in this work is to increase the use of `RBKI` by providing efficient pseudocode, detailed theory, and numerical experiments that highlight the benefits of this algorithm.

4. Goals and consequences. In this section, we identify the goals and consequences of randomized low-rank matrix approximation. We answer the questions, "What is the purpose of randomized low-rank approximation?" and "What would an ideal low-rank approximation algorithm accomplish?"

4.1. Goals. Ideally, a low-rank approximation algorithm should satisfy three main criteria: (1) speed, (2) accuracy, and (3) utility for downstream matrix computations.

The first design criterion is speed. Randomized low-rank approximation is intended to quickly extract structure from a high-dimensional matrix $\boldsymbol{A} \in \mathbb{R}^{L \times N}$. To that end, we develop multiplication-rich algorithms, which harness the efficiency of matrix multiplications on modern computing platforms. We perform most of the computations in a sequence of m matrix multiplications between $\boldsymbol{A}$ or $\boldsymbol{A}^*$ and a block of k vectors. Even with dense matrix–matrix multiplication, this approach costs only $\mathcal{O}(kmLN)$ arithmetic operations, so it can be much faster than a traditional (full) singular value decomposition or eigendecomposition, which expends $\mathcal{O}(LN \min\{L, N\})$ operations. See section 7 for examples of computational speedups spanning 1 to 3 orders of magnitude.

The second design criterion is accuracy. Our goal is to produce an approximation $\hat{\boldsymbol{A}}$ that is competitive with the best rank-r approximation in spectral norm:

$$\|\boldsymbol{A} - \hat{\boldsymbol{A}}\| \approx \|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_r\|. \tag{4.1}$$

We allow the rank of the approximation $\hat{\boldsymbol{A}}$ to be slightly larger than the comparison rank r. Our experiments (section 7) and theory (sections 8 and 9) show how randomized low-rank approximation achieves the goal (4.1).

The third design criterion is utility for downstream matrix calculations. We focus on algorithms that return a factorized approximation, which can be stored and manipulated efficiently. In the general setting, we seek a singular value decomposition $\hat{\boldsymbol{A}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$. In the psd setting, we instead seek an eigenvalue decomposition $\hat{\boldsymbol{A}} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^*$. The output of these algorithms identifies the singular vectors or eigenvectors of the approximation $\hat{\boldsymbol{A}}$, which serve as proxies for the singular vectors or eigenvectors of the input matrix $\boldsymbol{A}$. Additionally, the factorized output leads to accelerated routines for computing matrix–vector products or solving regularized linear systems.

4.2. Consequences. As we have noted, our goal is to produce an approximation that is close to the input matrix in spectral norm, say, $\|\boldsymbol{A} - \hat{\boldsymbol{A}}\| \leq \varepsilon$. This type of bound allows us to substitute the approximation in place of the input matrix in many different contexts.

1. **Matrix–vector multiplications**. We can use the low-rank approximation $\hat{\boldsymbol{A}}$ for matrix–vector products, and the error is controlled as

$$\|\boldsymbol{A}\boldsymbol{v} - \hat{\boldsymbol{A}}\boldsymbol{v}\| \leq \|\boldsymbol{A} - \hat{\boldsymbol{A}}\|\|\boldsymbol{v}\|.$$

2. **Regularized linear systems**. If $\boldsymbol{A}$ is psd and we generate a psd Nyström approximation $\hat{\boldsymbol{A}}$, we can use the approximation to solve the regularized linear system $(\boldsymbol{A} + \mu\mathbf{I})\boldsymbol{x} = \boldsymbol{y}$ with $\mu > 0$. By the resolvent identity, the error is

controlled by

$$\|(\boldsymbol{A}+\mu\mathbf{I})^{-1}-(\hat{\boldsymbol{A}}+\mu\mathbf{I})^{-1}\| = \|(\boldsymbol{A}+\mu\mathbf{I})^{-1}(\boldsymbol{A}-\hat{\boldsymbol{A}})(\hat{\boldsymbol{A}}+\mu\mathbf{I})^{-1}\| \leq \frac{\|\boldsymbol{A}-\hat{\boldsymbol{A}}\|}{\mu^2}.$$

Hence, the linear solve is guaranteed to be accurate as long as $\|\boldsymbol{A}-\hat{\boldsymbol{A}}\| \ll \mu^2$.

3. **Singular values**. We can use the low-rank approximation $\hat{\boldsymbol{A}}$ to compute singular values. Weyl's inequality [56, Thm. 4.3.1] guarantees that the error is controlled by

$$|\sigma_i(\boldsymbol{A}) - \sigma_i(\hat{\boldsymbol{A}})| \leq \|\boldsymbol{A}-\hat{\boldsymbol{A}}\|$$

for each $1 \leq i \leq \min\{L, N\}$.

4. **Singular vectors**. The low-rank approximation $\hat{\boldsymbol{A}}$ can be used to compute singular vectors. Wedin's theorem [108, 95] (also see [80, Thm. 4]) shows that

$$\langle \boldsymbol{u}_i(\boldsymbol{A}), \boldsymbol{u}_i(\hat{\boldsymbol{A}})\rangle^2 + \langle \boldsymbol{v}_i(\boldsymbol{A}), \boldsymbol{v}_i(\hat{\boldsymbol{A}})\rangle^2 \geq 2 - \frac{8\|\boldsymbol{A}-\hat{\boldsymbol{A}}\|^2}{\delta_i^2(\boldsymbol{A})}. \tag{4.2}$$

for each $1 \leq i < \min\{L, N\}$. Here, $\boldsymbol{u}_i(\boldsymbol{M})$ denotes the ith left singular vector, $\boldsymbol{v}_i(\boldsymbol{M})$ denotes the ith right singular vector, and $\delta_i(\boldsymbol{M}) = \min_{j\neq i} |\sigma_j(\boldsymbol{M}) - \sigma_i(\boldsymbol{M})|$ denotes the ith singular value gap. As a consequence of (4.2), the ith singular vectors of $\boldsymbol{A}$ and $\hat{\boldsymbol{A}}$ are closely aligned as long as $\|\boldsymbol{A}-\hat{\boldsymbol{A}}\| \ll \delta_i(\boldsymbol{A})$.

In summary, randomized low-rank approximation can be a helpful tool for performing many fast, approximate matrix computations.

5. Pseudocode. In this section, we present pseudocode for all of the algorithms we study. Efficient versions of `RSVD`, `RSI`, `NysSVD` and `NysSI` have already appeared in [87, 54, 50, 100], but we have developed a faster implementation of `RBKI`. Our `RBKI` pseudocode stores and reuses matrix multiplications, which reduces the number of matrix–vector products by roughly 33%. Additionally, to the best of our knowledge, the `NysBKI` algorithm is new, and it provides an additional factor-of-$\sqrt{2}$ savings in the number of matrix–vector products needed to achieve a fixed accuracy.

The section is structured as follows: subsections 5.1 to 5.3 provide pseudocode for `RSVD`, `RSI`, and `RBKI`; subsection 5.4 discusses Nyström approximation algorithms. and subsection 5.5 discusses modifications to the pseudocode that reduce or eliminate orthogonalization steps.

5.1. `RSVD`. The simplest approach for randomized low-rank matrix approximation is the *randomized singular value decomposition*, which was introduced in [81, 91] and refined in [54]. `RSVD` has been applied to thousands of problems in spectral clustering [20], biology [55, 103], uncertainty quantification [22, 58], and image processing [25, 61]. The algorithm is now implemented in scikit-learn using the command "randomized_svd" with parameter "n_iter" set to 0 [84] and in Matlab using the command "svdsketch" with parameter "NumPowerIterations" set to 0 [72].

`RSVD` generates a low-rank approximation

$$\hat{\boldsymbol{A}} = \boldsymbol{\Pi}_{\boldsymbol{A}\boldsymbol{\Omega}}\boldsymbol{A}, \tag{RSVD}$$

where $\boldsymbol{\Omega} \in \mathbb{R}^{N\times k}$ is a random matrix, typically Gaussian, and k is a block size parameter chosen by the user, typically $10 \leq k \leq 1{,}000$. Algorithm 5.1 provides `RSVD` pseudocode, which is optimized for efficiency in three ways:

Algorithm 5.1 `RSVD`

Input: Matrix $\boldsymbol{A} \in \mathbb{R}^{L\times N}$; block size k
Output: Orthogonal $\boldsymbol{U} \in \mathbb{R}^{L\times k}$, orthogonal $\boldsymbol{V} \in \mathbb{R}^{N\times k}$, and diagonal $\boldsymbol{\Sigma} \in \mathbb{R}^{k\times k}$ such that $\boldsymbol{A} \approx \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$

```
Generate a random matrix Ω ∈ R^{N×k}
X = AΩ
[X, ∼] = qr_econ(X)          ▷ economy-sized QR or stabilized QR (5.1)
Y = A*X
[Û, Σ, V] = svd_econ(Y*)     ▷ economy-sized SVD factorization
U = XÛ
```

1. The low-rank approximation

$$\boldsymbol{\Pi}_{\boldsymbol{A}\boldsymbol{\Omega}}\boldsymbol{A} = \boldsymbol{A}\boldsymbol{\Omega}(\boldsymbol{A}\boldsymbol{\Omega})^{\dagger}\boldsymbol{A}$$

is generated without forming the pseudoinverse $(\boldsymbol{A}\boldsymbol{\Omega})^{\dagger}$. As a cheaper and stabler approach, the columns of $\boldsymbol{A}\boldsymbol{\Omega}$ are orthogonalized to produce a matrix $\boldsymbol{X} = \texttt{orth}(\boldsymbol{A}\boldsymbol{\Omega})$ and the low-rank approximation is generated using

$$\boldsymbol{\Pi}_{\boldsymbol{A}\boldsymbol{\Omega}}\boldsymbol{A} = \boldsymbol{X}(\boldsymbol{X}^*\boldsymbol{A}).$$

2. The orthogonalization step $\boldsymbol{X} = \texttt{orth}(\boldsymbol{A}\boldsymbol{\Omega})$ can be performed by taking the $\boldsymbol{Q}$ matrix from a standard economy-sized QR factorization. However, this leads to numerical issues when the requested rank of $\hat{\boldsymbol{A}}$ exceeds the true rank of $\boldsymbol{A}$. For such problems, we advocate a stabler approach in which we first calculate the SVD

$$\boldsymbol{A}\boldsymbol{\Omega} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*.$$

Then, we identify all the indices $i = s_1, \dots, s_r$ for which the diagonal entry σ_{ii} exceeds a threshold proportional to the machine precision $\varepsilon_{\text{mach}}$ and return the approximate factorization

$$\boldsymbol{A}\boldsymbol{\Omega} \approx \boldsymbol{Q}\boldsymbol{R}, \text{ where } \begin{cases} \boldsymbol{Q} = \begin{bmatrix} \boldsymbol{u}_{s_1} & \cdots & \boldsymbol{u}_{s_r} \end{bmatrix}, \\ \boldsymbol{R} = \begin{bmatrix} \sigma_{s_1}\boldsymbol{v}_{s_1} & \cdots & \sigma_{s_r}\boldsymbol{v}_{s_r} \end{bmatrix}^*. \end{cases} \tag{5.1}$$

This is not a traditional QR factorization since $\boldsymbol{R}$ is not upper triangular, but it is a stabilized QR factorization since it reliably exposes the range of $\boldsymbol{A}\boldsymbol{\Omega}$ [46, Sec. 5.5.8]. We can take $\texttt{orth}(\boldsymbol{A}\boldsymbol{\Omega})$ to be the $\boldsymbol{Q}$ factor from the stabilized QR factorization, as usual.

3. The SVD of $\boldsymbol{X}(\boldsymbol{X}^*\boldsymbol{A})$ is evaluated by first computing the SVD of the wide matrix

$$\boldsymbol{X}^*\boldsymbol{A} = \hat{\boldsymbol{U}}\boldsymbol{\Sigma}\boldsymbol{V}^*$$

and then applying the rotation

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

These implementation techniques, which are mostly standard [54, Sec. 1.5], lead to the fast and stable pseudocode that is presented in Algorithm 5.1.

Next, we comment on the main user choice when implementing RSVD: choosing the block size k. As a general rule, increasing the block size k increases the cost of RSVD, but it also improves the approximation quality. We provide a quick proof of this fact below:

LEMMA 5.1 (Bigger projection, smaller error). *Consider a matrix $\boldsymbol{A} \in \mathbb{R}^{L\times N}$ and orthogonal projections $\boldsymbol{P}_1, \boldsymbol{P}_2 \in \mathbb{R}^{L\times L}$ such that* $\operatorname{range}(\boldsymbol{P}_1) \subseteq \operatorname{range}(\boldsymbol{P}_2)$. *Then,*

$$\|\boldsymbol{A} - \boldsymbol{P}_2\boldsymbol{A}\|_p \le \|\boldsymbol{A} - \boldsymbol{P}_1\boldsymbol{A}\|_p \tag{5.2}$$

for any Schatten p-norm with $1 \le p \le \infty$. The same inequality holds for any unitarily invariant norm.

Proof. Because $\operatorname{range}(\boldsymbol{P}_1) \subseteq \operatorname{range}(\boldsymbol{P}_2)$, we have $\boldsymbol{P}_1 \preccurlyeq \boldsymbol{P}_2$, and the complementary projectors satisfy $\mathbf{I} - \boldsymbol{P}_2 \preccurlyeq \mathbf{I} - \boldsymbol{P}_1$. Since conjugation preserves the psd order,

$$\boldsymbol{A}^*(\mathbf{I} - \boldsymbol{P}_2)\boldsymbol{A} \preccurlyeq \boldsymbol{A}^*(\mathbf{I} - \boldsymbol{P}_1)\boldsymbol{A}.$$

Consequently, for each $1 \le i \le \min\{L, N\}$,

$$\begin{aligned}\sigma_i(\boldsymbol{A} - \boldsymbol{P}_2\boldsymbol{A})^2 &= \lambda_i(\boldsymbol{A}^*(\mathbf{I} - \boldsymbol{P}_2)\boldsymbol{A}) \\ &\le \lambda_i(\boldsymbol{A}^*(\mathbf{I} - \boldsymbol{P}_1)\boldsymbol{A}) = \sigma_i(\boldsymbol{A} - \boldsymbol{P}_1\boldsymbol{A})^2.\end{aligned}$$

The inequality is Weyl's monotonicity theorem [56, Thm. 4.3.1]. This statement implies the p-norm inequality (5.2) and the result for unitarily invariant norms. ∎

Since increasing the block size k improves the approximation quality, the user of RSVD can always raise k to address more difficult problems. As an adaptive strategy, the user can increase k until the Frobenius–norm error hits a target precision, i.e., $\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_{\mathrm{F}}^2 = \|\boldsymbol{A}\|_{\mathrm{F}}^2 - \|\hat{\boldsymbol{A}}\|_{\mathrm{F}}^2 < \varepsilon$. This adaptive version of RSVD is currently implemented in the "svdsketch" function for Matlab [72]. For more discussion of the block size and adaptive stopping rules, see section 6.

5.2. RSI. The early developers of RSVD acknowledged that the algorithm leads to large random errors when applied to a matrix with slowly decaying singular values. When targeting a matrix with slow singular value decay, they proposed an alternative strategy called *randomized subspace iteration* [87, 54]. RSI is based on the randomized low-rank approximation

$$\hat{\boldsymbol{A}} = \boldsymbol{\Pi}_{(\boldsymbol{A}\boldsymbol{A}^*)^{q-1}\boldsymbol{A}\boldsymbol{\Omega}}\boldsymbol{A}, \tag{RSI}$$

where $\boldsymbol{\Omega} \in \mathbb{R}^{N\times k}$ is a random matrix, typically Gaussian, and q is a depth parameter, typically $2 \le q \le 5$. RSI is implemented in scikit-learn using the command "randomized_svd" [84] and in Matlab using the command "svdsketch" [72].

RSI requires q matrix multiplications with $\boldsymbol{A}$ and q multiplications with $\boldsymbol{A}^*$. The algorithm with $q = 1$ is identical to RSVD. Increasing the depth q systematically improves the approximation quality, but the cost grows linearly with q.

We provide simple, stable pseudocode for RSI in Algorithm 5.2. However, note that the pseudocode has the slightly awkward requirement of performing an *even* number of matrix multiplications (q multiplications with $\boldsymbol{A}$ and q multiplications with $\boldsymbol{A}^*$). Therefore we ask, "Is there any way to perform RSI with an *odd* number

Algorithm 5.2 `RSI`, simple version

Input: Matrix $\boldsymbol{A} \in \mathbb{R}^{L\times N}$; block size k; iteration count q
Output: Orthogonal $\boldsymbol{U} \in \mathbb{R}^{L\times k}$, orthogonal $\boldsymbol{V} \in \mathbb{R}^{N\times k}$, and diagonal $\boldsymbol{\Sigma} \in \mathbb{R}^{k\times k}$ such that $\boldsymbol{A} \approx \boldsymbol{U\Sigma V}^*$

Generate a random matrix $\boldsymbol{Y} \in \mathbb{R}^{N\times k}$
for $i = 1, \dots, q$ **do**
  $\boldsymbol{X} = \boldsymbol{AY}$
  $[\boldsymbol{X}, \sim] = \mathrm{qr\_econ}(\boldsymbol{X})$ ▷ Optional: stabilized QR (5.1)
  $\boldsymbol{Y} = \boldsymbol{A}^*\boldsymbol{X}$
end for
$[\hat{\boldsymbol{U}}, \boldsymbol{\Sigma}, \boldsymbol{V}] = \mathrm{svd\_econ}(\boldsymbol{Y}^*)$
$\boldsymbol{U} = \boldsymbol{X}\hat{\boldsymbol{U}}$

Algorithm 5.3 `RSI`, extended version

Input: Matrix $\boldsymbol{A} \in \mathbb{R}^{L\times N}$; block size k; stopping criterion
Output: Orthogonal $\boldsymbol{U} \in \mathbb{R}^{L\times k}$, orthogonal $\boldsymbol{V} \in \mathbb{R}^{N\times k}$, and diagonal $\boldsymbol{\Sigma} \in \mathbb{R}^{k\times k}$ such that $\boldsymbol{A} \approx \boldsymbol{U\Sigma V}^*$

Generate a random matrix $\boldsymbol{Y} \in \mathbb{R}^{N\times k}$
$i = 1$ ▷ i counts the number of multiplications
$\boldsymbol{X} = \boldsymbol{AY}$
$[\boldsymbol{X}, \sim] = \mathrm{qr\_econ}(\boldsymbol{X})$
while stopping criterion is not met **do**
  $i = i + 1$
  if i is even **then**
    $\boldsymbol{Y} = \boldsymbol{A}^*\boldsymbol{X}$
    $[\boldsymbol{Y}, \boldsymbol{R}] = \mathrm{qr\_econ}(\boldsymbol{Y})$ ▷ Optional: stabilized QR (5.1)
    $\boldsymbol{T} = \boldsymbol{R}^*$ ▷ $\boldsymbol{T}$ is lower triangular
  else
    $\boldsymbol{X} = \boldsymbol{AY}$
    $[\boldsymbol{X}, \boldsymbol{S}] = \mathrm{qr\_econ}(\boldsymbol{X})$ ▷ Optional: stabilized QR (5.1)
    $\boldsymbol{T} = \boldsymbol{S}$ ▷ $\boldsymbol{T}$ is upper triangular
  end if
end while
$[\hat{\boldsymbol{U}}, \boldsymbol{\Sigma}, \hat{\boldsymbol{V}}] = \mathrm{svd\_econ}(\boldsymbol{T})$
$\boldsymbol{U} = \boldsymbol{X}\hat{\boldsymbol{U}}$
$\boldsymbol{V} = \boldsymbol{Y}\hat{\boldsymbol{V}}$

of multiplications?" Bjarkason [14] answered this question in the affirmative (also see earlier work of [87, Sec. 4.3]), by replacing the approximation $\hat{\boldsymbol{A}} = \boldsymbol{\Pi}_{(\boldsymbol{AA}^*)^{q-1}\boldsymbol{A\Omega}}\boldsymbol{A}$ with the closely related approximation

$$\hat{\boldsymbol{A}} = \boldsymbol{A}\boldsymbol{\Pi}_{(\boldsymbol{A}^*\boldsymbol{A})^q\boldsymbol{\Omega}}. \tag{5.3}$$

The approach requires $q+1$ matrix-matrix multiplications with $\boldsymbol{A}$ and only q matrix–matrix multiplications with $\boldsymbol{A}^*$. When multiplications with $\boldsymbol{A}$ are much cheaper than multiplications with $\boldsymbol{A}^*$ (as in some PDE models [14]), we essentially get the final matrix multiplication for free, and we obtain an improved approximation.

Bjarkason's insight leads to an extended version of `RSI` that uses either an odd or an even number of matrix multiplications. We provide an implementation in Al-

gorithm 5.3, which incorporates the following features:

1. The algorithm generates orthonormal matrices $\boldsymbol{X} \in \mathbb{R}^{L\times k}$ and $\boldsymbol{Y} \in \mathbb{R}^{N\times k}$, representing the range and co-range of the low-rank approximation. The low-rank approximation takes the form $\hat{\boldsymbol{A}} = \boldsymbol{X}\boldsymbol{T}\boldsymbol{Y}^*$ for a core matrix $\boldsymbol{T} \in \mathbb{R}^{k\times k}$.
2. At odd iterations, the algorithm updates $\boldsymbol{X}$ and $\boldsymbol{T}$. At even iterations, the algorithm updates $\boldsymbol{Y}$ and $\boldsymbol{T}$.
3. The algorithm uses the fact that

$$\boldsymbol{T} = \hat{\boldsymbol{U}}\boldsymbol{\Sigma}\hat{\boldsymbol{V}}^* \quad \text{implies} \quad \hat{\boldsymbol{A}} = (\boldsymbol{X}\hat{\boldsymbol{U}})\boldsymbol{\Sigma}(\boldsymbol{Y}\hat{\boldsymbol{V}})^*,$$

to efficiently compute an SVD for the low-rank approximation $\hat{\boldsymbol{A}}$.

To optimize the number of multiplications, we can run Algorithm 5.3 with an adaptive stopping criterion. For example, we can stop the algorithm as soon as the square Frobenius–norm error $\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_{\mathrm{F}}^2 = \|\boldsymbol{A}\|_{\mathrm{F}}^2 - \|\hat{\boldsymbol{A}}\|_{\mathrm{F}}^2$ falls below a target precision $\varepsilon^2 \cdot \|\boldsymbol{A}\|_{\mathrm{F}}^2$. See section 6 for a discussion.

5.3. `RBKI`**.** In light of Lemma 5.1, it is always better to project onto a bigger subspace, and *randomized block Krylov iteration* projects onto the biggest subspace generated by the sequential matrix products. We define the Krylov subspace by

$$\begin{aligned}\mathcal{K}_q(\boldsymbol{A}\boldsymbol{A}^*; \boldsymbol{A}\boldsymbol{\Omega}) &= \operatorname{range}\begin{bmatrix}\boldsymbol{A}\boldsymbol{\Omega} & (\boldsymbol{A}\boldsymbol{A}^*)\boldsymbol{A}\boldsymbol{\Omega} & \cdots & (\boldsymbol{A}\boldsymbol{A}^*)^{q-1}\boldsymbol{A}\boldsymbol{\Omega}\end{bmatrix} \\ &= \bigcup\nolimits_{\deg(\phi)\le q-1} \operatorname{range}[\phi(\boldsymbol{A}\boldsymbol{A}^*)\boldsymbol{A}\boldsymbol{\Omega}].\end{aligned}$$

In this expression, ϕ varies over polynomials with degree not exceeding $q-1$. The `RBKI` method compresses the input matrix to the Krylov subspace:

(`RBKI`) $$\hat{\boldsymbol{A}} = \boldsymbol{\Pi}_{\left[\boldsymbol{A}\boldsymbol{\Omega} \ \cdots \ (\boldsymbol{A}\boldsymbol{A}^*)^{q-1}\boldsymbol{A}\boldsymbol{\Omega}\right]}\boldsymbol{A}.$$

The Krylov subspace is much larger than the projection space used in `RSI`, which only contains the range of the final power $(\boldsymbol{A}\boldsymbol{A}^*)^{q-1}\boldsymbol{A}\boldsymbol{\Omega}$. As a consequence, the `RBKI` approximation has the potential to be much more accurate.

`RBKI` was first introduced by Rokhlin, Szlam, and Tygert [87]. The paper [53] contains a numerical study in the context of principal component analysis. The first theoretical analysis is due to Musco and Musco [74]. Over the past decade, `RBKI` has been applied to problems in genetics [41], meteorology [19], and geophysical imaging [111], leading to reported higher accuracy than `RSI`. Nonetheless, `RBKI` has remained under-utilized in applications (as reflected in the computational genetics review [103]).

Algorithm 5.4 presents pseudocode for a simple version of `RBKI`, while Algorithm 5.5 displays pseudocode for an extended version of `RBKI` that uses an even or odd number of multiplications. Note that the block Gram–Schmidt step (lines 4 and 5 in Algorithm 5.4) must be performed twice for numerical stability. In subsequent algorithm displays, we indicate this repetition with the label (2×) rather than writing out the formula twice.

Our `RBKI` pseudocode is more efficient than existing `RBKI` implementations [87, 53, 74, 41, 113, 33, 14, 111, 70]. Indeed, the existing procedures all require the following series of multiplications:

(a) q times: multiply $\boldsymbol{A}$ with a matrix of size $N \times k$;
(b) $q-1$ times: multiply $\boldsymbol{A}^*$ with a matrix of size $L \times k$; and
(c) 1 time: multiply $\boldsymbol{A}^*$ with a block matrix of size $L \times kq$.

In contrast, Algorithm 5.4 uses a reduced set of multiplications:

(a) q times: multiply $\boldsymbol{A}$ with a matrix of size $N \times k$; and

Algorithm 5.4 `RBKI`, simple version

Input: Matrix $\boldsymbol{A} \in \mathbb{R}^{L \times N}$; block size k; iteration count q
Output: Orthogonal $\boldsymbol{U} \in \mathbb{R}^{L \times qk}$, orthogonal $\boldsymbol{V} \in \mathbb{R}^{N \times qk}$, and diagonal $\boldsymbol{\Sigma} \in \mathbb{R}^{qk \times qk}$ such that $\boldsymbol{A} \approx \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$

Generate a random matrix $\boldsymbol{Y}_0 \in \mathbb{R}^{N \times K}$.
for $i = 1, \dots, q$ **do**
  $\boldsymbol{X}_i = \boldsymbol{A}\boldsymbol{Y}_{i-1}$
  $\boldsymbol{X}_i = \boldsymbol{X}_i - \sum_{j<i} \boldsymbol{X}_j(\boldsymbol{X}_j^*\boldsymbol{X}_i)$ ▷ Orthogonalize w.r.t. past iterates
  $\boldsymbol{X}_i = \boldsymbol{X}_i - \sum_{j<i} \boldsymbol{X}_j(\boldsymbol{X}_j^*\boldsymbol{X}_i)$ ▷ The repetition ensures stability
  $[\boldsymbol{X}_i, \sim] = \mathrm{qr\_econ}(\boldsymbol{X}_i)$ ▷ Optional: stabilized QR (5.1)
  $\boldsymbol{Y}_i = \boldsymbol{A}^*\boldsymbol{X}_i$
end for
$[\hat{\boldsymbol{U}}, \boldsymbol{\Sigma}, \boldsymbol{V}] = \mathrm{svd\_econ}\big(\begin{bmatrix}\boldsymbol{Y}_1 & \cdots & \boldsymbol{Y}_q\end{bmatrix}^*\big)$
$\boldsymbol{U} = \begin{bmatrix}\boldsymbol{X}_1 & \cdots & \boldsymbol{X}_q\end{bmatrix}\hat{\boldsymbol{U}}$

(b) q times: multiply $\boldsymbol{A}^*$ with a matrix of size $L \times k$.

Our new procedure removes step (c), which involves an expensive multiplication of $\boldsymbol{A}^*$ with a block matrix of size $L \times kq$. We have substituted a single multiplication with a smaller matrix of size $L \times k$.

If we measure computational cost in the simplest way, by counting matrix–vector products, our new pseudocode results in cost savings of roughly 33%. The savings are even higher when matrix–vector multiplications are more expensive with $\boldsymbol{A}^*$ than with $\boldsymbol{A}$ (as in some PDE models [14]). Our new pseudocode does not change the asymptotic arithmetic cost of `RBKI`, which is still $\mathcal{O}(kmLN)$ for dense matrices, yet it makes a noticeable difference in applications. There are also potential reductions in communication costs.

Our new pseudocode enables a clean comparison between `RSI` and `RBKI` because the two algorithms now perform matrix multiplications with precisely the same block size. Although `RBKI` performs more arithmetic than `RSI` outside the matrix products, this arithmetic is rarely the predominant cost of the algorithm. We acknowledge that `RBKI` requires $\mathcal{O}(km(L+N))$ storage whereas `RSI` only requires $\mathcal{O}(k(L+N))$ storage. However, in typical applications involving data matrices (see section 7), this is not an issue because storing the low-rank approximation $\hat{\boldsymbol{A}}$ is much cheaper than storing the original matrix $\boldsymbol{A}$.

As the main distinction, `RBKI` makes better use of the matrix products than `RSI`, resulting in a far more accurate approximation. Overall, we suspect that `RSI` is preferable to `RBKI` only if $\boldsymbol{A}$ is an abstract linear operator and working storage is the limiting factor for the computation.

5.4. Positive-semidefinite matrices. Suppose that $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ is psd and we have generated a matrix $\boldsymbol{M} \in \mathbb{R}^{N \times k}$ whose range is aligned with leading eigenvectors of $\boldsymbol{A}$. Then, the simplest approximation for $\boldsymbol{A}$, which is used in `RSVD`, `RSI`, and `RBKI`, is based on compressing the range or co-range of $\boldsymbol{A}$ as follows:

$$\boldsymbol{\Pi}_{\boldsymbol{M}}\boldsymbol{A} = \boldsymbol{M}(\boldsymbol{M}^*\boldsymbol{M})^\dagger(\boldsymbol{A}\boldsymbol{M})^* \qquad \text{or} \qquad \boldsymbol{A}\boldsymbol{\Pi}_{\boldsymbol{M}} = (\boldsymbol{A}\boldsymbol{M})(\boldsymbol{M}^*\boldsymbol{M})^\dagger\boldsymbol{M}^*.$$

Algorithm 5.5 `RBKI`, extended version

Input: Matrix $\boldsymbol{A} \in \mathbb{R}^{L\times N}$; block size k; stopping criterion
Output: Orthogonal $\boldsymbol{U}$, orthogonal $\boldsymbol{V}$, and diagonal $\boldsymbol{\Sigma}$ such that $\boldsymbol{A} \approx \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$

Generate a random matrix $\boldsymbol{Y}_0 \in \mathbb{R}^{N\times k}$
$i = 1$ ▷ i counts the number of multiplications
$\boldsymbol{X}_1 = \boldsymbol{A}\boldsymbol{Y}_0$
$[\boldsymbol{X}_1, \sim] = \text{qr\_econ}(\boldsymbol{X}_1)$
$\boldsymbol{R} = [\,]; \boldsymbol{S} = [\,]$
while stopping criterion is not met **do**
  $i = i + 1$
  if i is even **then**
    $\boldsymbol{Y}_i = \boldsymbol{A}^*\boldsymbol{X}_{i-1}$
    $\boldsymbol{R}_{\bullet i} = \begin{bmatrix}\boldsymbol{Y}_2 & \boldsymbol{Y}_4 & \cdots & \boldsymbol{Y}_{i-2}\end{bmatrix}^* \boldsymbol{Y}_i$
    $\boldsymbol{Y}_i = \boldsymbol{Y}_i - \sum_{\text{even } j<i} \boldsymbol{Y}_j(\boldsymbol{Y}_j^*\boldsymbol{Y}_i)$ ▷ Orthog. w.r.t. even iterates (2×)
    $[\boldsymbol{Y}_i, \boldsymbol{R}_{ii}] = \text{qr\_econ}(\boldsymbol{Y}_i)$ ▷ Optional: stabilized QR (5.1)
    $\boldsymbol{R} = \begin{bmatrix}\boldsymbol{R} & \boldsymbol{R}_{\bullet i}\\ \boldsymbol{0} & \boldsymbol{R}_{ii}\end{bmatrix}$
    $\boldsymbol{T} = \boldsymbol{R}^*$
  else
    $\boldsymbol{X}_i = \boldsymbol{A}\boldsymbol{Y}_{i-1}$
    $\boldsymbol{S}_{\bullet i} = \begin{bmatrix}\boldsymbol{X}_1 & \boldsymbol{X}_3 & \cdots & \boldsymbol{X}_{i-2}\end{bmatrix}^* \boldsymbol{X}_i$
    $\boldsymbol{X}_i = \boldsymbol{X}_i - \sum_{\text{odd } j<i} \boldsymbol{X}_j(\boldsymbol{X}_j^*\boldsymbol{X}_i)$ ▷ Orthog. w.r.t. odd iterates (2×)
    $[\boldsymbol{X}_i, \boldsymbol{S}_{ii}] = \text{qr\_econ}(\boldsymbol{X}_i)$ ▷ Optional: stabilized QR (5.1)
    $\boldsymbol{S} = \begin{bmatrix}\boldsymbol{S} & \boldsymbol{S}_{\bullet i}\\ \boldsymbol{0} & \boldsymbol{S}_{ii}\end{bmatrix}$
    $\boldsymbol{T} = \boldsymbol{S}$
  end if
end while
$[\hat{\boldsymbol{U}}, \boldsymbol{\Sigma}, \hat{\boldsymbol{V}}] = \text{svd\_econ}(\boldsymbol{T})$
$\boldsymbol{U} = \begin{bmatrix}\boldsymbol{X}_1 & \boldsymbol{X}_3 & \cdots\end{bmatrix}\hat{\boldsymbol{U}}$
$\boldsymbol{V} = \begin{bmatrix}\boldsymbol{Y}_2 & \boldsymbol{Y}_4 & \cdots\end{bmatrix}\hat{\boldsymbol{V}}$

Since $\boldsymbol{A}$ is psd, we can employ the same data more effectively using the *Nyström approximation* [70, Sec. 14]:

$$\boldsymbol{A}\langle\boldsymbol{M}\rangle := \boldsymbol{A}^{1/2}\boldsymbol{\Pi}_{\boldsymbol{A}^{1/2}\boldsymbol{M}}\boldsymbol{A}^{1/2} = (\boldsymbol{A}\boldsymbol{M})(\boldsymbol{M}^*(\boldsymbol{A}\boldsymbol{M}))^\dagger(\boldsymbol{A}\boldsymbol{M})^*.$$

The Nyström approximation always improves over the simpler approximations.

LEMMA 5.2 (Nyström helps). *Consider a psd matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ and any matrix $\boldsymbol{M} \in \mathbb{R}^{N\times k}$. Then*

$$\|\boldsymbol{A} - \boldsymbol{A}\langle\boldsymbol{M}\rangle\|_p \le \|\boldsymbol{A} - \boldsymbol{\Pi}_{\boldsymbol{M}}\boldsymbol{A}\|_p = \|\boldsymbol{A} - \boldsymbol{A}\boldsymbol{\Pi}_{\boldsymbol{M}}\|_p \tag{5.4}$$

for any Schatten p-norm with $1 \le p \le \infty$. The same inequality holds for any unitarily invariant norm.

Proof. Since $\boldsymbol{\Pi}_{\boldsymbol{M}}\boldsymbol{A}$ and $\boldsymbol{A}\langle\boldsymbol{M}\rangle$ only depend on the range of $\boldsymbol{M}$, we can assume without loss of generality that $\boldsymbol{M}$ has orthonormal columns and change basis so that $\boldsymbol{M} = \left[\begin{smallmatrix}\mathbf{I}\\ \mathbf{0}\end{smallmatrix}\right]$. With this transformation, the matrices $\boldsymbol{A}$, $\boldsymbol{\Pi}_{\boldsymbol{M}}\boldsymbol{A}$, and $\boldsymbol{A}\langle\boldsymbol{M}\rangle$ can be

Algorithm 5.6 `NysSVD`

Input: Psd matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$; block size k; shift $\varepsilon > 0$
Output: Orthogonal $\boldsymbol{U} \in \mathbb{R}^{N\times k}$ and diagonal $\boldsymbol{\Lambda} \in \mathbb{R}^{k\times k}$ such that $\boldsymbol{A} \approx \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^*$

Generate a random matrix $\boldsymbol{\Omega} \in \mathbb{R}^{N\times k}$
$\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{\Omega} + \varepsilon\boldsymbol{\Omega}$ ▷ Apply ε shift
$\boldsymbol{C} = \mathrm{chol}(\boldsymbol{\Omega}^*\boldsymbol{Y})$
$\boldsymbol{Z} = \boldsymbol{Y}\boldsymbol{C}^{-1}$ ▷ Triangular solve
$[\boldsymbol{U}, \boldsymbol{\Sigma}, \sim] = \mathrm{svd\_econ}(\boldsymbol{Z})$
$\boldsymbol{\Lambda} = \max\{\boldsymbol{0}, \boldsymbol{\Sigma}^2 - \varepsilon\boldsymbol{I}\}$ ▷ Remove ε shift

written in block form:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{bmatrix}, \quad \boldsymbol{\Pi}_{\boldsymbol{M}}\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}, \quad \boldsymbol{A}\langle\boldsymbol{M}\rangle = \begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{21}\boldsymbol{A}_{11}^\dagger\boldsymbol{A}_{12} \end{bmatrix}.$$

Using the psd order relations $\boldsymbol{A}_{22} - \boldsymbol{A}_{21}\boldsymbol{A}_{11}^\dagger\boldsymbol{A}_{12} \preccurlyeq \boldsymbol{A}_{22}$ and $\boldsymbol{A}_{22}^2 \preccurlyeq \boldsymbol{A}_{22}^2 + \boldsymbol{A}_{21}\boldsymbol{A}_{12}$, we calculate

$$\begin{aligned} \sigma_i(\boldsymbol{A} - \boldsymbol{A}\langle\boldsymbol{M}\rangle)^2 &= \lambda_i\left(\begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}_{22} - \boldsymbol{A}_{21}\boldsymbol{A}_{11}^\dagger\boldsymbol{A}_{12} \end{bmatrix}\right)^2 \\ &\le \lambda_i\left(\begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}_{22} \end{bmatrix}\right)^2 = \lambda_i\left(\begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}_{22}^2 \end{bmatrix}\right) \\ &\le \lambda_i\left(\begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}_{22}^2 + \boldsymbol{A}_{21}\boldsymbol{A}_{12} \end{bmatrix}\right) = \lambda_i((\boldsymbol{A} - \boldsymbol{\Pi}_{\boldsymbol{M}}\boldsymbol{A})(\boldsymbol{A} - \boldsymbol{\Pi}_{\boldsymbol{M}}\boldsymbol{A})^*) \\ &= \sigma_i(\boldsymbol{A} - \boldsymbol{\Pi}_{\boldsymbol{M}}\boldsymbol{A})^2 \end{aligned}$$

for each $1 \le i \le N$, which confirms (5.4). ∎

Next, we show how to incorporate Nyström approximation in three different randomized low-rank approximation algorithms: `NysSVD` (subsection 5.4.1), `NysSI` (subsection 5.4.2), and `NysBKI` (subsection 5.4.3).

5.4.1. `NysSVD`**.** In *randomized Nyström approximation*, we generate a low-rank approximation:

(`NysSVD`) $$\hat{\boldsymbol{A}} = \boldsymbol{A}\langle\boldsymbol{\Omega}\rangle,$$

where $\boldsymbol{\Omega} \in \mathbb{R}^{N\times k}$ is a random matrix. In this work, $\boldsymbol{\Omega}$ is always a Gaussian matrix [100], mimicking the strategy used in `RSVD`. For parallelism, we call the resulting algorithm `NysSVD` even though it produces an eigenvalue decomposition.

Algorithm 5.6 presents `NysSVD` pseudocode, which is optimized for efficiency in the following ways:

1. The low-rank approximation

$$\boldsymbol{A}\langle\boldsymbol{\Omega}\rangle = \boldsymbol{A}\boldsymbol{\Omega}(\boldsymbol{\Omega}^*\boldsymbol{A}\boldsymbol{\Omega})^\dagger\boldsymbol{\Omega}^*\boldsymbol{A}^*,$$

is generated without ever forming the pseudoinverse $(\boldsymbol{\Omega}^*\boldsymbol{A}\boldsymbol{\Omega})^\dagger$ explicitly. Instead, we compute an upper-triangular Cholesky factor $\boldsymbol{C} \in \mathbb{R}^{k\times k}$ so that $\boldsymbol{\Omega}^*(\boldsymbol{A}\boldsymbol{\Omega}) = \boldsymbol{C}^*\boldsymbol{C}$. The low-rank approximation is generated as

$$\boldsymbol{A}\langle\boldsymbol{\Omega}\rangle = (\boldsymbol{A}\boldsymbol{\Omega}\boldsymbol{C}^\dagger)(\boldsymbol{A}\boldsymbol{\Omega}\boldsymbol{C}^\dagger)^*.$$

Algorithm 5.7 `NysSI`

Input: Psd matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$; block size k; stopping criterion; shift $\varepsilon > 0$
Output: Orthogonal $\boldsymbol{U} \in \mathbb{R}^{N\times k}$ and diagonal $\boldsymbol{\Lambda} \in \mathbb{R}^{k\times k}$ such that $\boldsymbol{A} \approx \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^*$

Generate a random matrix $\boldsymbol{Y} \in \mathbb{R}^{N\times k}$
$i = 0$
while stopping criterion is not met **do**
  $i = i + 1$
  $[\boldsymbol{X}, \sim] = \text{qr\_econ}(\boldsymbol{Y})$ ▷ Optional: stabilized QR (5.1)
  $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}$
end while
$\boldsymbol{Y} = \boldsymbol{Y} + \varepsilon\boldsymbol{X}$ ▷ Apply ε shift
$\boldsymbol{C} = \text{chol}(\boldsymbol{X}^*\boldsymbol{Y})$ ▷ Cholesky decomposition
$\boldsymbol{Z} = \boldsymbol{Y}\boldsymbol{C}^{-1}$ ▷ Triangular solve
$[\boldsymbol{U}, \boldsymbol{\Sigma}, \sim] = \text{svd\_econ}(\boldsymbol{Z})$
$\boldsymbol{\Lambda} = \max\{\boldsymbol{0}, \boldsymbol{\Sigma}^2 - \varepsilon\mathbf{I}\}$ ▷ Remove ε shift

2. To ensure numerical stability, the Nyström approximation is applied to a shifted operator $\boldsymbol{A}_\varepsilon = \boldsymbol{A} + \varepsilon\mathbf{I}$, where the shift parameter $\varepsilon = \varepsilon_{\text{mach}}\text{tr}(\boldsymbol{A})$ depends on the machine precision. To approximately counteract the shift, the eigenvalues of the approximation $\hat{\boldsymbol{A}}_\varepsilon$ are all reduced by ε.

These improvements, previously recommended in [67], lead to the fast and robust `NysSVD` implementation in Algorithm 5.6.

Remark 5.3 (Column Nyström approximation). When evaluating each entry of the input matrix $\boldsymbol{A}$ is expensive, we might prefer to construct a Nyström approximation with the range determined from just a few columns of $\boldsymbol{A}$. Equivalently, the test matrix $\boldsymbol{\Omega}$ contains (random) coordinate vectors [109]. To find an informative set of columns, we need additional ideas; see the paper [24] and its background references.

5.4.2. `NysSI`**.** In *randomized subspace iteration with Nyström approximation* [54, Alg. 5.5], we approximate a psd matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ as:

$$\hat{\boldsymbol{A}} = \boldsymbol{A}\langle\boldsymbol{M}\rangle, \quad \text{where} \quad \boldsymbol{M} = \boldsymbol{A}^{m-1}\boldsymbol{\Omega}. \tag{NysSI}$$

Algorithm 5.7 presents `NysSI` pseudocode, which can be implemented with any adaptive stopping criterion. For example, we can stop the algorithm as soon as the trace–norm error $\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_* = \|\boldsymbol{A}\|_* - \|\hat{\boldsymbol{A}}\|_*$ falls below a target precision $\varepsilon \cdot \|\boldsymbol{A}\|_*$. `NysSI` achieves a target precision more quickly than `RSI` by one-half of a matrix–matrix multiplication, as reflected in our experiments (Figures 5 and 6) and theory (section 9).

5.4.3. `NysBKI`**.** In *randomized block Krylov iteration with Nyström approximation*, we approximate a psd matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ as

$$\hat{\boldsymbol{A}} = \boldsymbol{A}\langle\boldsymbol{M}\rangle, \quad \text{where} \quad \boldsymbol{M} = \begin{bmatrix}\boldsymbol{\Omega} & \boldsymbol{A}\boldsymbol{\Omega} & \boldsymbol{A}^2\boldsymbol{\Omega} & \cdots & \boldsymbol{A}^{m-1}\boldsymbol{\Omega}\end{bmatrix}. \tag{NysBKI}$$

`NysBKI` uses the output of *every* matrix multiplication, not just every second matrix multiplication, to build an enriched approximation space. As a result, `NysBKI` is more efficient than `RBKI` by a factor of $\sqrt{2}$ matrix–matrix multiplications (Figures 5 and 6 and section 9). To the best of our knowledge, `NysBKI` has not appeared in the matrix approximation literature before now. `NysBKI` pseudocode appears in Algorithm 5.8.

Algorithm 5.8 `NysBKI`

Input: Psd matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$; block size k; stopping criterion; shift $\varepsilon > 0$
Output: Orthogonal $\boldsymbol{U}$ and diagonal $\boldsymbol{\Lambda}$ such that $\boldsymbol{A} \approx \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^*$

1 Generate a random matrix $\boldsymbol{Y}_0 \in \mathbb{R}^{N\times k}$
2 $i = 0$
3 **while** stopping criterion is not met **do**
4 $\quad \boldsymbol{X}_i = \boldsymbol{Y}_i$
5 $\quad \boldsymbol{X}_i = \boldsymbol{X}_i - \sum_{j<i} \boldsymbol{X}_j(\boldsymbol{X}_j^*\boldsymbol{X}_i)$ ▷ Orthog. w.r.t. past iterates (2×)
6 $\quad [\boldsymbol{X}_i, \sim] = \mathrm{qr\_econ}(\boldsymbol{X}_i)$ ▷ Optional: stabilized QR (5.1)
7 $\quad \boldsymbol{Y}_{i+1} = \boldsymbol{A}\boldsymbol{X}_i + \varepsilon\boldsymbol{X}_i$
8 $\quad i = i + 1$
9 **end while**
10 $\boldsymbol{C} = \mathrm{chol}\big(\begin{bmatrix}\boldsymbol{X}_0 & \cdots & \boldsymbol{X}_{i-1}\end{bmatrix}^* \begin{bmatrix}\boldsymbol{Y}_1 & \cdots & \boldsymbol{Y}_i\end{bmatrix}\big)$
11 $\boldsymbol{Z} = \begin{bmatrix}\boldsymbol{Y}_1 & \cdots & \boldsymbol{Y}_i\end{bmatrix} \boldsymbol{C}^{-1}$
12 $[\boldsymbol{U}, \boldsymbol{\Sigma}, \sim] = \mathrm{svd\_econ}(\boldsymbol{Z})$
13 $\boldsymbol{\Lambda} = \max\{\boldsymbol{0}, \boldsymbol{\Sigma}^2 - \varepsilon\mathbf{I}\}$

5.5. Additional opportunities for speedups. So far, we have presented simple, stable pseudocode that requires the minimal number of matrix multiplications with $\boldsymbol{A}$ and $\boldsymbol{A}^*$. However, we have not necessarily optimized the operations involving smaller matrices, and the algorithms use a large number of QR and SVD factorizations. Here we discuss some possibilities for computational speedups by removing or replacing these factorizations. This section is intended for numerical linear algebra experts, and most readers can skip it with impunity.

As the first speedup opportunity, we can remove all the SVD steps in our pseudocode [25]. We can redesign `RSVD` to return a factorization $\hat{\boldsymbol{A}} = \boldsymbol{X}\boldsymbol{Y}^*$, without ever calculating the singular values and vectors. We can make similar adjustments to the pseudocode for other randomized low-rank approximation algorithms. After making these changes, the algorithms return the same low-rank approximation (in exact precision arithmetic), but the singular values and singular vectors of $\hat{\boldsymbol{A}}$ are no longer easily accessible as before.

As the second speedup opportunity, we can replace all the QR factorizations in the Nyström-based algorithms and all but the last QR factorization in `RSI` with cheaper matrix decompositions. Indeed, the sole purpose of the QR factorizations is to prevent the iterates from becoming ill-conditioned, but we can also avoid ill-conditioning using an LU factorization [67] or a randomized QR decomposition [9, 10]. These alternatives use roughly 50% of the arithmetic of the standard pivoted QR factorization.

For `RBKI` and `NysBKI`, we can also replace the block orthogonalization steps (such as lines 4–5 in Algorithm 5.4) by a shorter Lanczos-type recurrence [76]:

$$\boldsymbol{X}_i = \boldsymbol{X}_i - \sum\nolimits_{j=i-2}^{i-1} \boldsymbol{X}_j(\boldsymbol{X}_j^*\boldsymbol{X}_i).$$

This reduces the total orthogonalization cost from $\mathcal{O}(k^2q^2(L+N))$ to $\mathcal{O}(k^2q(L+N))$ arithmetic operations. On the other hand, in finite precision arithmetic it may result in a basis $\boldsymbol{M} = \begin{bmatrix}\boldsymbol{X}_1 & \cdots & \boldsymbol{X}_q\end{bmatrix}$ that is not orthogonal.

Last, orthogonalization is used in `RSVD`, `RSI`, and `RBKI` to compute a low-rank approximation $\hat{\boldsymbol{A}} = \boldsymbol{\Pi}_{\boldsymbol{M}}\boldsymbol{A}$ from a nonorthogonal basis matrix $\boldsymbol{M}$. However, if we are willing to accept some additional error, Nakatsukasa [75, Eq. 3] has proposed

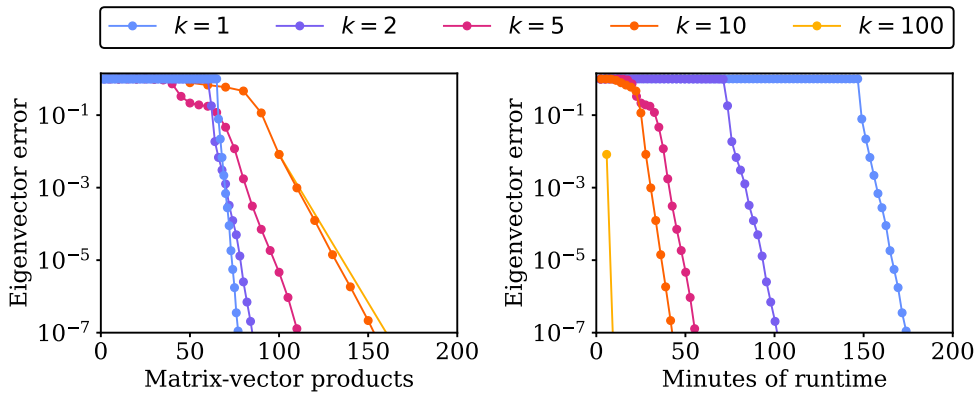

FIG. 2. ***Runtime comparisons.*** *Eigenvector approximation error of* `NysBKI` *with block size* $k = 1, 2, 5, 10,$ *or* 100 *applied to the* $250{,}000 \times 250{,}000$ *kernel matrix, as constructed in subsection 7.3. The left panel measures the number of matrix–vector products while the right panel measures the runtime.*

replacing $\boldsymbol{\Pi}_{\boldsymbol{M}}\boldsymbol{A} = \boldsymbol{M}\boldsymbol{M}^{\dagger}\boldsymbol{A}$ with a cheaper approximation

$$\hat{\boldsymbol{A}} = \boldsymbol{M}(\boldsymbol{S}\boldsymbol{M})^{\dagger}\boldsymbol{S}\boldsymbol{A}, \tag{5.5}$$

where $\boldsymbol{S}$ is a randomized embedding matrix that reduces the dimensionality of $\boldsymbol{M}$ and $\boldsymbol{A}$. Nakatsukasa [75] finds the additional error to be small in numerical tests.

6. Parameter choices. In this section, we discuss strategies for choosing the block size k and the number of matrix–matrix multiplications m in low-rank approximation algorithms. In `RSVD` and `NysSVD`, there is no choice but to increase k to handle more difficult problems. However, in the other algorithms, a major question concerns the relative advantages of using a large depth ($m \gg 1$) versus a large block size ($k \gg 1$). Section 6.1 considers the tradeoff between k and m, while subsection 6.2 discusses adaptive strategies for choosing k and m that ensure a high-quality approximation.

6.1. Tradeoff between the block size and depth. A recent analysis [73] of Meyer, Musco, and Musco gives evidence that $k = 1$ often leads to the highest accuracy in `RBKI`, assuming a fixed number of matrix–vector products km. Setting $k = 1$ leads to the classical Lanczos iteration. However, Meyer and coauthors also acknowledge two problems with setting $k = 1$. First, the approximation quality can be poor if any singular values have multiplicity greater than one (or have exponentially small singular value gaps), limiting the applicability to eigenvalue problems in physics and chemistry with high multiplicities (e.g., [42, Sec. 14]). Second, the Lanczos method is intrinsically serial, making it slow to run on modern computers.

We have performed a runtime analysis for `NysBKI` applied to the $250{,}000 \times 250{,}000$ kernel matrix that will be introduced later in subsection 7.3, leading to the results pictured in Figure 2. The figure presents the eigenvector approximation error

$$\|\boldsymbol{\Pi}_3(\boldsymbol{A}) - \boldsymbol{\Pi}_3(\hat{\boldsymbol{A}})\|,$$

where $\boldsymbol{\Pi}_3(\cdot)$ denotes the orthogonal projection onto the leadings three eigenvectors, and $\hat{\boldsymbol{A}}$ is the stochastic approximation of the kernel matrix $\boldsymbol{A}$ from a single run of

NysBKI. We vary the block size to be either $k = 1$, 2, 5, 10, or 100 (different color lines). The results in the left panel of Figure 2 support the conclusion of Meyer et al. [73] that the smallest block size $k = 1$ leads to the highest accuracy, given a fixed number of matrix–vector products. However, the number of matrix–vector multiplications is not a good indicator of the runtime cost, as shown in the right panel of Figure 2. To reach a tolerance of $\varepsilon = 10^{-5}$, NysBKI with a block size $k = 1$ takes 167 minutes on a laptop computer, whereas NysBKI with a block size $k = 100$ takes 11 minutes, **making it** $15\times$ **faster**. We often anticipate an order-of-magnitude speedup by switching to a much larger block size.

In earlier work, Li and coauthors [67] performed similar speed tests comparing a Matlab implementation of RSI against Lanczos implementations in ARPACK [65] and PROPACK [63]. Their results reinforce the practical advantages of using a large block size, both for reducing runtimes and for reducing errors:

> On strictly serial processors with no complicated caching (such as the processors of many decades ago), the most careful implementations of Lanczos iterations... could likely attain performance nearing the randomized methods'... The randomized methods can attain much higher performance on parallel and distributed processors and generally are easier to use—setting their parameters properly is trivial (defaults perform well)...

The conclusions of Li et al. are based on hundreds of tests with dense matrices as large as 100,000 × 100,000 and sparse matrices as large as 3,000,000 × 3,000,000.

In light of the computational evidence, we advocate using a block size of at least $k = 10$ and potentially $k = 100$–1,000 for large-scale problems ($N \geq 10^5$). After choosing the block size, we recommend running RBKI or NysBKI with an adaptive stopping rule to determine the minimal number of multiplications. See subsection 6.2 for a discussion of stopping rules that ensure the quality of the low-rank approximation.

6.2. Quality assurance. Here we evaluate two strategies for controlling the quality of a randomized low-rank approximation $\hat{\boldsymbol{A}}$. The first strategy is based on measuring and controlling the global approximation error. In this strategy, we increase the block size k or depth parameter m until achieving an error tolerance $\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_{\mathrm{F}} < \varepsilon \cdot \|\boldsymbol{A}\|_{\mathrm{F}}$ or $\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_{*} < \varepsilon \cdot \|\boldsymbol{A}\|_{*}$. These error bounds imply that $\hat{\boldsymbol{A}}$ can be reliably used in place of $\boldsymbol{A}$ in various matrix computations (subsection 4.2).

An RSI implementation that controls the global approximation error is provided in the "svdsketch" [72] function for Matlab, based on pseudocode from [112]:

- As the first step, this function evaluates the square Frobenius norm $\|\boldsymbol{A}\|_{\mathrm{F}}^2$ using a single pass through the entries of $\boldsymbol{A}$.
- At each subsequent step, the function adds new columns to the initialization matrix $\boldsymbol{\Omega}$, updates the low-rank approximation $\hat{\boldsymbol{A}}$, and updates the Frobenius norm $\|\hat{\boldsymbol{A}}\|_{\mathrm{F}}$.
- The procedure terminates as soon as

$$\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_{\mathrm{F}}^2 = \|\boldsymbol{A}\|_{\mathrm{F}}^2 - \|\hat{\boldsymbol{A}}\|_{\mathrm{F}}^2 < \varepsilon^2 \cdot \|\boldsymbol{A}\|_{\mathrm{F}}^2.$$

"Svdsketch" always returns a low-rank approximation which accounts for a $1 - \varepsilon^2$ proportion of the square Frobenius norm of the target matrix.

Two improvements to "svdsketch" could make the procedure even more powerful. First, "svdsketch" adaptively chooses the block size k, but our numerical experiments (section 7) suggest that adaptively choosing the number of multiplications m would

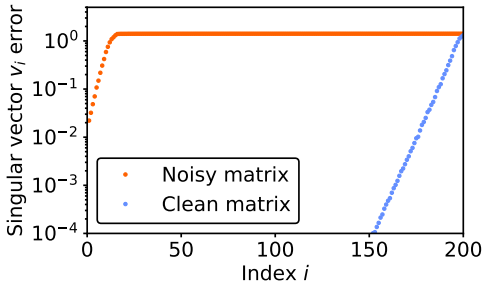

FIG. 3. ***(Accuracy of the singular vectors).*** *Singular vector approximation error for the clean matrix* $\boldsymbol{A}$ *(blue) and the noisy matrix* $\boldsymbol{B}$ *(orange), as constructed in subsection 2.3.*

lead to even greater efficiency. When computations are serial, adding to the depth more quickly reduces the errors than adding to the block size (Figure 2 left panel). Second, "svdsketch" can only be applied to matrices stored entry-wise on a computer, because of the way it calculates the Frobenius-norm error. The recent paper [37, Sec. 2] describes a different strategy for approximating the Frobenius norm error based on matrix multiplications, which remains valid even when $\boldsymbol{A}$ is an abstract linear operator.

Next we consider a different adaptive strategy, in which we increase k or m until each one of the leading singular vector triplets $(\hat{\boldsymbol{u}}_i, \hat{\sigma}_i, \hat{\boldsymbol{v}}_i)$ achieves the residual accuracy

$$r_i = \left(\|(\boldsymbol{A} - \hat{\boldsymbol{A}})^* \hat{\boldsymbol{u}}_i\|^2 + \|(\boldsymbol{A} - \hat{\boldsymbol{A}}) \hat{\boldsymbol{v}}_i\|^2 \right)^{1/2} < \varepsilon. \tag{6.1}$$

As soon as $r_i < \varepsilon$, the ith singular vector triplet is stable in the sense of backward error [52]. Namely, there is a perturbation matrix $\boldsymbol{E} \in \mathbb{R}^{L \times N}$ with $\|\boldsymbol{E}\|_{\mathrm{F}} < \varepsilon$ such that $\boldsymbol{A} + \boldsymbol{E}$ has exactly the singular vector triplet $(\hat{\boldsymbol{u}}_i, \hat{\sigma}_i, \hat{\boldsymbol{v}}_i)$. With assistance from Maksim Melnichenko and Riley Murray, we have developed adaptive versions of `RBKI` and `NysBKI` implementing this strategy, with the pseudocode appearing in Algorithms A.1 and A.2.

Both adaptive strategies presented here would lead to a more reliable approximation than the earlier approach [54, 70] of applying an r-truncated singular value decomposition to the low-rank approximation with a default value $r = k - 2$ or $r = \lfloor k/2 \rfloor$. A sufficiently small truncation parameter r can in principle eliminate inaccurate singular vector estimates, but selecting r is notoriously difficult in practice. For example, Figure 3 evaluates the error

$$\left(\mathbb{E}\|\boldsymbol{\Pi}_{\boldsymbol{v}_i(\boldsymbol{A})} - \boldsymbol{\Pi}_{\boldsymbol{v}_i(\hat{\boldsymbol{A}})}\|^2\right)^{1/2} \quad \text{or} \quad \left(\mathbb{E}\|\boldsymbol{\Pi}_{\boldsymbol{v}_i(\boldsymbol{B})} - \boldsymbol{\Pi}_{\boldsymbol{v}_i(\hat{\boldsymbol{B}})}\|^2\right)^{1/2},$$

for the estimated singular vectors of the clean matrix $\boldsymbol{A}$ and noisy matrix $\boldsymbol{B}$ described in subsection 2.3. Here, $\boldsymbol{\Pi}_{\boldsymbol{v}_i(\cdot)}$ denotes the orthogonal projection onto the ith right singular vector, and the expectation is evaluated over 100 runs of `RBKI` with $k = 100$ and $m = 5$. From the results in Figure 3, we would need to apply truncation with $r \leq 4$ for the noisy matrix but we could take r as high as 186 for the clean matrix, in order to achieve an error tolerance $\varepsilon = .1$. Unfortunately, there is no default truncation parameter r which performs well for both matrices.

7. Applications. In this section, we present experiments comparing different randomized low-rank approximation algorithms (subsection 7.1), and we apply the

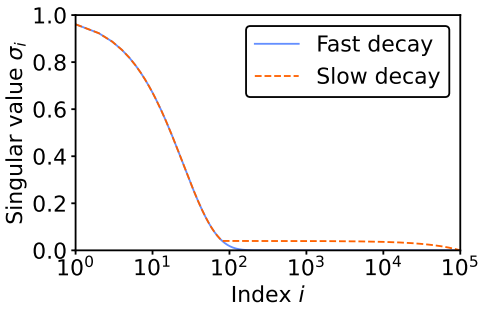

FIG. 4. ***(Fast vs. slow singular value decay).*** *Fast singular value decay of the matrix* $\boldsymbol{A}$ *versus slow singular value decay of the matrix* $\boldsymbol{B}$*, as constructed in subsection 7.1.*

algorithms to scientific problems that require principal component analysis (subsection 7.2) or kernel spectral clustering (subsection 7.3).

7.1. Comparison. As a simple, direct comparison, we apply `RSVD`, `RSI`, `RBKI`, and their Nyström-based variants to approximate two psd matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ with singular values

$$\sigma_i(\boldsymbol{A}) = \mathrm{e}^{-i/25}, \qquad \sigma_i(\boldsymbol{B}) = \max\left\{\mathrm{e}^{-i/25}, \frac{1 - i/10^5}{25}\right\}, \qquad i = 1, 2, \dots, 10^5.$$

Matrix $\boldsymbol{B}$ models a noisy version of $\boldsymbol{A}$, with noise affecting the singular values $\sigma_i(\boldsymbol{B})$ for $i \geq 81$ (see Figure 4). The noise singular values are small, ranging from 0.00 to 0.04 in magnitude, but given the high-dimensionality ($L = N = 10^5$), they threaten to drown out the signal. We construct $\boldsymbol{A}$ and $\boldsymbol{B}$ as diagonal matrices, as the choice of singular vectors does not affect the performance of our algorithms in exact arithmetic (Lemma 8.3).

Figure 5 evaluates the computational cost and approximation error of several low-rank approximation algorithms. The horizontal axis measures the computational cost using the number of matrix–vector products km. We allow the block size k to vary for `RSVD` and `NysSVD`. For the other algorithms, we fix the block size to $k = 100$ and allow the number of matrix–matrix multiplications m to vary. The vertical axis measures the root-mean-square spectral-norm approximation error

$$\left(\mathbb{E}\left\|\hat{\boldsymbol{A}} - \boldsymbol{A}\right\|^2\right)^{1/2} \quad \text{or} \quad \left(\mathbb{E}\left\|\hat{\boldsymbol{B}} - \boldsymbol{B}\right\|^2\right)^{1/2}.$$

The expectation is approximated empirically over 100 independent Gaussian initializations. Note that we are using absolute (not relative) errors here.

The results in Figure 5 show that various randomized low-rank approximation algorithms, including `NysSVD` and `RSVD`, can produce a high-quality approximation of the matrix $\boldsymbol{A}$ with fast singular value decay. Since `NysSVD` and `RSVD` use fewer matrix–matrix multiplications ($m = 1$ for `NysSVD`, $m = 2$ for `RSVD`), these strategies are more cost-efficient when approximating such a matrix. `NysSVD` is more efficient than `RSVD` since it achieves the same approximation accuracy using half as many matrix–matrix multiplications.

In contrast, the Krylov methods yield the highest accuracy approximations for the matrix $\boldsymbol{B}$ with slowly decaying singular values. The `NysBKI` method is the top performer, since it achieves this high approximation accuracy using a factor of $\sqrt{2}$

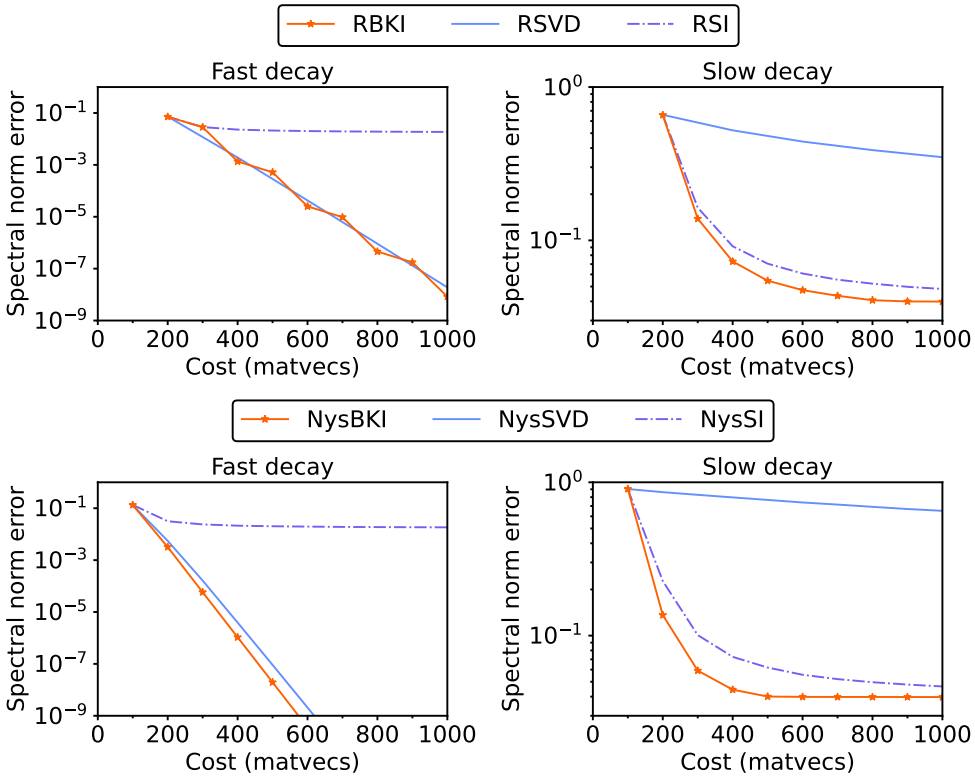

FIG. 5. ***(Matrix approximation comparisons).*** *Matrix approximation error for the matrix* $\boldsymbol{A}$ *with fast singular value decay (left) and the matrix* $\boldsymbol{B}$ *with slow singular value decay (right), as constructed in subsection 7.1. Algorithms for general matrices are on top, algorithms for psd matrices are on bottom.*

fewer matrix–matrix multiplications than `RBKI`.

Next, Figure 6 compares the singular vector approximations from several low-rank approximation algorithms. The singular vectors are vital for principal component analysis and kernel spectral clustering, as we will show through examples in subsections 7.2 and 7.3. We evaluate the error in the estimated singular vectors as

$$\left(\mathbb{E}\left\|\boldsymbol{\Pi}_{75}(\hat{\boldsymbol{A}}) - \boldsymbol{\Pi}_{75}(\boldsymbol{A})\right\|^2\right)^{1/2} \quad \text{or} \quad \left(\mathbb{E}\left\|\boldsymbol{\Pi}_{75}(\hat{\boldsymbol{B}}) - \boldsymbol{\Pi}_{75}(\boldsymbol{B})\right\|^2\right)^{1/2},$$

where $\boldsymbol{\Pi}_{75}(\cdot)$ denotes the orthogonal projection onto the dominant 75 right singular vectors. We find that `RSVD` and `NysSVD` lead to accurate singular vector approximations for the matrix $\boldsymbol{A}$ with rapid singular value decay. However, `RBKI` and `NysBKI` produce **10× to 300× more accurate singular vector approximations** for the matrix $\boldsymbol{B}$ than the other low-rank approximation methods.

7.2. Human genetic diversity data. Principal component analysis is frequently applied to large matrices containing single-nucleotide polymorphism (SNP) data, encoding the variations in DNA at specific locations in the genome [1, 41]. The principal components of the SNP data can be used to cluster individuals that share similar genomes.

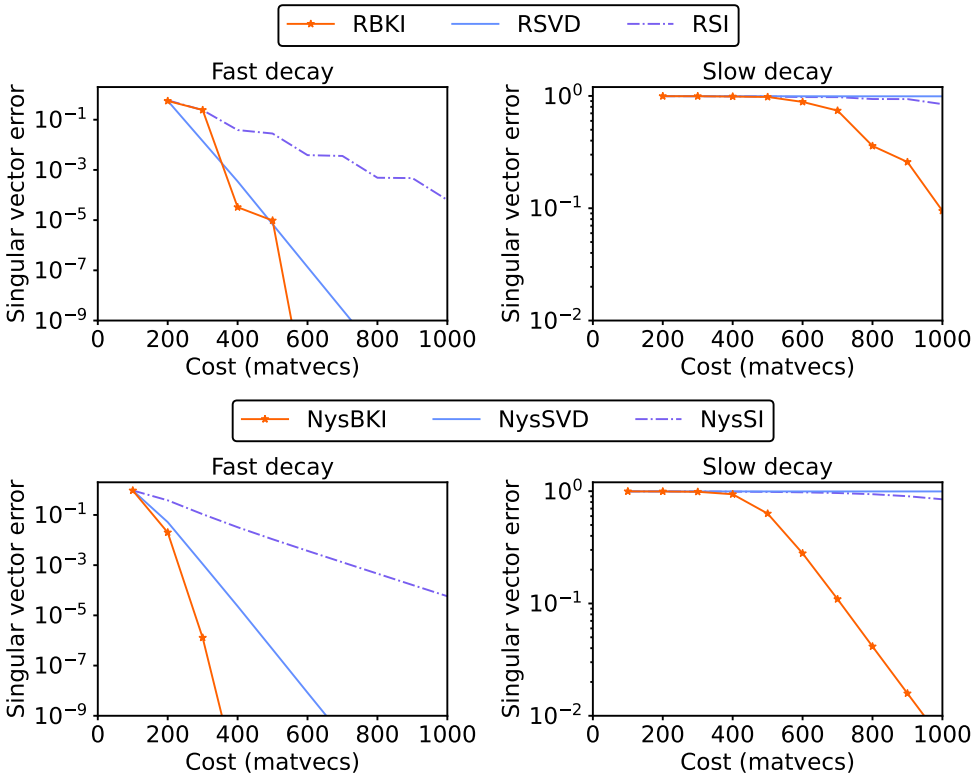

FIG. 6. ***(Singular vector approximation comparisons).*** *Error in the dominant 75 singular vectors for the matrix* $\boldsymbol{A}$ *with fast singular value decay (left) and the matrix* $\boldsymbol{B}$ *with slow singular value decay (right), as constructed in subsection 7.1. Algorithms for general matrices are on top, algorithms for psd matrices are on bottom.*

Here, we apply principal component analysis to the HapMap3 data set [57], one of the earliest data sets measuring human genetic diversity, which we downloaded from [2]. The data is organized into a matrix $\boldsymbol{A} \in \mathbb{R}^{957\times 14{,}079}$ containing counts of SNPs for 957 individuals in 14,079 chromosomal locations. The entries of $\boldsymbol{A}$ are 0, 1, or 2, corresponding to the number of affected chromosomes. To normalize the matrix entries, we apply the transformation

$$\boldsymbol{B}_{ij} = \frac{\boldsymbol{A}_{ij} - \mu_j}{\sqrt{\frac{1}{2}\mu_j(1-\frac{1}{2}\mu_j)}}, \quad \text{where} \quad \mu_j = \frac{1}{957}\sum_{i=1}^{957} \boldsymbol{A}_{ij}.$$

Our goal is to extract the principal components, which are defined as the right singular vectors of $\boldsymbol{B}$. Once we have identified the principal components, we can cluster individuals by genetic ancestry as depicted in Figure 7.

HapMap3 is a small data set by today's standards, and we can obtain the principal components using a full singular value decomposition on a laptop. Yet modern genetic data sets may contain millions of individuals and genetic markers, making a full singular value decomposition infeasible [17]. Therefore, we consider a more scalable approach to principal component analysis based on randomized low-rank ap-

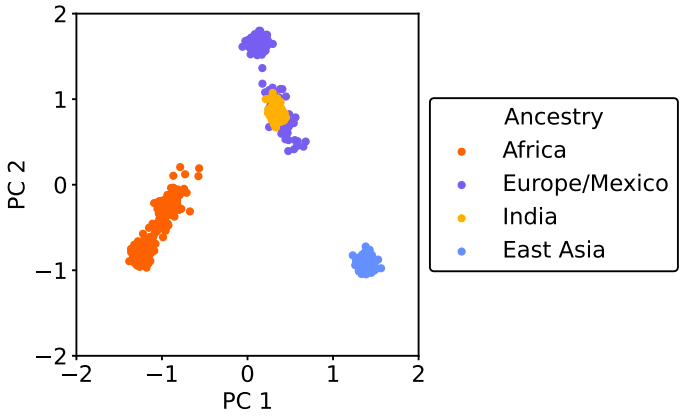

FIG. 7. ***(Clustering in genetics).*** *k-means clustering applied to the top 5 principal components of the HapMap3 data set, as described in subsection 7.2. Figure shows the projection onto the top 2 principal components.*

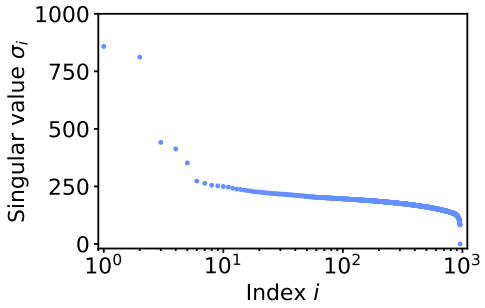

FIG. 8. ***(Singular value decay in genetics).*** *Singular value decay for the HapMap3 data set, as described in subsection 7.2.*

proximation.

The randomized approach to principal component analysis has been questioned in the past. In 2013, Chen et al. wrote [23],

> "In theory, randomized eigenvector approximations [RSVD] can reduce the running time... (Rokhlin et al., 2009). However, a colleague of ours reports that efforts to apply this approach to genetic data have not yet been successful, as in large datasets, eigenvalues may be highly significant (reflecting real population structure in the data) but only slightly larger than background noise eigenvalues, and thus sometimes missed by randomized methods (N. Patterson, personal communication)."

The authors are concerned that randomized algorithms cannot accurately identify principal components when the singular values are close together. This would present a major limitation, since genetic data sets often have small singular value gaps, as shown for the HapMap3 data in Figure 8.

Figure 9 compares the accuracy of RSVD, RSI, and RBKI (the latter two with block size $k = 10$) when approximating the principal components of the HapMap3

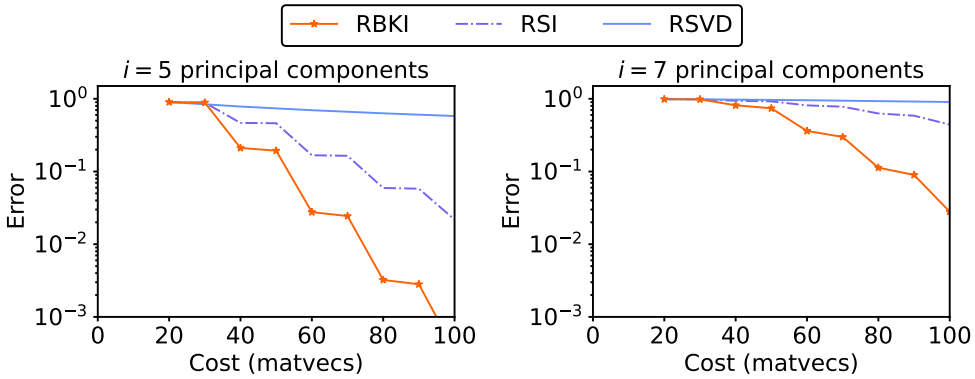

FIG. 9. ***(Approximating principal components in genetics).*** *Error in the top $i = 5$ principal components (left) or top $i = 7$ principal components (right) for the HapMap3 data, as described in subsection 7.2.*

data. The vertical axis evaluates error in the principal components using

$$\left(\mathbb{E}\left\|\boldsymbol{\Pi}_i(\hat{\boldsymbol{B}}) - \boldsymbol{\Pi}_i(\boldsymbol{B})\right\|^2\right)^{1/2},$$

where $\boldsymbol{\Pi}_i(\cdot)$ represents the orthogonal projection onto the top $i = 5$ or $i = 7$ right singular vectors and the expectation is measured empirically over 100 independent Gaussian initializations. The top $i = 5$ principal components (left column) are separated by a sizable spectral gap, whereas the top $i = 7$ principal components (right column) are separated by a tiny singular value gap leading to larger errors. The results in Figure 9 confirm that `RSVD` and `RSI` struggle to identify the top $i = 7$ principal components in the data. Only `RBKI` identifies all 7 principal components up to a precision of $\varepsilon = 0.1$.

`RBKI` also performs well at separating individuals into clusters. With just $km = 40$ matrix–vector products, the algorithm identifies the top $i = 5$ principal components well enough to match the ideal clustering results in Figure 7. In contrast, we would need to perform `RSVD` with **20× as many matvecs** ($km = 800$) to achieve the same clustering accuracy.

These investigations show that `RBKI` is both fast and accurate when performing principal component analysis with genetic data. `RBKI` is faster than traditional singular value decomposition, since we can take km to be a small fraction of the number of rows. Moreover `RBKI` is highly accurate, even when the signal barely rises above the noise.

7.3. Spectral clustering. Kernel spectral clustering is a powerful approach for analyzing the dynamics of proteins using data from molecular dynamics simulations [43]. Spectral clustering is useful for identifying metastable states that the protein occupies for a long time, with rare transitions between states (e.g., folded and unfolded states). However in the past, kernel spectral clustering has mainly been limited to data sets with $N \leq 10^4$ data points, because it requires calculating the dominant eigenvectors of an $N \times N$ matrix.

Here, we use `NysBKI` to extend kernel spectral clustering to a large data set with $N = 250{,}000$ points. The data comes from a 250 ns simulation of alanine dipeptide ($CH_3–CO–NH–C_\alpha HCH_3–CO–NH–CH_3$), which we downloaded from [77]. The

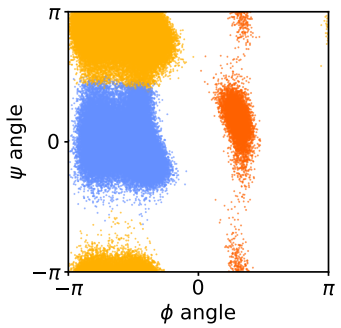

FIG. 10. ***(Clustering in biochemistry).*** *Kernel spectral clustering with $k = 3$ clusters, $r = 3$ eigenvectors, and a bandwidth $\sigma = .05$nm, applied to the alanine dipeptide data set ($N = 250{,}000$, $d = 30$) as described in subsection 7.3. Figure shows the projection onto the ϕ and ψ dihedral angles.*

data points $\boldsymbol{x}^{(i)} \in \mathbb{R}^{30}$ identify the spatial (x, y, z)-positions of the 10 non-hydrogen atoms at 1 ps intervals. We will perform kernel spectral clustering as follows.

1. Form the psd Gaussian kernel matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ with entries
$$a_{ij} = \exp\left(-\frac{1}{2\sigma^2}\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\|^2\right),$$
where $\sigma > 0$ is a tunable bandwidth.
2. Form the diagonal matrix $\boldsymbol{D} \in \mathbb{R}^{N \times N}$ containing the row sums of $\boldsymbol{A}$.
3. Calculate the dominant r eigenvectors $\boldsymbol{V} = \begin{bmatrix}\boldsymbol{v}^{(1)} & \cdots & \boldsymbol{v}^{(r)}\end{bmatrix}$ for the psd matrix $\boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$, where r is a tunable parameter.
4. Apply the diagonal rescaling $\boldsymbol{V} \leftarrow \boldsymbol{D}^{-1/2}\boldsymbol{V}$, so that $\boldsymbol{V}$ contains right eigenvectors of $\boldsymbol{D}^{-1}\boldsymbol{A}$.
5. Apply k-means clustering to the rows of $\boldsymbol{V}$.

The spectral clustering algorithm identifies three clusters in the 30-dimensional alanine dipeptide data space. These clusters are highly correlated with the dihedral angle ϕ between C, N, C_α, and C and ψ between N, C_α, C, and N, as shown in Figure 10. Yet we emphasize that the algorithm does not have access to the dihedral angles: it discovers the clusters organically, reproducing the past work [78].

It is computationally challenging to complete the third step of kernel spectral clustering, which requires calculating the dominant eigenvectors of an $N \times N$ psd matrix. For the alanine dipeptide problem, the matrix $\boldsymbol{B} = \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$ requires 222GB storage and is therefore too large to fit in working memory on a 64GB laptop. Therefore, to produce the results in Figure 10, we store the matrix on disk and use `NysBKI` to approximate the leading eigenvectors. With parameter settings $k = 100$ and $m = 2$, `NysBKI` runs in just 11 minutes and 99.9% of the time is spent loading blocks of the matrix and performing blockwise matrix multiplications.

To provide additional insight into the difficulties of kernel spectral clustering, we take a subset of $N = 25{,}000$ equally spaced data points (so we can perform a full eigendecomposition as a reference) and consider the impact of varying the bandwidth σ. A small bandwidth σ is needed to obtain physically relevant clusters with sharp boundaries between clusters. Unfortunately, however, the eigenvalues decay more slowly when σ is small (Figure 11), leading to a difficult approximation problem.

Figure 12 presents the eigenvector approximation error when `NysSVD`, `NysSI`, and

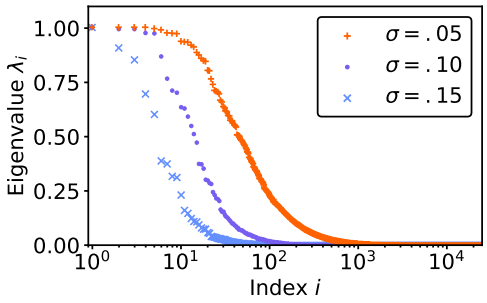

FIG. 11. ***(Eigenvalue decay in biochemistry).*** *Eigenvalues of the kernel matrix for a subset of $N = 25{,}000$ alanine dipeptide data points, as described in subsection 7.3. The kernel matrix is evaluated with bandwidth parameter $\sigma = .05$, .10, or .15nm.*

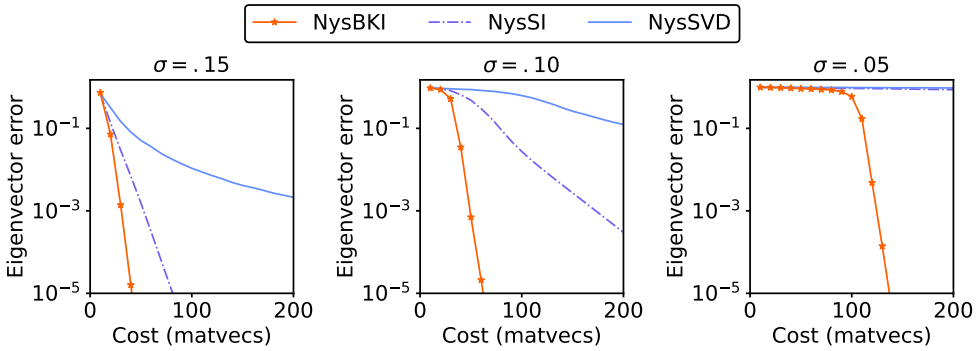

FIG. 12. ***(Approximating eigenvectors in biochemistry).*** *Error in the top 3 eigenvectors of the kernel matrix for a subset of $N = 25{,}000$ alanine dipeptide data points, as described in subsection 7.3. The kernel matrix is evaluated with bandwidth parameter $\sigma = .15$ (left), $\sigma = .10$ (center) or $\sigma = .05$nm (right).*

`NysBKI` (the latter two with block size $k = 10$) are applied to the subset of $N = 25{,}000$ alanine dipeptide data points. We measure the eigenvector approximation error using

$$\left(\mathbb{E}\left\|\boldsymbol{\Pi}_3(\hat{\boldsymbol{B}}) - \boldsymbol{\Pi}_3(\boldsymbol{B})\right\|^2\right)^{1/2}$$

where $\boldsymbol{\Pi}_3$ is the orthogonal projection onto the dominant 3 eigenvectors, and the expectation is evaluated empirically over 100 independent runs. The results show that a relatively small number of matvecs ($km < 100$) are sufficient to approximate the top 3 eigenvectors when $\sigma = .15$ (left) and even when $\sigma = .10$ (center). However, a larger number of matvecs is needed when $\sigma = .05$ (right), which is the physically relevant parameter setting for alanine dipeptide. The accuracy is dramatically higher when using `NysBKI`, instead of `RSVD` or `RSI`.

In summary, we find that kernel spectral clustering leads to challenging, high-dimensional eigenvalue problems, especially when the bandwidth σ is small. To solve these eigenvalue problems with high accuracy and scalability, we recommend the `NysBKI` algorithm. `NysBKI` works robustly across different bandwidth settings, and it leads to dramatic computational speedups. With $N = 250{,}000$ data points and an approximation rank of $km = 200$, the difference between the traditional $\mathcal{O}(N^3)$

operation count and NysBKI's $\mathcal{O}(kmN^2)$ operation count is **over 3 orders of magnitude**.

8. Analysis of RSVD and NysSVD. In this section, we derive error bounds for RSVD (Algorithm 5.1) and NysSVD (Algorithm 5.6), which are the simplest randomized low-rank approximation algorithms. These results provide the foundation for studying the more sophisticated methods (RSI, RBKI, NysSI, NysBKI). Although related results are already well-established [54], we have found opportunities for simplification and improvement.

We compare the RSVD approximation to an optimal rank-r approximation $\lfloor \boldsymbol{A} \rfloor_r$, which arises from an r-truncated singular value decomposition, and we establish the following main result. The proof appears below in subsection 8.3.

THEOREM 8.1 (RSVD error). *Fix a matrix $\boldsymbol{A} \in \mathbb{R}^{L\times N}$ and a target rank $r \geq 1$. For any $2 \leq p \leq \infty$ and $u, t \geq 0$, RSVD with $k \geq r$ Gaussian initialization vectors generates a random rank-k approximation $\hat{\boldsymbol{A}} \in \mathbb{R}^{L\times N}$ that satisfies*

$$\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_p^2 \leq \|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_r\|_p^2 + \frac{utr}{k-r+1}\|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_r\|_{\mathrm{F}}^2 \tag{RSVD1}$$

with failure probability at most $\mathrm{e}^{-(u-2)/4} + \sqrt{\pi r}(t/\mathrm{e})^{-(k-r+1)/2}$.

Additionally, if the block size k is no smaller than $r+2$, then

$$\mathbb{E}\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_p^2 \leq \|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_r\|_p^2 + \frac{r}{k-r-1}\|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_r\|_{\mathrm{F}}^2. \tag{RSVD2}$$

The result for NysSVD is a corollary of the result for RSVD.

COROLLARY 8.2 (NysSVD error). *Fix a psd matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ and a target rank $r \geq 1$. For any $1 \leq p \leq \infty$ and $u, t \geq 0$, NysSVD with $k \geq r$ Gaussian initialization vectors generates a random rank-k approximation $\hat{\boldsymbol{A}} \in \mathbb{R}^{N\times N}$ that satisfies*

$$\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_p \leq \|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_r\|_p + \frac{utr}{k-r+1}\|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_r\|_* \tag{NysSVD1}$$

with failure probability at most $\mathrm{e}^{-(u-2)/4} + \sqrt{\pi r}(t/\mathrm{e})^{-(k-r+1)/2}$.

Additionally, if the block size k is no smaller than $r+2$, then

$$\mathbb{E}\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_p \leq \|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_r\|_p + \frac{r}{k-r-1}\|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_r\|_*. \tag{NysSVD2}$$

The error bounds in Theorem 8.1 and Corollary 8.2 only depend on the singular values $\sigma_i(\boldsymbol{A})$ for $i \geq r+1$. If these singular values are small and rapidly decaying, the theorem immediately establishes the accuracy of the RSVD and NysSVD approximations.

Interpreting these error bounds, we identify two limitations of RSVD and NysSVD. First, when the singular values decay slowly, the terms $\|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_r\|_{\mathrm{F}}$ and $\|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_r\|_*$ are large, in which case RSVD and NysSVD perform poorly, as confirmed by our experiments in section 7. To target matrices with slow singular value decay, we need more powerful algorithms such as RBKI and NysBKI. As another limitation, the expectation bounds are not applicable when $k = r$ or $k = r+1$, and large random errors can occur in these settings. When $k-r$ is small, there is a fundamental problem that the range of k random Gaussian initialization vectors might barely align with the leading r singular vectors.

RSVD and NysSVD were previously analyzed in [54, 50, 100]. However, by revisiting and reworking the details of these analyses, we obtain a more flexible framework

that delivers many new bounds. The probability bounds (RSVD1) and (NysSVD1) are especially novel and informative, since previous analyses [50, 99] separated the cases $k = r$, $k = r+1$, and $k \geq r+2$ whereas we provide a unified treatment. Additionally, our analytic approach leads to a broad range of expectation bounds for RSVD and NysSVD, many of them new, to be presented in subsection 8.4.

The new technical idea in this section is to decompose the Schatten p-norm error of RSVD using *parallel sums* of psd matrices [4]. Parallel sums, which generalize the harmonic mean to the case of psd matrices, are sufficiently tractable that they allow us to produce a wide range of probability and expectation bounds. To our knowledge, parallel sums have not been previously exploited in the randomized matrix approximation literature.

The rest of this section contains a complete and up-to-date analysis of RSVD and NysSVD. Subsection 8.1 presents a reduction to the case of diagonal, psd matrices; subsection 8.2 analyzes the error of RSVD using parallel sums; subsection 8.3 uses random matrix theory to prove our main error bounds; and subsection 8.4 derives additional expectation bounds for RSVD and NysSVD. Afterward, section 9 deploys these results to describe the behavior of RSI, NysSI, RBKI, and NysBKI.

8.1. Diagonal, psd reduction. As the first step in our analysis, we show that the error of many randomized low-rank approximation algorithms depends only on the singular values of the target matrix, regardless of the singular vectors. This reduction is standard.

LEMMA 8.3 (Diagonal, psd reduction). *When we apply RSVD, RSI, RBKI, NysSVD, NysSI, or NysBKI to approximate a matrix $\boldsymbol{A} \in \mathbb{R}^{L \times N}$ using a Gaussian initialization matrix $\boldsymbol{\Omega} \in \mathbb{R}^{N \times k}$, the Schatten p-norm error $\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_p$ has a distribution that only depends on the singular values of $\boldsymbol{A}$, regardless of the singular vectors, for any $1 \leq p \leq \infty$.*

Proof. Let $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$ be a singular value decomposition for $\boldsymbol{A}$ and set $\boldsymbol{\Omega}' = \boldsymbol{V}^*\boldsymbol{\Omega}$. The randomized low-rank approximation $\hat{\boldsymbol{A}}$ can be written:

(RSVD)	$\boldsymbol{U}\big(\boldsymbol{\Pi}_{\boldsymbol{\Sigma}\boldsymbol{\Omega}'}\boldsymbol{\Sigma}\big)\boldsymbol{V}^*,$
(RSI, even m)	$\boldsymbol{U}\big(\boldsymbol{\Pi}_{\boldsymbol{\Sigma}^{m-1}\boldsymbol{\Omega}'}\boldsymbol{\Sigma}\big)\boldsymbol{V}^*,$
(RSI, odd m)	$\boldsymbol{U}\big(\boldsymbol{\Sigma}\boldsymbol{\Pi}_{\boldsymbol{\Sigma}^{m-1}\boldsymbol{\Omega}'}\big)\boldsymbol{V}^*,$
(RBKI, even m)	$\boldsymbol{U}\big(\boldsymbol{\Pi}_{[\,\boldsymbol{\Sigma}^{m-1}\boldsymbol{\Omega}'\ \boldsymbol{\Sigma}^{m-3}\boldsymbol{\Omega}'\ \cdots\ \boldsymbol{\Sigma}\boldsymbol{\Omega}'\,]}\boldsymbol{\Sigma}\big)\boldsymbol{V}^*,$
(RBKI, odd m)	$\boldsymbol{U}\big(\boldsymbol{\Sigma}\boldsymbol{\Pi}_{[\,\boldsymbol{\Sigma}^{m-1}\boldsymbol{\Omega}'\ \boldsymbol{\Sigma}^{m-3}\boldsymbol{\Omega}'\ \cdots\ \boldsymbol{\Sigma}^{2}\boldsymbol{\Omega}'\,]}\big)\boldsymbol{V}^*,$

The Schatten p-norm is unitarily invariant, so the p-norm error only depends on $\boldsymbol{\Sigma}$ and the matrix $\boldsymbol{\Omega}' = \boldsymbol{V}^*\boldsymbol{\Omega}$, which is a random Gaussian matrix.

Similarly, if $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ is psd, let $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^*$ be an eigenvalue decomposition for $\boldsymbol{A}$ and set $\boldsymbol{\Omega}' = \boldsymbol{U}^*\boldsymbol{\Omega}$. The randomized low-rank approximation $\hat{\boldsymbol{A}}$ can be written:

(NysSVD)	$\boldsymbol{U}\big(\boldsymbol{\Sigma}^{1/2}\boldsymbol{\Pi}_{\boldsymbol{\Sigma}^{1/2}\boldsymbol{\Omega}'}\boldsymbol{\Sigma}^{1/2}\big)\boldsymbol{U}^*,$
(NysSI)	$\boldsymbol{U}\big(\boldsymbol{\Sigma}^{1/2}\boldsymbol{\Pi}_{\boldsymbol{\Sigma}^{m-1/2}\boldsymbol{\Omega}'}\boldsymbol{\Sigma}^{1/2}\big)\boldsymbol{U}^*,$
(NysBKI)	$\boldsymbol{U}\big(\boldsymbol{\Sigma}^{1/2}\boldsymbol{\Pi}_{[\boldsymbol{\Sigma}^{m-1/2}\boldsymbol{\Omega}'\ \boldsymbol{\Sigma}^{m-3/2}\boldsymbol{\Omega}'\ \cdots\ \boldsymbol{\Sigma}^{1/2}\boldsymbol{\Omega}']}\boldsymbol{\Sigma}^{1/2}\big)\boldsymbol{U}^*.$

The same conclusion follows as before: the error only depends on $\boldsymbol{\Sigma}$ and the Gaussian matrix $\boldsymbol{\Omega}' = \boldsymbol{U}^*\boldsymbol{\Omega}$. ∎

As a consequence of Lemma 8.3 and without loss of generality, the subsequent analysis will assume that we are approximating a matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ that is diagonal and psd, with nonincreasing diagonal entries.

8.2. Deterministic analysis using parallel sums. In this section, we develop a simple formula for the error of RSVD when approximating a diagonal, psd matrix. We fix the target matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$, fix the initialization matrix $\boldsymbol{\Omega} \in \mathbb{R}^{N \times k}$, and analyze the p-norm error $\|\boldsymbol{A} - \boldsymbol{\Pi}_{\boldsymbol{A}\boldsymbol{\Omega}}\boldsymbol{A}\|_p$. For this section, it does not matter whether the test matrix $\boldsymbol{\Omega}$ is deterministic or random.

Our analysis is based on *parallel sums* of psd matrices. Parallel sums were introduced by Anderson & Duffin [4] to give a mathematical framework for analyzing electric networks of capacitors and resistors. Parallel sums have become a favorite topic in linear algebra textbooks (e.g., [13, Ch. 4]), and they satisfy the following simple and natural properties:

LEMMA 8.4 (Parallel sums). *For psd matrices $\boldsymbol{A}, \boldsymbol{B}$ with the same dimension, define the* parallel sum $\boldsymbol{A} : \boldsymbol{B}$ *to be the psd matrix*

$$\boldsymbol{A} : \boldsymbol{B} = \boldsymbol{A} - \boldsymbol{A}(\boldsymbol{A} + \boldsymbol{B})^{\dagger}\boldsymbol{A}.$$

Then the following properties hold:

1. *The parallel sum is symmetric; that is, $\boldsymbol{A} : \boldsymbol{B} = \boldsymbol{B} : \boldsymbol{A}$.*
2. *If $\boldsymbol{A}$ and $\boldsymbol{B}$ are strictly positive definite, then $\boldsymbol{A} : \boldsymbol{B} = (\boldsymbol{A}^{-1} + \boldsymbol{B}^{-1})^{-1}$.*
3. *The mapping $\boldsymbol{B} \mapsto \boldsymbol{A} : \boldsymbol{B}$ is monotone and concave with respect to the psd ordering, and it is bounded above as $\boldsymbol{A} : \boldsymbol{B} \preccurlyeq \boldsymbol{A}$.*
4. *$\|\boldsymbol{A} : \boldsymbol{B}\|_p \le \|\boldsymbol{A}\|_p : \|\boldsymbol{B}\|_p$ for any Schatten p-norm with $1 \le p \le \infty$.*

Proof. Anderson & Duffin [4] established properties (1)–(3) and property (4) in the special cases $p = 1$ and $p = \infty$. Ando extended property (4) to the general case $1 \le p \le \infty$; see [5, eq. 3.13]. ∎

Next, we use parallel sums to analyze the RSVD error. Our error decomposition is similar to [54, Theorem 9.1], but parallel sums allow us to highlight the main reasoning and shorten the proof.

PROPOSITION 8.5 (Error decomposition). *Fix a diagonal, psd matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ and a test matrix $\boldsymbol{\Omega} \in \mathbb{R}^{N \times k}$. Partition the matrices as*

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_1 & \\ & \boldsymbol{A}_2 \end{bmatrix}, \qquad \boldsymbol{\Omega} = \begin{bmatrix} \boldsymbol{\Omega}_1 \\ \boldsymbol{\Omega}_2 \end{bmatrix},$$

so that $\boldsymbol{A}_1$ is a $r \times r$ matrix and $\boldsymbol{\Omega}_1$ is a $r \times k$ matrix with $r \le k$. Assume that $\operatorname{rank}(\boldsymbol{\Omega}_1) = r$. *Then the orthogonal projection $\boldsymbol{Q} = \boldsymbol{\Pi}_{\boldsymbol{A}\boldsymbol{\Omega}\boldsymbol{\Omega}_1^{\dagger}}$ satisfies*

$$\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{A}\|_p^2 \le \|\boldsymbol{A}_2\|_p^2 + \|\boldsymbol{A}_1\|_p^2 : \|\boldsymbol{A}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^{\dagger}\|_p^2 \tag{8.1}$$

$$\le \|\boldsymbol{A}_2\|_p^2 + \|\boldsymbol{A}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^{\dagger}\|_p^2. \tag{8.2}$$

for any Schatten p-norm with $2 \le p \le \infty$.

Proof. We make the following calculation:

$$\|(\mathbf{I}-\boldsymbol{Q})\boldsymbol{A}\|_p^2 = \|(\mathbf{I}-\boldsymbol{Q})\boldsymbol{A}^2(\mathbf{I}-\boldsymbol{Q})\|_{p/2} \tag{8.3}$$

$$\leq \left\|(\mathbf{I}-\boldsymbol{Q})\begin{bmatrix}\mathbf{0} & \\ & \boldsymbol{A}_2^2\end{bmatrix}(\mathbf{I}-\boldsymbol{Q})\right\|_{p/2} + \left\|(\mathbf{I}-\boldsymbol{Q})\begin{bmatrix}\boldsymbol{A}_1^2 & \\ & \mathbf{0}\end{bmatrix}(\mathbf{I}-\boldsymbol{Q})\right\|_{p/2} \tag{8.4}$$

$$\leq \left\|\begin{bmatrix}\mathbf{0} & \\ & \boldsymbol{A}_2\end{bmatrix}\right\|_p^2 + \left\|(\mathbf{I}-\boldsymbol{Q})\begin{bmatrix}\boldsymbol{A}_1 & \\ & \mathbf{0}\end{bmatrix}\right\|_p^2 \tag{8.5}$$

$$= \|\boldsymbol{A}_2\|_p^2 + \left\|\begin{bmatrix}\boldsymbol{A}_1 & \\ & \mathbf{0}\end{bmatrix}(\mathbf{I}-\boldsymbol{Q})\begin{bmatrix}\boldsymbol{A}_1 & \\ & \mathbf{0}\end{bmatrix}\right\|_{p/2}. \tag{8.6}$$

Equations (8.3), (8.5), and (8.6) use identities $\|\boldsymbol{M}\|_p^2 = \|\boldsymbol{M}^*\boldsymbol{M}\|_{p/2} = \|\boldsymbol{M}\boldsymbol{M}^*\|_{p/2}$, while (8.4) follows from the triangle inequality, and (8.5) relies on the property that the orthogonal projection $\mathbf{I}-\boldsymbol{Q}$ is a p-norm contraction.

Next, the orthogonal projection $\boldsymbol{Q}$ is given explicitly by the formula

$$\boldsymbol{Q} = (\boldsymbol{A}\boldsymbol{\Omega}\boldsymbol{\Omega}_1^\dagger)\big[(\boldsymbol{A}\boldsymbol{\Omega}\boldsymbol{\Omega}_1^\dagger)^*(\boldsymbol{A}\boldsymbol{\Omega}\boldsymbol{\Omega}_1^\dagger)\big]^\dagger(\boldsymbol{A}\boldsymbol{\Omega}\boldsymbol{\Omega}_1^\dagger)^*. \tag{8.7}$$

By the assumption that $\operatorname{rank}[\boldsymbol{\Omega}_1] = r$, we have $\boldsymbol{\Omega}_1\boldsymbol{\Omega}_1^\dagger = \mathbf{I}$. Consequently,

$$\boldsymbol{A}\boldsymbol{\Omega}\boldsymbol{\Omega}_1^\dagger = \begin{bmatrix}\boldsymbol{A}_1 \\ \boldsymbol{A}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\end{bmatrix} =: \begin{bmatrix}\boldsymbol{A}_1 \\ \boldsymbol{F}\end{bmatrix}. \tag{8.8}$$

We have introduced the abbreviation $\boldsymbol{F} = \boldsymbol{A}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger$. A direct calculation involving (8.7) and (8.8) delivers an expression involving the parallel sum:

$$\begin{bmatrix}\boldsymbol{A}_1 & \\ & \mathbf{0}\end{bmatrix}(\mathbf{I}-\boldsymbol{Q})\begin{bmatrix}\boldsymbol{A}_1 & \\ & \mathbf{0}\end{bmatrix} = \begin{bmatrix}\boldsymbol{A}_1^2 : (\boldsymbol{F}^*\boldsymbol{F}) & \\ & \mathbf{0}\end{bmatrix}.$$

By Lemma 8.4 parts (3) and (4), we obtain the bound

$$\big\|\boldsymbol{A}_1^2 : (\boldsymbol{F}^*\boldsymbol{F})\big\|_{p/2} \leq \big\|\boldsymbol{A}_1^2\big\|_{p/2} : \big\|\boldsymbol{F}^*\boldsymbol{F}\big\|_{p/2} = \|\boldsymbol{A}_1\|_p^2 : \|\boldsymbol{F}\|_p^2 \leq \|\boldsymbol{F}\|_p^2.$$

This statement leads to (8.2) and completes the proof. ∎

The error bounds in Proposition 8.5 ensure that RSVD gives a high-quality approximation when the quantity $\|\boldsymbol{A}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\|_p^2$ is small. To understand when this quantity is likely to be small, we need additional tools from random matrix theory, to be presented in the next section.

8.3. Calculations using random matrix theory. To complete the proof of our main result, we apply the following error bounds for Gaussian matrices, which are derived in Appendix B.1.

PROPOSITION 8.6 (Gaussian matrix products). *Fix a matrix $\boldsymbol{S} \in \mathbb{R}^{(N-r)\times(N-r)}$ and consider independent Gaussian matrices $\boldsymbol{G} \in \mathbb{R}^{(N-r)\times k}$ and $\boldsymbol{H} \in \mathbb{R}^{r\times k}$, where $k \geq r$. Then, for any $u, t \geq 0$,*

$$\mathbb{P}\left\{\|\boldsymbol{S}\boldsymbol{G}\boldsymbol{H}^\dagger\|_{\mathrm{F}}^2 > \frac{utr}{k-r+1}\|\boldsymbol{S}\|_{\mathrm{F}}^2\right\} \leq \mathrm{e}^{-(u-2)/4} + \sqrt{\pi r}\left(\frac{t}{\mathrm{e}}\right)^{-(k-r+1)/2}. \tag{8.9}$$

Additionally, if $k \geq r+2$,

$$\mathbb{E}\|\boldsymbol{S}\boldsymbol{G}\boldsymbol{H}^\dagger\|_{\mathrm{F}}^2 = \frac{r}{k-r-1}\|\boldsymbol{S}\|_{\mathrm{F}}^2. \tag{8.10}$$

The expectation formula (8.10) was previously presented in [54, Thm. 10.5]. It follows from the expectation formula for an inverse Wishart matrix [86, pg. 119]. The probability bound (8.9) improves on previous results [54, Thm. 10.7]. It is derived using a Lipschitz concentration inequality for Gaussian random fields, together with a new concentration inequality for the trace of an inverse Wishart matrix.

The random matrix theory allow us to complete the proof of our main results, Theorem 8.1 and Corollary 8.2, as follows.

Proof of Theorem 8.1 and Corollary 8.2. Without loss of generality, assume that $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ is diagonal and psd, with nonincreasing diagonal entries. We partition the Gaussian test matrix $\boldsymbol{\Omega} = \left[\begin{smallmatrix}\boldsymbol{\Omega}_1\\ \boldsymbol{\Omega}_2\end{smallmatrix}\right]$, where $\boldsymbol{\Omega}_1$ is a $r \times k$ matrix and $\boldsymbol{\Omega}_2$ is a $(N-r)\times k$ matrix. The two submatrices $\boldsymbol{\Omega}_1$ and $\boldsymbol{\Omega}_2$ are independent Gaussian matrices, and $\boldsymbol{\Omega}_1$ almost surely has rank r.

`RSVD` produces an approximation $\hat{\boldsymbol{A}} \in \mathbb{R}^{N\times n}$ that satisfies

$$\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_p^2 = \|(\mathbf{I} - \boldsymbol{\Pi}_{\boldsymbol{A\Omega}})\boldsymbol{A}\|_p^2 \le \|(\mathbf{I} - \boldsymbol{\Pi}_{\boldsymbol{A\Omega\Omega}_1^\dagger})\boldsymbol{A}\|_p^2$$

because $\boldsymbol{A\Omega}$ has a bigger range than $\boldsymbol{A\Omega\Omega}_1^\dagger$; see Lemma 5.1. Next, Proposition 8.5 shows that, almost surely,

$$\|(\mathbf{I} - \boldsymbol{\Pi}_{\boldsymbol{A\Omega\Omega}_1^\dagger})\boldsymbol{A}\|_p^2 \le \|\boldsymbol{A}_2\|_p^2 + \|\boldsymbol{A}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\|_p^2 \le \|\boldsymbol{A}_2\|_p^2 + \|\boldsymbol{A}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\|_{\mathrm{F}}^2.$$

Since $\boldsymbol{\Omega}_1$ and $\boldsymbol{\Omega}_2$ are independent Gaussian matrices, the error bounds from Proposition 8.6 deliver the `RSVD` error bounds (RSVD1) and (RSVD2).

As for the corollary, `NysSVD` produces an approximation $\hat{\boldsymbol{A}} \in \mathbb{R}^{N\times N}$ that satisfies

$$\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_p = \|\boldsymbol{A}^{1/2}(\mathbf{I} - \boldsymbol{\Pi}_{\boldsymbol{A}^{1/2}\boldsymbol{\Omega}})\boldsymbol{A}^{1/2}\|_p = \|(\mathbf{I} - \boldsymbol{\Pi}_{\boldsymbol{A}^{1/2}\boldsymbol{\Omega}})\boldsymbol{A}^{1/2}\|_{2p}^2.$$

We recognize $\|(\mathbf{I} - \boldsymbol{\Pi}_{\boldsymbol{A}^{1/2}\boldsymbol{\Omega}})\boldsymbol{A}^{1/2}\|_{2p}^2$ as the square error of the `RSVD`, applied to the matrix $\boldsymbol{A}^{1/2}$. By invoking the `RSVD` error bounds for $\|(\mathbf{I}-\boldsymbol{\Pi}_{\boldsymbol{A}^{1/2}\boldsymbol{\Omega}})\boldsymbol{A}^{1/2}\|_{2p}^2$, we confirm the `NysSVD` error bounds (NysSVD1) and (NysSVD2). ∎

8.4. Additional expectation bounds. The analysis of `RSVD` using parallel sums of psd matrices leads to a family of expectation bounds with respect to various Schatten p-norms. We give a collection of these bounds in Theorem 8.7, with the proof appearing in Appendix B.2. The bounds underscore the potential for high errors when $k - r$ is small.

THEOREM 8.7 (`RSVD` expectation bounds). *Fix a matrix $\boldsymbol{A} \in \mathbb{R}^{L\times N}$, a target rank $r \ge 1$, and a block size $k \ge r$. Then `RSVD` with a block of k Gaussian starting vectors generates a random approximation $\hat{\boldsymbol{A}} \in \mathbb{R}^{L\times N}$ that satisfies*

$$\frac{\mathbb{E}\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_{\mathrm{F}}^2}{\|\boldsymbol{A} - \lfloor\boldsymbol{A}\rfloor_r\|_{\mathrm{F}}^2} \le \begin{cases} 1 + \dfrac{r}{k-r-1}, & k \ge r+2; \\ 1 + r\log\left(1 + \dfrac{\|\lfloor\boldsymbol{A}\rfloor_r\|_{\mathrm{F}}^2}{\|\boldsymbol{A} - \lfloor\boldsymbol{A}\rfloor_r\|_{\mathrm{F}}^2}\right), & k = r+1; \\ 1 + r\left(\dfrac{\pi\|\lfloor\boldsymbol{A}\rfloor_r\|_{\mathrm{F}}^2}{2\|\boldsymbol{A} - \lfloor\boldsymbol{A}\rfloor_r\|_{\mathrm{F}}^2}\right)^{1/2}, & k = r. \end{cases}$$

Additionally, if the block size $k \ge r+2$,

$$\mathbb{E}\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|^2 \le \left(1 + \frac{2r}{k-r-1}\right)\left(\|\boldsymbol{A} - \lfloor\boldsymbol{A}\rfloor_r\|^2 + \frac{\mathrm{e}^2}{k-r}\|\boldsymbol{A} - \lfloor\boldsymbol{A}\rfloor_r\|_{\mathrm{F}}^2\right).$$

Last, if the block size $k \geq r + 4$,

$$\left(\mathbb{E}\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_4^4\right)^{1/2} \leq \left(1 + \frac{r+1}{k-r-3}\right)\left(\|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_r\|_4^2 + \frac{1}{\sqrt{k-r}}\|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_r\|_{\mathrm{F}}^2\right);$$
$$\left(\mathbb{E}\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|^4\right)^{1/2} \leq \left(1 + \frac{2(r+1)}{k-r-3}\right)\left(\|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_r\|^2 + \frac{\sqrt{3}\,\mathrm{e}^2}{k-r}\|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_r\|_{\mathrm{F}}^2\right).$$

Once we have derived expectation bounds for `RSVD`, we obtain matching bounds for `NysSVD` due to the close relationship between the two algorithms:

COROLLARY 8.8 (`NysSVD` expectation bounds). *Fix a psd matrix* $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ *and a target rank* $r \geq 1$. *Then* ***`NysSVD`*** *with* $k \geq r + 2$ *Gaussian initialization vectors produces a random approximation* $\hat{\boldsymbol{A}} \in \mathbb{R}^{N \times N}$ *that satisfies*

$$\frac{\mathbb{E}\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_*}{\|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_r\|_*} \leq \begin{cases} 1 + \dfrac{r}{k-r-1}, & k \geq r+2; \\ 1 + r\log\left(1 + \dfrac{\|\lfloor \boldsymbol{A} \rfloor_r\|_*}{\|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_r\|_*}\right), & k = r+1; \\ 1 + r\left(\dfrac{\pi\|\lfloor \boldsymbol{A} \rfloor_r\|_*}{2\|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_r\|_*}\right)^{1/2}, & k = r. \end{cases}$$

Additionally, if the block size $k \geq r + 2$,

$$\mathbb{E}\|\boldsymbol{A} - \hat{\boldsymbol{A}}\| \leq \left(1 + \frac{2r}{k-r-1}\right)\left(\|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_r\| + \frac{\mathrm{e}^2}{k-r}\|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_r\|_*\right).$$

Last, if the block size $k \geq r + 4$,

$$\left(\mathbb{E}\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_{\mathrm{F}}^2\right)^{1/2} \leq \left(1 + \frac{r+1}{k-r-3}\right)\left(\|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_r\|_{\mathrm{F}} + \frac{1}{\sqrt{k-r}}\|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_r\|_*\right);$$
$$\left(\mathbb{E}\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|^2\right)^{1/2} \leq \left(1 + \frac{2(r+1)}{k-r-3}\right)\left(\|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_r\| + \frac{\sqrt{3}\mathrm{e}^2}{k-r}\|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_r\|_*\right).$$

Proof. `NysSVD` produces an approximation $\boldsymbol{A}\langle\boldsymbol{\Omega}\rangle$ satisfying

$$\|\boldsymbol{A} - \boldsymbol{A}\langle\boldsymbol{\Omega}\rangle\|_p = \|\boldsymbol{A}^{1/2}(\mathbf{I} - \boldsymbol{\Pi}_{\boldsymbol{A}^{1/2}\boldsymbol{\Omega}})\boldsymbol{A}^{1/2}\|_p = \|(\mathbf{I} - \boldsymbol{\Pi}_{\boldsymbol{A}^{1/2}\boldsymbol{\Omega}})\boldsymbol{A}^{1/2}\|_{2p}^2,$$

where $\boldsymbol{\Omega} \in \mathbb{R}^{N \times k}$ is a random Gaussian matrix. We recognize $\|(\mathbf{I} - \boldsymbol{\Pi}_{\boldsymbol{A}^{1/2}\boldsymbol{\Omega}})\boldsymbol{A}^{1/2}\|_{2p}^2$ as the square error of `RSVD` applied to the matrix $\boldsymbol{A}^{1/2}$ and apply the error bounds given in Theorem 8.7. ∎

The error bounds in Theorem 8.7 and Corollary 8.8 can be used in the analysis of related randomized algorithms. For example, the recent paper [38] uses Theorem 8.7 and Corollary 8.8 to control the variance of several randomized trace estimators.

9. Analysis of Krylov algorithms. In this section, we analyze `RBKI` and `NysBKI` and provide a matching analysis of `RSI` and `NysSI` for comparison. We develop both *gapless* error bounds that do not require any distance between the singular values and *gapped* error bounds that grow increasingly strong with the size of singular value gaps.

The gapless error bounds are given in Theorem 9.1, and the proof appears in subsection 9.3. Instead of bounding the error ratio

$$\frac{\mathbb{E}\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|^2}{\sigma_{r+1}(\boldsymbol{A})^2},$$

which is ideally close to one, these bounds apply to the log-error ratio

$$\log\left(\frac{\mathbb{E}\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|^2}{\sigma_{r+1}(\boldsymbol{A})^2}\right),$$

which is ideally close to zero. We demonstrate that the log-error ratio decreases at a rate $\mathcal{O}(1/m)$ for `RSI` or `NystSI` and a faster rate $\mathcal{O}(1/m^2)$ for `RBKI` or `NysBKI`. In each case, m is the total number of multiplications with $\boldsymbol{A}$ or $\boldsymbol{A}^*$.

THEOREM 9.1 (Gapless error bounds). *Fix a matrix $\boldsymbol{A} \in \mathbb{R}^{L\times N}$ and a target rank $r \geq 1$ such that $\sigma_{r+1}(\boldsymbol{A}) > 0$. Then `RSI` with $k \geq r + 2$ Gaussian initialization vectors and m matrix multiplications generates a random approximation $\hat{\boldsymbol{A}} \in \mathbb{R}^{L\times N}$ that satisfies*

$$\text{(RSI)} \qquad \log\left(\frac{\mathbb{E}\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|^2}{\sigma_{r+1}(\boldsymbol{A})^2}\right) \leq \frac{1}{m-1}\log\left(1 + \frac{r}{k-r-1}\sum_{i>r}\frac{\sigma_i(\boldsymbol{A})^2}{\sigma_{r+1}(\boldsymbol{A})^2}\right).$$

With the same initialization and the same number of multiplications, `RBKI` satisfies

$$\text{(RBKI)} \qquad \log\left(\frac{\mathbb{E}\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|^2}{\sigma_{r+1}(\boldsymbol{A})^2}\right) \leq \frac{1}{4(m-2)^2}\left[\log\left(4 + \frac{4r}{k-r-1}\sum_{i>r}\frac{\sigma_i(\boldsymbol{A})^2}{\sigma_{r+1}(\boldsymbol{A})^2}\right)\right]^2.$$

If $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ is a psd matrix, `NystSI` satisfies

$$\text{(NystSI)} \qquad \log\left(\frac{\mathbb{E}\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|^2}{\sigma_{r+1}(\boldsymbol{A})^2}\right) \leq \frac{1}{m-\frac{1}{2}}\log\left(1 + \frac{r}{k-r-1}\sum_{i>r}\frac{\sigma_i(\boldsymbol{A})^2}{\sigma_{r+1}(\boldsymbol{A})^2}\right),$$

and `NysBKI` satisfies

$$\text{(NysBKI)} \quad \log\left(\frac{\mathbb{E}\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|^2}{\sigma_{r+1}(\boldsymbol{A})^2}\right) \leq \frac{1}{8(m-\frac{3}{2})^2}\left[\log\left(4 + \frac{4r}{k-r-1}\sum_{i>r}\frac{\sigma_i(\boldsymbol{A})^2}{\sigma_{r+1}(\boldsymbol{A})^2}\right)\right]^2.$$

The gapless error bounds lead to algorithms insights. For example, `RBKI` and `NysBKI` converge at a fundamentally faster rate than the other algorithms. However, these bounds only describe the worst-case performance, which applies when the gaps between singular values are negligibly small. In more realistic applications, including all the test problems in section 7, we find that noticeable gaps appear. Therefore, we develop a set of gapped error bounds in Theorem 9.2 that guarantee faster, exponential convergence rates. The proof appears in subsection 9.4.

THEOREM 9.2 (Gapped error bounds). *Fix a matrix $\boldsymbol{A} \in \mathbb{R}^{L\times N}$, a Schatten p-norm with $2 \leq p \leq \infty$, and two singular values $\sigma_r(\boldsymbol{A})$ and $\sigma_s(\boldsymbol{A})$ with $r < s$. Define the singular value gap*

$$\gamma = \frac{\sigma_r(\boldsymbol{A}) - \sigma_s(\boldsymbol{A})}{\sigma_r(\boldsymbol{A}) + \sigma_s(\boldsymbol{A})} \in [0, 1],$$

and let $\lfloor \boldsymbol{A} \rfloor_r$ *and* $\lfloor \boldsymbol{A} \rfloor_{s-1}$ *denote optimal rank-*r *and rank-*$(s-1)$ *approximations of* $\boldsymbol{A}$*. Then,* `RSI` *with* $k \geq s+1$ *Gaussian initialization vectors and* m *matrix multiplications generates a random approximation* $\hat{\boldsymbol{A}} \in \mathbb{R}^{L \times N}$ *that satisfies*

$$\text{(RSI)} \qquad \mathbb{E}\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_p^2 \leq \|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_r\|_p^2 + \mathrm{e}^{-4(m-2)\gamma} \cdot \frac{s-1}{k-s} \|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_{s-1}\|_{\mathrm{F}}^2.$$

With the same initialization and the same number of multiplications, `RBKI` *satisfies*

$$\text{(RBKI)} \qquad \mathbb{E}\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_p^2 \leq \|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_r\|_p^2 + 4\,\mathrm{e}^{-4(m-2)\sqrt{\gamma}} \cdot \frac{s-1}{k-s} \|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_{s-1}\|_{\mathrm{F}}^2.$$

If $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ *is a psd matrix and* $1 \leq p \leq \infty$*,* `NysSI` *satisfies*

$$\text{(NysSI)} \qquad \mathbb{E}\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_p^2 \leq \|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_r\|_p^2 + \mathrm{e}^{-4(m-3/2)\gamma} \cdot \frac{s-1}{k-s} \|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_{s-1}\|_{\mathrm{F}}^2.$$

and `NysBKI` *satisfies*

$$\text{(NysBKI)} \quad \mathbb{E}\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_p^2 \leq \|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_r\|_p^2 + 4\,\mathrm{e}^{-4(m-3/2)\sqrt{2\gamma}} \cdot \frac{s-1}{k-s} \|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_{s-1}\|_{\mathrm{F}}^2.$$

The gapped error bounds show how Krylov methods capitalize upon small singular value gaps $\gamma \ll 1$ to achieve accelerated convergence. For example, we can ensure an error bound

$$\mathbb{E}\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_{\mathrm{F}}^2 \leq (1+\varepsilon) \cdot \|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_r\|_{\mathrm{F}}^2,$$

if we set $m = \mathcal{O}(\log(1/\varepsilon)/\gamma)$ in `RSI` or $m = \mathcal{O}(\log(1/\varepsilon)/\sqrt{\gamma})$ in `RBKI`. The change from a $1/\gamma$-dependence to a $1/\sqrt{\gamma}$-dependence leads to an accelerated convergence rate in many applications.

Theorems 9.1 and 9.2 improve upon existing results for `RSI` [54, 50, 14], `RBKI` [74, 107, 33, 98, 7, 8, 73], `NysSI`, and `NysBKI`. Musco and Musco [74] previously established the $\mathcal{O}(m^{-2})$ gapless convergence rate for `RBKI`, and we improve this result by obtaining explicit constants. Our gapped error bounds for `RBKI` and `RSI` also feature sharper constants than previous results [50, 14, 33, 98]. Last, the bounds for `NysSI` and `NysBKI` are completely new.

Because of the sharp, explicit nature of our estimates, we are able to make quantitative predictions: the gapless (Theorem 9.1) and gapped (Theorem 9.2) bounds predict that `NysSI` outperforms `RSI` by one-half of a matrix–matrix multiplication while `NysBKI` outperforms `RBKI` by a factor of $\sqrt{2}$ matrix–matrix multiplications. These predictions are in close agreement with numerical experiments (Figures 5 and 6). To illustrate the quantitative nature of our bounds, we evaluate these bounds for the matrix $\boldsymbol{B}$ as constructed in subsection 7.1 and present the output in Figure 13.

Our error bounds represent a step forward in the theoretical understanding of Krylov methods, but they are not the end of the story. Recent work [6, 8, 73] has begun to develop gapless error bounds for `RBKI` in the Frobenius norm, which complement the spectral norm approximation guarantees given here. Additionally, our main error bounds in Theorems 9.1 and 9.2 only apply when the block size k exceeds the target rank r. Although it is impossible to establish any *gapless* error bounds in the case $k < r$, Meyer and coauthors [73] (also see earlier work of [113]) have established a *gapped* error bound for `RBKI` for $k < r$ [73, Thm. 1]. Their result is more complicated than Theorems 9.1 and 9.2 and lacks explicit constants, but it provides additional evidence of the computational advantages of block Krylov methods.

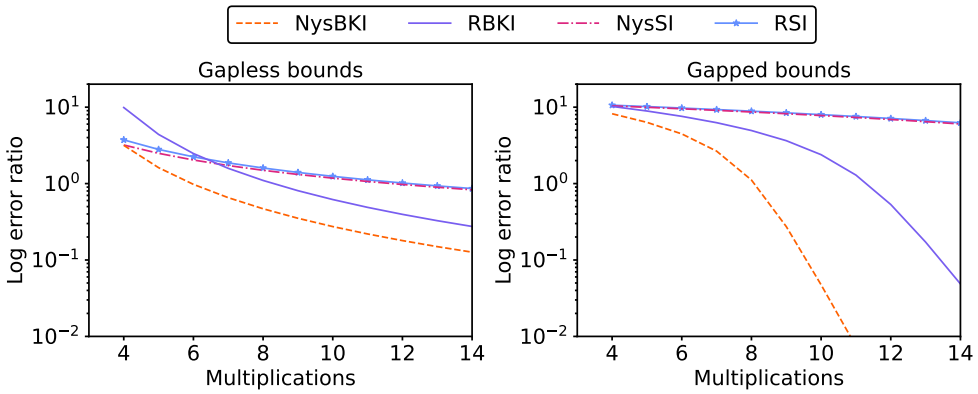

FIG. 13. ***(Gapless vs. gapped bounds).*** *Gapless (left) versus gapped (right) error bounds on the log error ratio* $\log\bigl(\mathbb{E}\|\boldsymbol{B}-\hat{\boldsymbol{B}}\|^2/\sigma_{r+1}(\boldsymbol{B})^2\bigr)$ *when approximating the matrix* $\boldsymbol{B}$ *from subsection 7.1 with block size* $k = 100$ *and target rank* $r = 75$.

The rest of this section contains an overview of our technical approach (subsection 9.1) and background material on Chebyshev polynomials (subsection 9.2), followed by a derivation of the the gapped error bounds (subsection 9.4) and the gapless error bounds (subsection 9.3).

9.1. Overview of technical approach. To prove the gapped and gapless error bounds, our main approach is to apply a polynomial filter $\phi(\boldsymbol{A})$ that increases the top singular values and decreases the bottom singular values. As we choose a filter, we need to ensure that $\phi(\boldsymbol{A})\boldsymbol{\Omega}$ lies inside the approximation space. Therefore, we use power function filters to analyze subspace iteration methods, and we use Chebyshev polynomial filters to analyze Krylov methods. The enhanced ability of Chebyshev polynomials to separate the top singular values from the bottom singular values is why we obtain such powerful Krylov bounds. Chebyshev polynomials have long been used in this context; for example, see Kuczyński and Woźniakowski [60].

Our filtering analysis relies on the fact that

$$\|(\mathbf{I} - \boldsymbol{\Pi}_{\phi(\boldsymbol{A})\boldsymbol{\Omega}})\phi(\boldsymbol{A})\|, \tag{9.1}$$

is the error of `RSVD` applied to a matrix $\phi(\boldsymbol{A})$, and the `RSVD` error is bounded by Theorem 8.1. The main difficulty is relating the `RSVD` error (9.1) to the quantity

$$\|(\mathbf{I} - \boldsymbol{\Pi}_{\phi(\boldsymbol{A})\boldsymbol{\Omega}})\boldsymbol{A}\|,$$

the error actually attained by `RSI` and `RBKI` algorithms.

We present two mathematical arguments that relate $\|(\mathbf{I} - \boldsymbol{\Pi}_{\phi(\boldsymbol{A})\boldsymbol{\Omega}})\phi(\boldsymbol{A})\|$ with $\|(\mathbf{I} - \boldsymbol{\Pi}_{\phi(\boldsymbol{A})\boldsymbol{\Omega}})\boldsymbol{A}\|$. First, we use convexity to establish Lemma 9.4, which allows us to prove the gapless bounds. For `RSI`, the convexity argument is already visible in [54, Prop. 8.6]. For `RBKI`, the crucial new observation is that Chebyshev polynomials are not globally convex, yet they admit supporting lines on the range $x \geq 1$. The supporting lines are all that we need to extend the convexity-based arguments. See Figure 14 for an illustration. Second, as a more standard mathematical result, we use matrix submultiplicativity to establish Lemma 9.5, which allows us to prove the gapped bounds.

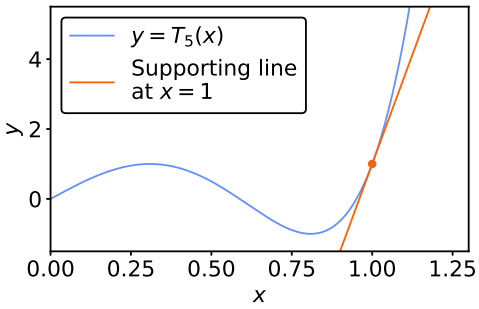

FIG. 14. ***(Chebyshev polynomials are almost convex).*** *The Chebyshev polynomial $T_5(x)$ admits supporting lines on the range $x \geq 1$.*

The gapless and gapped error bounds apply to the *untruncated* approximation $\hat{\boldsymbol{A}}$, which is the approximation returned by our pseudocode and recommended for most practical applications. However, similar bounds extend to the approximation $\lfloor \hat{\boldsymbol{A}} \rfloor_r$, which comes from applying an r-truncated singular value decomposition to $\hat{\boldsymbol{A}}$. The gapped error bounds (Theorem 9.2) remain valid even when we replace $\hat{\boldsymbol{A}}$ with $\lfloor \hat{\boldsymbol{A}} \rfloor_r$, due to the reverse Eckart–Young inequality [50, Thm. 3.4]. Likewise, we can prove gapless error bounds for $\lfloor \hat{\boldsymbol{A}} \rfloor_r$ by pursuing Musco and Musco's observation [74] that either there is a singular value gap and the gapped error bounds apply or else there is no gap and the approximation is close to optimal. We omit a detailed treatment.

9.2. Chebyshev polynomials. The qth Chebyshev polynomial $x \mapsto T_q(x)$ is a polynomial of degree q that shares some properties with the qth power function $x \mapsto x^q$. The values of both polynomials lie inside $[-1, 1]$ for $0 \leq x \leq 1$, and both polynomials exhibit a sharp rate of increase for $x \geq 1$. Lemma 9.3 provides more fine-grained estimates that will be used in the analysis.

LEMMA 9.3 (Chebyshev polynomials). *For $q \in \{0, 1, 2, \ldots\}$ and $x \geq 1$, define the Chebyshev polynomial T_q of the first kind by*

$$T_q(x) = \begin{cases} \cos(q \arccos x), & 0 \leq x \leq 1, \\ \cosh(q \operatorname{arcosh} x), & x \geq 1. \end{cases} \tag{9.2}$$

Then, the following properties hold:

1. *T_q is a polynomial of degree q with the same parity as q. If q is odd, then T_q is odd. If q is even, then T_q is even.*
2. *$T_q(x)$ is bounded above by*

$$|T_q(x)| \leq 1, \qquad 0 \leq x \leq 1.$$

3. *$T_q(x)$ is increasing on the interval $x \geq 1$, and it is bounded below as*

$$T_q\left(\frac{1+r}{1-r}\right) \geq \frac{1}{2} \mathrm{e}^{2q\sqrt{r}}, \qquad 0 \leq r < 1.$$

In contrast, the quantity x^q exhibits a slower rate of increase:

$$\left(\frac{1+r}{1-r}\right)^q \geq \mathrm{e}^{2qr}, \qquad 0 \leq r < 1.$$

4. *The functional inverse of the Chebyshev polynomial satisfies*

$$T_q^{-1}(x) \le \exp\left(\frac{1}{2}\left[\frac{\log(2x)}{q}\right]^2\right), \qquad x \ge 1.$$

5. *The function $\phi(x) = xT_q(x^{1/2})^2$ has a supporting line at every point $(x, \phi(x))$ with $x \ge 1$, as does the function $\phi(x) = xT_q(x^{1/4})^2$.*

Proof. Parts 1 and 2 are standard facts presented in [71, Ch. 2]. Parts 3–5 are technical results proved in Appendix C. ∎

9.3. Gapless error bounds. To prove the gapless error bounds, we will need the following version of Jensen's inequality. Compare this result with [54, Prop. 8.6].

LEMMA 9.4 (Jensen's inequality with "almost" convex functions). *Consider a diagonal, psd matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$, a random rank-$r$ orthogonal projection $\boldsymbol{Q} \in \mathbb{R}^{N\times N}$, and a function $f : [0,\infty) \to \mathbb{R}$ that has a supporting line at every point $(x, f(x))$ for $x \ge \sigma_{r+1}(\boldsymbol{A})^2$. Then*

$$f(\mathbb{E}\|(\mathbf{I}-\boldsymbol{Q})\boldsymbol{A}\|^2) \le \mathbb{E}\|(\mathbf{I}-\boldsymbol{Q})f(\boldsymbol{A}^2)(\mathbf{I}-\boldsymbol{Q})\|. \tag{9.3}$$

Proof. We will repeatedly use the fact that $\boldsymbol{QA}$ is a rank-r approximation of $\boldsymbol{A}$ and hence $\|\boldsymbol{A}-\boldsymbol{QA}\| \ge \sigma_{r+1}(\boldsymbol{A})$ by Weyl's inequality [56, Thm. 4.3.1]. First, we set $x_0 = \mathbb{E}\|\boldsymbol{A}-\boldsymbol{QA}\|^2 \ge \sigma_{r+1}(\boldsymbol{A})^2$ and use the supporting line property to argue

$$f(x_0) = \mathbb{E}[f(x_0) + f'(x_0)(\|\boldsymbol{A}-\boldsymbol{QA}\|^2 - x_0)] \le \mathbb{E}\, f(\|\boldsymbol{A}-\boldsymbol{QA}\|^2)$$

Next, we define the random vector $\boldsymbol{v}$ to be a dominant eigenvector of $(\mathbf{I}-\boldsymbol{Q})\boldsymbol{A}^2(\mathbf{I}-\boldsymbol{Q})$, lying in the range of $\mathbf{I}-\boldsymbol{Q}$ and normalized so that $\|\boldsymbol{v}\| = 1$. We set $X = \|(\mathbf{I}-\boldsymbol{Q})\boldsymbol{A}\|^2$ and observe that

$$X = \boldsymbol{v}^*(\mathbf{I}-\boldsymbol{Q})\boldsymbol{A}^2(\mathbf{I}-\boldsymbol{Q})\boldsymbol{v}^* = \boldsymbol{v}^*\boldsymbol{A}^2\boldsymbol{v}^* = \sum\nolimits_{i=1}^N a_{ii}^2 v_i^2, \tag{9.4}$$

where a_{ii} denotes the ith diagonal entry of $\boldsymbol{A}$ (recall that $\boldsymbol{A}$ is a diagonal matrix) and v_i is the ith entry of $\boldsymbol{v}$. Last, we use (9.4) and the supporting line property to argue

$$\begin{aligned} f(X) &= \sum\nolimits_{i=1}^N v_i^2\left[f(X) + f'(X)(a_{ii}^2 - X)\right] \\ &\le \sum\nolimits_{i=1}^N v_i^2 f(a_{ii}^2) \\ &= \boldsymbol{v}^*(\mathbf{I}-\boldsymbol{Q})f(\boldsymbol{A}^2)(\mathbf{I}-\boldsymbol{Q})\boldsymbol{v} \\ &\le \|(\mathbf{I}-\boldsymbol{Q})f(\boldsymbol{A}^2)(\mathbf{I}-\boldsymbol{Q})\|, \end{aligned}$$

thus verifying the stated inequality (9.3). ∎

We use Lemma 9.4 to prove the main result, Theorem 9.1.

Proof of Theorem 9.1. We assume that $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ is diagonal and psd, with nonincreasing diagonal entries (Lemma 8.3). By replacing $\boldsymbol{A}$ with $\boldsymbol{A}/\sigma_{r+1}(\boldsymbol{A})$ if necessary, we also assume that $\sigma_{r+1}(\boldsymbol{A}) = 1$. Last, we partition the test matrix $\boldsymbol{\Omega} = \left[\begin{smallmatrix}\boldsymbol{\Omega}_1\\ \boldsymbol{\Omega}_2\end{smallmatrix}\right]$ into a $r \times k$ matrix $\boldsymbol{\Omega}_1$ and a $(N-r)\times k$ matrix $\boldsymbol{\Omega}_2$.

To analyze `RSI`, we introduce the orthogonal projection $\boldsymbol{Q} = \boldsymbol{\Pi}_{\boldsymbol{A}^{m-1}\boldsymbol{\Omega}\boldsymbol{\Omega}_1^\dagger}$ and apply Lemma 9.4 with the convex function $f(x) = x^{m-1}$ to verify

$$\mathbb{E}\|(\mathbf{I}-\boldsymbol{Q})\boldsymbol{A}\|^{2(m-1)} \le \mathbb{E}\|(\mathbf{I}-\boldsymbol{Q})\boldsymbol{A}^{m-1}\|^2.$$

We identify $\mathbb{E}\|(\mathbf{I}-\boldsymbol{Q})\boldsymbol{A}^{m-1}\|^2$ as the mean-square error of `RSVD` applied to the matrix $\boldsymbol{A}^{m-1}$ with a test matrix $\boldsymbol{\Omega}\boldsymbol{\Omega}_1^\dagger$, and we mimic the proof of Theorem 8.1 to show that

$$\mathbb{E}\|(\mathbf{I}-\boldsymbol{Q})\boldsymbol{A}^{m-1}\|^2 \le 1 + \frac{r}{k-r-1}\sum\nolimits_{i=r+1}^{N}\sigma_i(\boldsymbol{A})^{2(m-1)}.$$

We take logarithms and divide through by $m-1$ to obtain

$$\log\bigl(\mathbb{E}\|(\mathbf{I}-\boldsymbol{Q})\boldsymbol{A}\|^2\bigr) \le \frac{1}{m-1}\log\biggl(1 + \frac{r}{k-r-1}\sum\nolimits_{i=r+1}^{N}\sigma_i(\boldsymbol{A})^{2(m-1)}\biggr),$$

which is a stronger version of the error bound for `RSI`.

To analyze `RBKI`, we define $\boldsymbol{Q} = \boldsymbol{\Pi}_{\boldsymbol{A}T_{m-2}(\boldsymbol{A})\boldsymbol{\Omega}\boldsymbol{\Omega}_1^\dagger}$. Then we apply Lemma 9.3 part (5) to identify the "almost" convex function $f(x) = x\bigl[T_{m-2}\bigl(x^{1/2}\bigr)\bigr]^2$, and we invoke Lemma 9.4 to obtain

$$\bigl[T_{m-2}\bigl(\bigl(\mathbb{E}\|(\mathbf{I}-\boldsymbol{Q})\boldsymbol{A}\|^2\bigr)^{1/2}\bigr)\bigr]^2 \le f(\mathbb{E}\|(\mathbf{I}-\boldsymbol{Q})\boldsymbol{A}\|^2) \le \mathbb{E}\|(\mathbf{I}-\boldsymbol{Q})\boldsymbol{A}T_{m-2}(\boldsymbol{A})\|^2.$$

We recognize $\mathbb{E}\|(\mathbf{I}-\boldsymbol{Q})\boldsymbol{A}T_{m-2}(\boldsymbol{A})\|^2$ as the mean-square error of `RSVD` applied to the matrix $\boldsymbol{A}T_{m-2}(\boldsymbol{A})$. Hence, by mimicking the proof of Theorem 8.1, we obtain

$$\mathbb{E}\|(\mathbf{I}-\boldsymbol{Q})\boldsymbol{A}T_{m-2}(\boldsymbol{A})\|^2 \le 1 + \frac{r}{k-r-1}\sum\nolimits_{i=r+1}^{N}\sigma_i(\boldsymbol{A})^2.$$

Last, we take square roots, apply the inverse Chebyshev polynomial T_{m-2}^{-1}, take logarithms, and multiply through by 2 to obtain

$$\begin{aligned}\log\bigl(\mathbb{E}\|(\mathbf{I}-\boldsymbol{Q})\boldsymbol{A}\|^2\bigr) &\le 2\log\biggl(T_{m-2}^{-1}\biggl(\sqrt{1 + \frac{r}{k-r-1}\sum\nolimits_{i=r+1}^{N}\sigma_i(\boldsymbol{A})^2}\biggr)\biggr)\\ &\le \frac{1}{4(m-2)^2}\biggl[\log\biggl(4 + \frac{4r}{k-r-1}\sum\nolimits_{i=r+1}^{N}\sigma_i(\boldsymbol{A})^2\biggr)\biggr]^2,\end{aligned}$$

which confirms the error bound for `RBKI`. The last line uses the bound for T_{m-2}^{-1} which was stated in Lemma 9.3 part 4.

We confirm the error bounds for `NysSI` and `NysBKI` in a similar fashion. For `NysSI`, we define the orthogonal projection $\boldsymbol{Q} = \boldsymbol{\Pi}_{\boldsymbol{A}^{m-1/2}\boldsymbol{\Omega}\boldsymbol{\Omega}_1^\dagger}$ and calculate

$$\begin{aligned}\|\boldsymbol{A} - \hat{\boldsymbol{A}}\| \le \|\boldsymbol{A} - \boldsymbol{A}\langle\boldsymbol{A}^{m-1}\boldsymbol{\Omega}\boldsymbol{\Omega}_1^\dagger\rangle\| &= \|\boldsymbol{A}^{1/2}\bigl(\mathbf{I}-\boldsymbol{Q}\bigr)\boldsymbol{A}^{1/2}\|\\ &= \|\bigl(\mathbf{I}-\boldsymbol{Q}\bigr)\boldsymbol{A}^{1/2}\|^2 = \|\bigl(\mathbf{I}-\boldsymbol{Q}\bigr)\boldsymbol{A}\bigl(\mathbf{I}-\boldsymbol{Q}\bigr)\| \le \|\bigl(\mathbf{I}-\boldsymbol{Q}\bigr)\boldsymbol{A}\|.\end{aligned}$$

By applying Lemma 9.4 with $f(x) = x^{m-1/2}$, we obtain

$$\mathbb{E}\|(\mathbf{I}-\boldsymbol{Q})\boldsymbol{A}\|^{2(m-1/2)} \le \mathbb{E}\|(\mathbf{I}-\boldsymbol{Q})\boldsymbol{A}^{m-1/2}\|^2,$$

which leads to an error bound for `NysSI`.

Last, for `NysBKI`, we define the orthogonal projection $\boldsymbol{Q} = \boldsymbol{\Pi}_{\boldsymbol{A}T_{2m-1}(\boldsymbol{A}^{1/2})\boldsymbol{\Omega}\boldsymbol{\Omega}_1^\dagger}$ so that $\|\boldsymbol{A} - \hat{\boldsymbol{A}}\| \le \|\bigl(\mathbf{I}-\boldsymbol{Q}\bigr)\boldsymbol{A}\|$. By applying Lemma 9.4 with $f(x) = x\bigl[T_{2m-1}\bigl(x^{1/4}\bigr)\bigr]^2$, we obtain

$$\bigl[T_{2m-1}\bigl(\bigl(\mathbb{E}\|(\mathbf{I}-\boldsymbol{Q})\boldsymbol{A}\|^2\bigr)^{1/4}\bigr)\bigr]^2 \le \mathbb{E}\|(\mathbf{I}-\boldsymbol{Q})\boldsymbol{A}T_{2m-1}(\boldsymbol{A}^{1/2})\|^2.$$

which leads to an error bound for `NysBKI` and completes the proof of Theorem 9.1. ∎

9.4. Gapped error bounds. To establish the gapped error bounds, we will need to apply a polynomial ϕ that enhances the singular value gap. To that end, we establish the following filtering lemma.

LEMMA 9.5 (Enhancing the gap). *Choose a diagonal, psd matrix with nonincreasing entries: $\boldsymbol{A} = \operatorname{diag}(a_{11}, a_{22}, \dots, a_{NN})$. Fix two indices $1 \le r < s \le N$ and a function $\phi(x)$ that is positive for $x \ge a_{rr}$, and partition the matrix $\boldsymbol{A}$ as*

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_1 & & \\ & \boldsymbol{A}_2 & \\ & & \boldsymbol{A}_3 \end{bmatrix},$$

where $\boldsymbol{A}_1$, $\boldsymbol{A}_2$, and $\boldsymbol{A}_3$ are square matrices with dimensions r, $s-1-r$, and $N-s+1$. If $\boldsymbol{\Omega} \in \mathbb{R}^{N \times k}$ is a Gaussian matrix with block size $k \ge s+1$, then

$$\mathbb{E}\|(\mathbf{I} - \boldsymbol{\Pi}_{\boldsymbol{A}\phi(\boldsymbol{A})\boldsymbol{\Omega}})\boldsymbol{A}\|_p^2 \le \left\| \begin{bmatrix} \mathbf{0} & & \\ & \boldsymbol{A}_2 & \\ & & \boldsymbol{A}_3 \end{bmatrix} \right\|_p^2 + \frac{s-1}{k-s} \cdot \frac{\|\boldsymbol{A}_3\phi(\boldsymbol{A}_3)\|_p^2}{\phi(a_{rr})^2}.$$

for any Schatten p-norm with $2 \le p \le \infty$.

Proof. We partition the matrix $\boldsymbol{\Omega}$ as $\boldsymbol{\Omega} = \left[\begin{smallmatrix} \boldsymbol{\Omega}_\alpha \\ \boldsymbol{\Omega}_\beta \end{smallmatrix}\right]$, where $\boldsymbol{\Omega}_\alpha$ is $(s-1) \times k$ and $\boldsymbol{\Omega}_\beta$ is $(N-s+1) \times k$. Then we set $\boldsymbol{Q} = \boldsymbol{\Pi}_{\boldsymbol{A}\phi(\boldsymbol{A})\boldsymbol{\Omega}\boldsymbol{\Omega}_\alpha^\dagger}$ and apply the coarse upper bound

$$\|(\mathbf{I} - \boldsymbol{Q})\boldsymbol{A}\|_p^2 \le \left\| \begin{bmatrix} \mathbf{0} & & \\ & \boldsymbol{A}_2 & \\ & & \boldsymbol{A}_3 \end{bmatrix} \right\|_p^2 + \left\| (\mathbf{I} - \boldsymbol{Q}) \begin{bmatrix} \boldsymbol{A}_1 & & \\ & \mathbf{0} & \\ & & \mathbf{0} \end{bmatrix} \right\|_p^2$$

We can bound the second term using

$$\left\| (\mathbf{I} - \boldsymbol{Q}) \begin{bmatrix} \boldsymbol{A}_1\phi(\boldsymbol{A}_1) & & \\ & \boldsymbol{A}_2\phi(\boldsymbol{A}_2) & \\ & & \mathbf{0} \end{bmatrix} \begin{bmatrix} \phi(\boldsymbol{A}_1)^{-1} & & \\ & \mathbf{0} & \\ & & \mathbf{0} \end{bmatrix} \right\|_p^2 \tag{9.5}$$

$$\le \frac{1}{\phi(a_{rr})^2} \left\| (\mathbf{I} - \boldsymbol{Q}) \begin{bmatrix} \boldsymbol{A}_1\phi(\boldsymbol{A}_1) & & \\ & \boldsymbol{A}_2\phi(\boldsymbol{A}_2) & \\ & & \mathbf{0} \end{bmatrix} \right\|_p^2 \tag{9.6}$$

$$\le \frac{1}{\phi(a_{rr})^2} \|\boldsymbol{A}_3\phi(\boldsymbol{A}_3)\boldsymbol{\Omega}_\beta\boldsymbol{\Omega}_\alpha^\dagger\|_p^2. \tag{9.7}$$

The inequality (9.6) depends on the fact that the operator norm of the right-most matrix is $1/\phi(a_{rr})$, while inequality (9.7) follows from Proposition 8.5. Last, we take expectations and apply the exact calculations for Gaussian matrices from Proposition 8.6 to show

$$\mathbb{E}\|\boldsymbol{A}_3\phi(\boldsymbol{A}_3)\boldsymbol{\Omega}_\beta\boldsymbol{\Omega}_\alpha^\dagger\|_p^2 \le \mathbb{E}\|\boldsymbol{A}_3\phi(\boldsymbol{A}_3)\boldsymbol{\Omega}_\beta\boldsymbol{\Omega}_\alpha^\dagger\|_{\mathrm{F}}^2 = \frac{s-1}{k-s}\,\mathbb{E}\|\boldsymbol{A}_3\phi(\boldsymbol{A}_3)\|_{\mathrm{F}}^2,$$

which completes the proof. ∎

We apply Lemma 9.5 to establish the main result, Theorem 9.2.

Proof of Theorem 9.2. We assume without loss of generality that $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ is diagonal and psd, with nonincreasing diagonal entries (Lemma 8.3). We then apply

Lemma 9.5 with $\phi(x) = x^{m-2}$ to bound the RSI error as follows:

$$\mathbb{E}\|(\mathbf{I} - \mathbf{\Pi}_{\boldsymbol{A}^{m-1}\boldsymbol{\Omega}})\boldsymbol{A}\|_p^2 \leq \|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_r\|_p^2 + \frac{s-1}{k-s} \sum_{i=s}^{N} \frac{\sigma_i(\boldsymbol{A})^{2(m-1)}}{\sigma_r(\boldsymbol{A})^{2(m-2)}}.$$

Lemma 9.3 part 3 gives the numerical identity

$$\left(\frac{\sigma_s(\boldsymbol{A})}{\sigma_r(\boldsymbol{A})}\right)^{2(m-2)} = \left(\frac{1-\gamma}{1+\gamma}\right)^{2(m-2)} \leq \mathrm{e}^{-4(m-2)\gamma},$$

which confirms the RSI error bound. The NysSI error bound is proved similarly. Observe that

$$\begin{aligned}\|\boldsymbol{A} - \boldsymbol{A}\langle \boldsymbol{A}^{m-1}\boldsymbol{\Omega}\rangle\|_p &= \|\boldsymbol{A}^{1/2}(\mathbf{I} - \mathbf{\Pi}_{\boldsymbol{A}^{m-1/2}\boldsymbol{\Omega}})\boldsymbol{A}^{1/2}\|_p = \|(\mathbf{I} - \mathbf{\Pi}_{\boldsymbol{A}^{m-1/2}\boldsymbol{\Omega}})\boldsymbol{A}^{1/2}\|_{2p}^2 \\ &= \|(\mathbf{I} - \mathbf{\Pi}_{\boldsymbol{A}^{m-1/2}\boldsymbol{\Omega}})\boldsymbol{A}(\mathbf{I} - \mathbf{\Pi}_{\boldsymbol{A}^{m-1/2}\boldsymbol{\Omega}})\|_p \leq \|(\mathbf{I} - \mathbf{\Pi}_{\boldsymbol{A}^{m-1/2}\boldsymbol{\Omega}})\boldsymbol{A}\|_p.\end{aligned}$$

We bound $\mathbb{E}\|(\mathbf{I} - \mathbf{\Pi}_{\boldsymbol{A}^{m-1/2}\boldsymbol{\Omega}})\boldsymbol{A}\|_p^2$ by applying Lemma 9.5 with $\phi(x) = x^{m-3/2}$.

To control the RBKI error, we apply Lemma 9.5 with $\phi(x) = T_{m-2}(x/\sigma_s(\boldsymbol{A}))$, which gives

$$\mathbb{E}\|\boldsymbol{A} - \mathbf{\Pi}_{\boldsymbol{A}\phi(\boldsymbol{A})\boldsymbol{\Omega}}\boldsymbol{A}\|_p^2 \leq \|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_r\|_p^2 + \frac{s-1}{k-s} \sum_{i=s}^{N} \sigma_i(\boldsymbol{A})^2 \left(\frac{\phi(\sigma_i(\boldsymbol{A}))}{\phi(\sigma_r(\boldsymbol{A}))}\right)^2.$$

Then we use the properties of Chebyshev polynomials (Lemma 9.3 parts (2) and (3)) to check that $|\phi(\sigma_i(\boldsymbol{A}))| \leq 1$ for $i = s, \dots, N$ and

$$\phi(\sigma_r(\boldsymbol{A})) = T_{m-2}\left(\frac{\sigma_r(\boldsymbol{A})}{\sigma_s(\boldsymbol{A})}\right) = T_{m-2}\left(\frac{1+\gamma}{1-\gamma}\right) \geq \frac{1}{2}\mathrm{e}^{2(m-2)\sqrt{\gamma}}.$$

Similarly, to bound the NysBKI error, we define $\phi(x) = T_{2m-3}(\sqrt{x/\sigma_s(\boldsymbol{A})})$ so that $\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_p \leq \|\boldsymbol{A} - \mathbf{\Pi}_{\boldsymbol{A}\phi(\boldsymbol{A})\boldsymbol{\Omega}}\boldsymbol{A}\|_p$. We apply Lemma 9.5, which gives

$$\mathbb{E}\|\boldsymbol{A} - \mathbf{\Pi}_{\boldsymbol{A}\phi(\boldsymbol{A})\boldsymbol{\Omega}}\boldsymbol{A}\|_p^2 \leq \|\boldsymbol{A} - \lfloor \boldsymbol{A} \rfloor_r\|_p^2 + \frac{s-1}{k-s} \sum_{i=s}^{N} \sigma_i(\boldsymbol{A})^2 \left(\frac{\phi(\sigma_i(\boldsymbol{A}))}{\phi(\sigma_r(\boldsymbol{A}))}\right)^2.$$

We bound $|\phi(\sigma_i(\boldsymbol{A}))| \leq 1$ for $i = s, \dots, N$ and

$$\phi(\sigma_r(\boldsymbol{A})) = T_{2m-3}\left(\sqrt{\frac{1+\gamma}{1-\gamma}}\right) \geq T_{2m-3}\left(\frac{1+\gamma/2}{1-\gamma/2}\right) \geq \frac{1}{2}\mathrm{e}^{2(m-3/2)\sqrt{2\gamma}},$$

where we have used the numerical identity

$$\sqrt{\frac{1+\gamma}{1-\gamma}} \geq \frac{1+\gamma/2}{1-\gamma/2}, \qquad \text{for } \gamma \geq 0.$$

This completes the proof. ∎

10. Conclusion. In this work, we have described randomized low-rank approximation algorithms and provided user recommendations regarding which algorithms are the most efficient. We have focused on computations involving high-dimensional matrices, in which matrix multiplications are the main computational bottleneck.

We have established simple, explicit bounds which allow for precise descriptions of the differences between algorithms. Our results demonstrate that `RSVD` and `NysSVD` are fast and accurate when the singular values of the target matrix decay quickly. However, in settings of slow singular value decay, `RBKI` and `NysBKI` are the available algorithms with the greatest speed and robustness. We have presented numerical tests demonstrating the utility of these Krylov methods for principal component analysis and kernel spectral clustering.

The widespread adoption of `RBKI` and `NysBKI` will require changes in software, since `RSI` is currently the default randomized low-rank matrix approximation algorithm in Matlab [72] and sci-kit learn [84]. Yet, `RBKI` and `NysBKI` are of great benefit to computational scientists who have found `RSI` to be expensive while resulting in large random errors [23, 61, 111, 15, 19, 64]. `RBKI` and `NysBKI` are more accurate and scalable algorithms. To support the broader use of these Krylov methods, we have provided simple and stable pseudocode that cuts down on the cost of `RBKI` by roughly 33% of the number of matrix–vector products, and we have provided the first pseudocode for `NysBKI`, which is faster than `RBKI` by a factor of $\sqrt{2}$ matrix–vector products.

Acknowledgments. We acknowledge many helpful conversations with Tyler Chen, Mateo Díaz, Barbara Engelhardt, Ethan Epperly, Maksim Melnichenko, Riley Murray, and Christopher Musco.

Appendix A. Quality assurance for singular vectors. Algorithms A.1 and A.2 are adaptive versions of `RBKI` and `NysBKI` that we developed with assistance from Maksim Melnichenko and Riley Murray. The algorithms calculate the singular vector residuals and stop when each of the first r singular vector residuals falls below a threshold ε.

In these algorithms, we use several shortcuts to evaluate the residual error formula (6.1). If the low-rank approximation takes the form $\hat{\boldsymbol{A}} = \boldsymbol{\Pi}_{\boldsymbol{M}}\boldsymbol{A}$, the first term in (6.1) is zero and the residual is $r_i = \|(\boldsymbol{A}-\hat{\boldsymbol{A}})\hat{\boldsymbol{v}}_i\|$. If instead $\hat{\boldsymbol{A}} = \boldsymbol{A}\boldsymbol{\Pi}_{\boldsymbol{M}}$, the second term in (6.1) is zero and $r_i = \|(\boldsymbol{A}-\hat{\boldsymbol{A}})^*\hat{\boldsymbol{u}}_i\|$. Last, if $\hat{\boldsymbol{A}} = \boldsymbol{A}\langle\boldsymbol{M}\rangle$ is a Nyström approximation, the left and right singular vectors are the same, and $r_i = \sqrt{2}\|(\boldsymbol{A} - \hat{\boldsymbol{A}})\hat{\boldsymbol{v}}_i\|$.

Appendix B. Random matrix theory. In this section, we bound the moments (Lemma B.1) and inverse moments (Lemmas B.2 and B.3) of Gaussian random matrices. These are the sharpest available expectation bounds and concentration inequalities of their type, and they are helpful for analyzing randomized matrix computations. We use these results to prove Proposition 8.6 in Appendix B.1 and Theorem 8.7 in Appendix B.2.

LEMMA B.1 (Moment bounds). *Consider fixed matrices $\boldsymbol{S} \in \mathbb{R}^{L\times N}$ and $\boldsymbol{T} \in \mathbb{R}^{P\times Q}$ and a Gaussian matrix $\boldsymbol{G} \in \mathbb{R}^{N\times P}$. Then, $\boldsymbol{SGT}$ satisfies the equalities*

$$\begin{aligned}\mathbb{E}\|\boldsymbol{SGT}\|_{\mathrm{F}}^2 &= \|\boldsymbol{S}\|_{\mathrm{F}}^2\|\boldsymbol{T}\|_{\mathrm{F}}^2,\\ \mathbb{E}\|\boldsymbol{SGT}\|_4^4 &= \|\boldsymbol{S}\|_4^4\|\boldsymbol{T}\|_{\mathrm{F}}^4 + \|\boldsymbol{T}\|_4^4\|\boldsymbol{S}\|_{\mathrm{F}}^4 + \|\boldsymbol{S}\|_4^4\|\boldsymbol{T}\|_4^4,\end{aligned}$$

as well as the upper bounds

$$\begin{aligned}(\mathbb{E}\|\boldsymbol{SGT}\|^2)^{1/2} &\le \|\boldsymbol{S}\|\|\boldsymbol{T}\|_{\mathrm{F}} + \|\boldsymbol{T}\|\|\boldsymbol{S}\|_{\mathrm{F}},\\ (\mathbb{E}\|\boldsymbol{SGT}\|^4)^{1/4} &\le \|\boldsymbol{S}\|(\|\boldsymbol{T}\|_{\mathrm{F}}^4 + 2\|\boldsymbol{T}\|_4^4)^{1/4} + \|\boldsymbol{T}\|(\|\boldsymbol{S}\|_{\mathrm{F}}^4 + 2\|\boldsymbol{S}\|_4^4)^{1/4}.\end{aligned}$$

Algorithm A.1 `RBKI` with quality assurance

Input: Matrix $\boldsymbol{A} \in \mathbb{R}^{L \times N}$; block size k; number of singular vectors r; tolerance ε_{tol}
Output: Orthogonal $\boldsymbol{U}$, orthogonal $\boldsymbol{V}$, and diagonal $\boldsymbol{\Sigma}$ such that $\boldsymbol{A} \approx \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$

Generate a random matrix $\boldsymbol{Y}_0 \in \mathbb{R}^{N \times k}$
$i = 1$
$\boldsymbol{X}_1 = \boldsymbol{A}\boldsymbol{Y}_0$
$[\boldsymbol{X}_1, \sim] = \text{qr\_econ}(\boldsymbol{X}_1)$
$\boldsymbol{W}_2 = \boldsymbol{A}^*\boldsymbol{X}_1$
$\boldsymbol{R} = [\,]$; $\boldsymbol{S} = [\,]$; $\boldsymbol{E} = [\,]$
while $i = 1$ or one of the first r columns of $\boldsymbol{E}$ has norm $> \varepsilon$ **do**
  $i = i + 1$
  if i is even **then**
    $\boldsymbol{Y}_i = \boldsymbol{W}_i$
    $\boldsymbol{R}_{\bullet i} = \begin{bmatrix}\boldsymbol{Y}_2 & \boldsymbol{Y}_4 & \cdots & \boldsymbol{Y}_{i-2}\end{bmatrix}^* \boldsymbol{Y}_i$
    $\boldsymbol{Y}_i = \boldsymbol{Y}_i - \sum_{\text{even } j<i} \boldsymbol{Y}_j(\boldsymbol{Y}_j^*\boldsymbol{Y}_i)$ ▷ Orthog. w.r.t. even iterates (2×)
    $[\boldsymbol{Y}_i, \boldsymbol{R}_{ii}] = \text{qr\_econ}(\boldsymbol{Y}_i)$ ▷ Optional: stabilized QR (5.1)
    $\boldsymbol{R} = \begin{bmatrix}\boldsymbol{R} & \boldsymbol{R}_{\bullet i} \\ \boldsymbol{0} & \boldsymbol{R}_{ii}\end{bmatrix}$
    $\boldsymbol{Z}_{i+1} = \boldsymbol{A}\boldsymbol{Y}_i$
    $[\hat{\boldsymbol{U}}, \boldsymbol{\Sigma}, \hat{\boldsymbol{V}}] = \text{svd\_econ}(\boldsymbol{R}^*)$
    $\boldsymbol{E} = \begin{bmatrix}\boldsymbol{Z}_3 & \boldsymbol{Z}_5 & \cdots\end{bmatrix}\hat{\boldsymbol{V}} - \begin{bmatrix}\boldsymbol{X}_1 & \boldsymbol{X}_3 & \cdots\end{bmatrix}\hat{\boldsymbol{U}}\boldsymbol{\Sigma}$ ▷ Residual evaluation
  else
    $\boldsymbol{X}_i = \boldsymbol{Z}_i$
    $\boldsymbol{S}_{\bullet i} = \begin{bmatrix}\boldsymbol{X}_1 & \boldsymbol{X}_3 & \cdots & \boldsymbol{X}_{i-2}\end{bmatrix}^* \boldsymbol{X}_i$
    $\boldsymbol{X}_i = \boldsymbol{X}_i - \sum_{\text{odd } j<i} \boldsymbol{X}_j(\boldsymbol{X}_j^*\boldsymbol{X}_i)$ ▷ Orthog. w.r.t. odd iterates (2×)
    $[\boldsymbol{X}_i, \boldsymbol{S}_{ii}] = \text{qr\_econ}(\boldsymbol{X}_i)$ ▷ Optional: stabilized QR (5.1)
    $\boldsymbol{S} = \begin{bmatrix}\boldsymbol{S} & \boldsymbol{S}_{\bullet i} \\ \boldsymbol{0} & \boldsymbol{S}_{ii}\end{bmatrix}$
    $\boldsymbol{W}_{i+1} = \boldsymbol{A}^*\boldsymbol{X}_i$
    $[\hat{\boldsymbol{U}}, \boldsymbol{\Sigma}, \hat{\boldsymbol{V}}] = \text{svd\_econ}(\boldsymbol{S})$
    $\boldsymbol{E} = \begin{bmatrix}\boldsymbol{W}_2 & \boldsymbol{W}_4 & \cdots\end{bmatrix}\hat{\boldsymbol{U}} - \begin{bmatrix}\boldsymbol{Y}_2 & \boldsymbol{Y}_4 & \cdots\end{bmatrix}\hat{\boldsymbol{V}}\boldsymbol{\Sigma}$ ▷ Residual evaluation
  end if
end while
$\boldsymbol{U} = \begin{bmatrix}\boldsymbol{X}_1 & \boldsymbol{X}_3 & \cdots\end{bmatrix}\hat{\boldsymbol{U}}$
$\boldsymbol{V} = \begin{bmatrix}\boldsymbol{Y}_2 & \boldsymbol{Y}_4 & \cdots\end{bmatrix}\hat{\boldsymbol{V}}$

Proof. The equalities come from direct calculations, e.g.,

$$\mathbb{E}\|\boldsymbol{S}\boldsymbol{G}\boldsymbol{T}\|_{\text{F}}^2 = \mathbb{E}\,\text{tr}\big(\boldsymbol{S}\boldsymbol{G}\boldsymbol{T}\boldsymbol{T}^*\boldsymbol{G}^*\boldsymbol{S}^*\big) = \mathbb{E}\sum_{ijkl}\boldsymbol{S}_{ij}\boldsymbol{G}_{jk}\boldsymbol{T}_{kl}\boldsymbol{T}_{kl}\boldsymbol{G}_{jk}\boldsymbol{S}_{ij} = \sum_{ij}\boldsymbol{S}_{ij}^2\sum_{kl}\boldsymbol{T}_{kl}^2.$$

We have used the fact that unmatched Gaussians $\boldsymbol{G}_{ab}\boldsymbol{G}_{cd}$ for $(a,b) \neq (c,d)$ vanish in expectation, so it is only necessary to consider the matched Gaussian terms $\boldsymbol{G}_{ab}\boldsymbol{G}_{ab}$.

The inequalities come from a comparison principle for Gaussian random variables. We compare two Gaussian random fields

$$X_{\boldsymbol{u}\boldsymbol{v}} = \langle \boldsymbol{S}^*\boldsymbol{u}, \boldsymbol{G}\boldsymbol{T}\boldsymbol{v}\rangle + \|\boldsymbol{S}^*\boldsymbol{u}\|\|\boldsymbol{T}\boldsymbol{v}\|\gamma,$$
$$Y_{\boldsymbol{u}\boldsymbol{v}} = \|\boldsymbol{S}^*\boldsymbol{u}\|\langle \boldsymbol{h}, \boldsymbol{T}\boldsymbol{v}\rangle + \|\boldsymbol{T}\boldsymbol{v}\|\langle \boldsymbol{g}, \boldsymbol{S}^*\boldsymbol{u}\rangle,$$

Algorithm A.2 `NysBKI` with quality assurance

Input: Psd matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$; block size k; number of eigenvectors r; tolerance ε_{tol}; shift $\varepsilon > 0$

Output: Orthogonal $\boldsymbol{U}$ and diagonal $\boldsymbol{\Lambda}$ such that $\boldsymbol{A} \approx \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^*$

Generate a random matrix $\boldsymbol{Y}_0 \in \mathbb{R}^{N\times k}$
$[\boldsymbol{X}_0, \sim] = \text{qr\_econ}(\boldsymbol{Y}_0)$
$i = 1$
$\boldsymbol{Y}_1 = \boldsymbol{A}\boldsymbol{X}_0$
$\boldsymbol{S}_0 = [\,]$; $\boldsymbol{E} = [\,]$
while $i = 1$ or one of the first r columns of $\boldsymbol{E}$ has norm $> \varepsilon$ **do**
  $\boldsymbol{X}_i = \boldsymbol{Y}_i + \varepsilon\boldsymbol{X}_{i-1}$
  $\boldsymbol{R}_{\bullet i} = \begin{bmatrix}\boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{X}_{i-2} & \boldsymbol{X}_{i-1}\end{bmatrix}^* \boldsymbol{X}_i$ ▷ $\boldsymbol{R}_{\bullet i} = \boldsymbol{X}_{i-1}^*\boldsymbol{X}_i$ if $i = 1$
  $\boldsymbol{X}_i = \boldsymbol{X}_i - \sum_{j<i} \boldsymbol{X}_j(\boldsymbol{X}_j^*\boldsymbol{X}_i)$ ▷ Orthog. w.r.t. past iterates (2×)
  $[\boldsymbol{X}_i, \boldsymbol{R}_{ii}] = \text{qr\_econ}(\boldsymbol{X}_i)$ ▷ Optional: stabilized QR (5.1)
  $\boldsymbol{C} = \text{chol}\left(\begin{bmatrix}\boldsymbol{S}_{i-1} & \boldsymbol{R}_{\bullet i}\end{bmatrix}\right)$
  $\boldsymbol{S}_i = \begin{bmatrix}\boldsymbol{S}_{i-1} & \boldsymbol{R}_{\bullet i}\\ \boldsymbol{0} & \boldsymbol{R}_{ii}\end{bmatrix}$ ▷ $\boldsymbol{A}\begin{bmatrix}\boldsymbol{X}_0 & \cdots & \boldsymbol{X}_{i-1}\end{bmatrix} = \begin{bmatrix}\boldsymbol{X}_0 & \cdots & \boldsymbol{X}_i\end{bmatrix}\boldsymbol{S}_i$
  $\boldsymbol{Z} = \boldsymbol{S}_i\boldsymbol{C}^{-1}$
  $[\hat{\boldsymbol{U}}, \boldsymbol{\Sigma}, \sim] = \text{svd\_econ}(\boldsymbol{Z})$
  $\boldsymbol{U} = \begin{bmatrix}\boldsymbol{X}_0 & \cdots & \boldsymbol{X}_i\end{bmatrix}\hat{\boldsymbol{U}}$
  $\boldsymbol{\Lambda} = \max\{\boldsymbol{0}, \boldsymbol{\Sigma}^2 - \varepsilon\boldsymbol{I}\}$
  $\boldsymbol{Y}_{i+1} = \boldsymbol{A}\boldsymbol{X}_i$
  $\boldsymbol{E} = \begin{bmatrix}\boldsymbol{Y}_1 & \cdots & \boldsymbol{Y}_{i+1}\end{bmatrix}\hat{\boldsymbol{U}} - \boldsymbol{U}\boldsymbol{\Sigma}$ ▷ Residual evaluation
  $i = i + 1$
end while

where $\gamma \in \mathbb{R}$ is a Gaussian random variable and $\boldsymbol{h} \in \mathbb{R}^p$ and $\boldsymbol{g} \in \mathbb{R}^n$ are Gaussian random vectors, independent from one another and independent from $\boldsymbol{G}$. The indices $\boldsymbol{u} \in \mathbb{R}^L$ and $\boldsymbol{v} \in \mathbb{R}^Q$ are unit-length vectors satisfying $\|\boldsymbol{u}\| = \|\boldsymbol{v}\| = 1$. A short calculation verifies that

$$\begin{aligned}\mathbb{E}[X_{\boldsymbol{u}\boldsymbol{v}}X_{\boldsymbol{u}'\boldsymbol{v}'}] &= \langle \boldsymbol{S}^*\boldsymbol{u}, \boldsymbol{S}^*\boldsymbol{u}'\rangle\langle \boldsymbol{T}\boldsymbol{v}, \boldsymbol{T}\boldsymbol{v}'\rangle + \|\boldsymbol{S}^*\boldsymbol{u}\|\|\boldsymbol{S}^*\boldsymbol{u}'\|\|\boldsymbol{T}\boldsymbol{v}\|\|\boldsymbol{T}\boldsymbol{v}'\|,\\ \mathbb{E}[Y_{\boldsymbol{u}\boldsymbol{v}}Y_{\boldsymbol{u}'\boldsymbol{v}'}] &= \|\boldsymbol{S}^*\boldsymbol{u}\|\|\boldsymbol{S}^*\boldsymbol{u}'\|\langle \boldsymbol{T}\boldsymbol{v}, \boldsymbol{T}\boldsymbol{v}'\rangle + \langle \boldsymbol{S}^*\boldsymbol{u}, \boldsymbol{S}^*\boldsymbol{u}'\rangle\|\boldsymbol{T}\boldsymbol{v}\|\|\boldsymbol{T}\boldsymbol{v}'\|.\end{aligned}$$

Hence,

$$\begin{aligned}&\mathbb{E}[X_{\boldsymbol{u}\boldsymbol{v}}X_{\boldsymbol{u}'\boldsymbol{v}'}] - \mathbb{E}[Y_{\boldsymbol{u}\boldsymbol{v}}Y_{\boldsymbol{u}'\boldsymbol{v}'}]\\ &\qquad = (\|\boldsymbol{S}^*\boldsymbol{u}\|\|\boldsymbol{S}^*\boldsymbol{u}'\| - \langle \boldsymbol{S}^*\boldsymbol{u}, \boldsymbol{S}^*\boldsymbol{u}'\rangle)(\|\boldsymbol{T}\boldsymbol{v}\|\|\boldsymbol{T}\boldsymbol{v}'\| - \langle \boldsymbol{T}\boldsymbol{v}, \boldsymbol{T}\boldsymbol{v}'\rangle) \geq 0.\end{aligned}$$

Consequently, the increments of X and Y satisfy

$$\mathbb{E}\,|X_{\boldsymbol{u}\boldsymbol{v}} - X_{\boldsymbol{u}'\boldsymbol{v}'}|^2 \leq \mathbb{E}\,|Y_{\boldsymbol{u}\boldsymbol{v}} - Y_{\boldsymbol{u}'\boldsymbol{v}'}|^2,$$

and Slepian's lemma [105, Sec. 7.2] gives the result

$$\mathbb{E}\,|\max_{\boldsymbol{u},\boldsymbol{v}} X_{\boldsymbol{u}\boldsymbol{v}}|^p \leq \mathbb{E}\,|\max_{\boldsymbol{u},\boldsymbol{v}} Y_{\boldsymbol{u}\boldsymbol{v}}|^p$$

for any $p \geq 1$. Using Jensen's inequality, we find also

$$\mathbb{E}\|\boldsymbol{S}\boldsymbol{G}\boldsymbol{T}\|^p = \mathbb{E}\max_{\boldsymbol{u},\boldsymbol{v}}\left|\boldsymbol{u}^*\boldsymbol{S}\boldsymbol{G}\boldsymbol{T}\boldsymbol{v}\right|^p \leq \mathbb{E}\,|\max_{\boldsymbol{u},\boldsymbol{v}} X_{\boldsymbol{u}\boldsymbol{v}}|^p \leq \mathbb{E}\,|\max_{\boldsymbol{u},\boldsymbol{v}} Y_{\boldsymbol{u}\boldsymbol{v}}|^p.$$

To bound $\mathbb{E}\,|\max_{\boldsymbol{u},\boldsymbol{v}} Y_{\boldsymbol{uv}}|^p$ in the case $p = 2$, we use the triangle inequality and a direct calculation involving second moments of Gaussians to calculate

$$\begin{aligned}\big(\mathbb{E}\,|\max_{\boldsymbol{u},\boldsymbol{v}} Y_{\boldsymbol{uv}}|^2\big)^{1/2} &\le \big(\mathbb{E}\big|\|\boldsymbol{S}\|\|\boldsymbol{T}^*\boldsymbol{h}\| + \|\boldsymbol{T}\|\|\boldsymbol{S}\boldsymbol{g}\|\big|^2\big)^{1/2} \\ &\le \big(\mathbb{E}\big|\|\boldsymbol{S}\|\|\boldsymbol{T}^*\boldsymbol{h}\|\big|^2\big)^{1/2} + \big(\mathbb{E}\big|\|\boldsymbol{T}\|\|\boldsymbol{S}\boldsymbol{g}\|\big|^2\big)^{1/2} \\ &= \|\boldsymbol{S}\|\|\boldsymbol{T}\|_{\mathrm{F}} + \|\boldsymbol{T}\|\|\boldsymbol{S}\|_{\mathrm{F}}.\end{aligned}$$

Similarly, to bound $\mathbb{E}\,|\max_{\boldsymbol{u},\boldsymbol{v}} Y_{\boldsymbol{uv}}|^p$ in the case $p = 4$, we calculate

$$\begin{aligned}\big(\mathbb{E}\,|\max_{\boldsymbol{u},\boldsymbol{v}} Y_{\boldsymbol{uv}}|^4\big)^{1/4} &\le \big(\mathbb{E}\big|\|\boldsymbol{S}\|\|\boldsymbol{T}^*\boldsymbol{h}\|\big|^4\big)^{1/4} + \big(\mathbb{E}\big|\|\boldsymbol{T}\|\|\boldsymbol{S}\boldsymbol{g}\|\big|^4\big)^{1/4} \\ &= \|\boldsymbol{S}\|(\|\boldsymbol{T}\|_{\mathrm{F}}^4 + 2\|\boldsymbol{T}\|_4^4)^{1/4} + \|\boldsymbol{T}\|(\|\boldsymbol{S}\|_{\mathrm{F}}^4 + 2\|\boldsymbol{S}\|_4^4)^{1/4},\end{aligned}$$

where the last equality comes from a direct calculation involving fourth moments of Gaussians. ∎

LEMMA B.2 (Inverse moment bounds). *Consider a Gaussian matrix $\boldsymbol{G} \in \mathbb{R}^{r\times k}$ with $r \le k$. Then, $(\boldsymbol{G}\boldsymbol{G}^*)^{-1}$ satisfies the moment formulas*

$$\begin{aligned}\mathbb{E}(\boldsymbol{G}\boldsymbol{G}^*)^{-1} &= \frac{1}{k-r-1}\mathbf{I}, && r \le k-2 \\ \mathbb{E}\|(\boldsymbol{G}\boldsymbol{G}^*)^{-1}\|_*^2 &= \frac{r^2(k-r) - 2r(r-1)}{(k-r)(k-r-1)(k-r-3)}, && r \le k-4 \\ \mathbb{E}\|(\boldsymbol{G}\boldsymbol{G}^*)^{-1}\|_{\mathrm{F}}^2 &= \frac{r(k-1)}{(k-r)(k-r-1)(k-r-3)}, && r \le k-4,\end{aligned}$$

and the upper bounds

$$\mathbb{E}\big[a : \mathrm{tr}(\boldsymbol{G}\boldsymbol{G}^*)^{-1}\big] \le \frac{r}{2}\log(1+2a), \qquad r = k-1, \tag{B.1}$$

$$\mathbb{E}\big[a : \mathrm{tr}(\boldsymbol{G}\boldsymbol{G}^*)^{-1}\big] \le \sqrt{\frac{\pi a}{2}}, \qquad r = k. \tag{B.2}$$

Last, for any $r \le k$ and $t \ge 0$,

$$\mathbb{P}\left\{\mathrm{tr}(\boldsymbol{G}\boldsymbol{G}^*)^{-1} > \frac{\mathrm{e}tr}{k-r+1}\right\} \le \sqrt{\frac{\pi r}{k-r+1}}\,t^{-(k-r+1)/2}. \tag{B.3}$$

Proof. Explicit formulas for $\mathbb{E}(\boldsymbol{G}\boldsymbol{G}^*)^{-1}$, $\mathbb{E}\|(\boldsymbol{G}\boldsymbol{G}^*)^{-1}\|_*^2$, and $\mathbb{E}\|(\boldsymbol{G}\boldsymbol{G}^*)^{-1}\|_{\mathrm{F}}^2$ are given in [86, pg. 119]. The remaining formulas are new, but the expectation bounds are related to the computations in [99].

To establish these formulas, we begin by calculating

$$\mathbb{E}\big[a : \mathrm{tr}(\boldsymbol{G}\boldsymbol{G}^*)^{-1}\big] = \mathbb{E}\left[a : \sum_{i=1}^r (\boldsymbol{G}\boldsymbol{G}^*)_{ii}^{-1}\right] \le \sum_{i=1}^r \mathbb{E}\left[a : (\boldsymbol{G}\boldsymbol{G}^*)_{ii}^{-1}\right],$$

where we have used the subadditivity of $x \mapsto a : x$ and the linearity of expectation. The inverse diagonal elements $1/(\boldsymbol{G}\boldsymbol{G}^*)_{ii}^{-1}$ for $1 \le i \le r$ have chi-squared distributions with $p = k - r + 1$ degrees of freedom [86, pg. 119]. Hence, their density function is

$$f(x) = \frac{1}{2^{p/2}\Gamma(p/2)} x^{p/2-1}\mathrm{e}^{-x/2}, \qquad x > 0.$$

In the case $p = 2$, this allows us to evaluate

$$\mathbb{E}\left[a : (\boldsymbol{G}\boldsymbol{G}^*)_{ii}^{-1}\right] = \frac{1}{2}\int_0^\infty \frac{\mathrm{e}^{-x/2}}{a^{-1}+x}\,dx = \int_{1/(2a)}^\infty \frac{\mathrm{e}^{1/(2a)-y}}{y}\,dy \le \frac{1}{2}\log(1+2a),$$

where we have substituted $y = x/2 + 1/(2a)$ and applied the bound [3, Sec. 5.1.20]

$$\int_x^\infty \frac{\mathrm{e}^{-t}}{t}\,dt \le \log\left(1+\frac{1}{x}\right)\mathrm{e}^{-x}, \qquad x > 0.$$

In the case $p = 1$, we evaluate

$$\mathbb{E}\left[a : (\boldsymbol{G}\boldsymbol{G}^*)_{ii}^{-1}\right] \le \frac{1}{2\sqrt{\pi}}\int_0^\infty \frac{x^{-1/2}}{a^{-1}+x}\,dx = \sqrt{\frac{a}{\pi}}\int_0^\infty \frac{1}{1+y^2}\,dy = \sqrt{\frac{\pi a}{2}},$$

where we have substituted $x = a^{-1}y^2$ and used the fact that $1/(1+y^2)$ is the derivative of $\arctan y$. Thus we establish the bounds (B.1) and (B.2).

Last, to establish the concentration inequality (B.3), we set $u = \mathrm{e}tr/p$ and argue using Markov's inequality

$$\mathbb{P}\big\{\mathrm{tr}(\boldsymbol{G}\boldsymbol{G}^*)^{-1} > u\big\} = \mathbb{P}\big\{u^p : \|(\boldsymbol{G}\boldsymbol{G}^*)^{-1}\|_*^p > u^p : u^p\big\} \le \frac{\mathbb{E}\big[u^p : \|(\boldsymbol{G}\boldsymbol{G}^*)^{-1}\|_*^p\big]}{u^p : u^p}.$$

We directly evaluate $u^p : u^p = \frac{1}{2}u^p$, and we apply the monotonicity and subadditivity of $x \mapsto u^p : x$ to obtain the upper bound

$$\begin{aligned} u^p : \|(\boldsymbol{G}\boldsymbol{G}^*)^{-1}\|_*^p &= u^p : \left(\sum_{i=1}^r (\boldsymbol{G}\boldsymbol{G}^*)_{ii}^{-1}\right)^p \le u^p : r^{p-1}\sum_{i=1}^r \big((\boldsymbol{G}\boldsymbol{G}^*)_{ii}^{-1}\big)^p \\ &\le \sum_{i=1}^r u^p : r^{p-1}\big((\boldsymbol{G}\boldsymbol{G}^*)_{ii}^{-1}\big)^p = r^{p-1}\sum_{i=1}^r \frac{u^p}{r^{p-1}} : \big((\boldsymbol{G}\boldsymbol{G}^*)_{ii}^{-1}\big)^p. \end{aligned}$$

Hence, we have shown

$$\mathbb{P}\big\{\mathrm{tr}(\boldsymbol{G}\boldsymbol{G}^*)^{-1} > u\big\} \le \frac{2r^{p-1}}{u^p}\sum_{i=1}^r \mathbb{E}\left[\frac{u^p}{r^{p-1}} : \big((\boldsymbol{G}\boldsymbol{G}^*)_{ii}^{-1}\big)^p\right],$$

Last, we set $a = u^p/r^{p-1}$ and for $1 \le i \le r$ we evaluate

$$\begin{aligned} \mathbb{E}\left[a : \big((\boldsymbol{G}\boldsymbol{G}^*)_{ii}^{-1}\big)^p\right] &\le \frac{1}{2^{p/2}\Gamma(p/2)}\int_0^\infty \frac{x^{p/2-1}}{a^{-1}+x^p}\,dx \\ &= \frac{\pi\sqrt{a}}{2^{p/2+1}\Gamma(p/2+1)} \le \frac{1}{2}\sqrt{\frac{\pi a}{p}}\left(\frac{\mathrm{e}}{p}\right)^{p/2}, \end{aligned}$$

where the last inequality follows from Stirling's approximation [3, Sec. 6.1.38]. We obtain

$$\mathbb{P}\big\{\mathrm{tr}(\boldsymbol{G}\boldsymbol{G}^*)^{-1} > u\big\} \le \sqrt{\frac{\pi r}{p}}\left(\frac{\mathrm{e}r}{up}\right)^{p/2},$$

and substituting $u = \mathrm{e}tr/p$ completes the proof. ∎

LEMMA B.3 (Inverse moment bounds, spectral norm). *For $0 \le p \le 18$ and $k \ge r + 2m$, the Gaussian matrix $\boldsymbol{G} \in \mathbb{R}^{r \times k}$ satisfies*

$$\left(\mathbb{E}\|(\boldsymbol{G}\boldsymbol{G}^*)^{-1}\|^p\right)^{1/p} \le \frac{\mathrm{e}^2(k+r)}{2(k-r)^2}. \tag{B.4}$$

Proof. Our starting point is a calculation due to Edelman [36, Prop. 5.1] that bounds the density $f_{\lambda_{\min}}$ for $\lambda_{\min}(\boldsymbol{G}\boldsymbol{G}^*)$ as

$$f_{\lambda_{\min}}(\lambda) \le \frac{\sqrt{\pi}\,\Gamma\left(\frac{k+1}{2}\right)}{2^{\frac{k-r+1}{2}}\,\Gamma\left(\frac{r}{2}\right)\Gamma\left(\frac{k-r+1}{2}\right)\Gamma\left(\frac{k-r+2}{2}\right)}\lambda^{\frac{k-r-1}{2}}\mathrm{e}^{-\frac{\lambda}{2}}, \qquad \lambda > 0.$$

Applying the Legendre duplication formula $\Gamma(z)\Gamma\left(z+\frac{1}{2}\right) = 2^{1-2z}\sqrt{\pi}\Gamma(2z)$, we obtain

$$f_{\lambda_{\min}}(\lambda) \le \frac{2^{\frac{k-r-1}{2}}\,\Gamma\left(\frac{k+1}{2}\right)}{\Gamma\left(\frac{r}{2}\right)\Gamma\left(k-r+1\right)}\lambda^{\frac{k-r-1}{2}}\mathrm{e}^{-\frac{\lambda}{2}}, \qquad \lambda > 0. \tag{B.5}$$

We can simplify (B.5) further by neglecting the $\mathrm{e}^{-\lambda/2}$ factor and noting

$$\frac{2^{\frac{k-r+1}{2}}\,\Gamma\left(\frac{k+1}{2}\right)}{\Gamma\left(\frac{r}{2}\right)} \le 2^{\frac{k-r+1}{2}}\prod_{i=0}^{k-r}\sqrt{\frac{r+i}{2}} = \prod_{i=0}^{k-r}\sqrt{r+i} \le \left(\frac{k+r}{2}\right)^{\frac{k-r+1}{2}},$$

where we have used $\Gamma\left(z+\frac{1}{2}\right) \le z^{1/2}\Gamma(z)$ [59] and $(r+i)(k-i) \le \left(\frac{k+r}{2}\right)^2$. We find

$$f_{\lambda_{\min}}(\lambda) \le \frac{\lambda^{\frac{k-r-1}{2}}}{2\Gamma(k-r+1)}\left(\frac{k+r}{2}\right)^{\frac{k-r+1}{2}}, \qquad \lambda > 0. \tag{B.6}$$

Integrating (B.6) with respect to λ gives the probability bound

$$P\{\lambda_{\min}(\boldsymbol{G}\boldsymbol{G}^*) \le t\} \le \frac{1}{\Gamma(k-r+2)}\left(\frac{t(k+r)}{2}\right)^{\frac{k-r+1}{2}}, \qquad t > 0. \tag{B.7}$$

which is nonvacuous for

$$t \le t_* = \Gamma(k-r+2)^{\frac{2}{k-r+1}}\left(\frac{2}{k+r}\right).$$

By an explicit integration of (B.7), we obtain the moment bound

$$\begin{aligned}
\mathbb{E}\|(\boldsymbol{G}\boldsymbol{G}^*)^{-1}\|^p &= p\int_0^\infty s^{p-1}P\{\lambda_{\min}(\boldsymbol{G}\boldsymbol{G}^*) \le s^{-1}\}\,ds \\
&\le t_*^{-p} + p\int_{t_*}^\infty s^{p-1}P\{\lambda_{\min}(\boldsymbol{G}\boldsymbol{G}^*) \le s^{-1}\}\,ds \\
&\le \left(1 + \frac{2p}{k-r+1-2p}\right)t_*^{-p} \\
&= \left(1 + \frac{2p}{k-r+1-2p}\right)\left(\frac{1}{\Gamma(k-r+2)}\right)^{\frac{2p}{k-r+1}}\left(\frac{k+r}{2}\right)^p.
\end{aligned}$$

The theorem is complete as soon as we can verify the numerical identity

$$\left(1 + \frac{2p}{x+1-2p}\right)\left(\frac{1}{\Gamma(x+2)}\right)^{\frac{2p}{x+1}} \le \left(\frac{\mathrm{e}}{x}\right)^{2p}, \qquad 0 \le p \le 18, \quad x \ge 2p. \tag{B.8}$$

To establish (B.8), we briefly argue that

$$x \mapsto \mathrm{e}\,\Gamma(x+2)^{\frac{1}{x+1}}\Big(1-\frac{2p}{x+1}\Big)^{2p} - x \tag{B.9}$$

is non-decreasing for $x \geq 2p$, which can be shown by explicitly examining its derivative and applying

$$\mathrm{e}\,\Gamma(x+2)^{\frac{1}{x+1}} \geq x+1, \qquad \frac{d}{dx}\Big(\mathrm{e}\,\Gamma(x+2)^{\frac{1}{x+1}}\Big) \geq 1.$$

Next, using the fact that (B.9) is non-decreasing and applying Stirling's approximation [3, Sec. 6.1.38], we calculate the lower bound

$$\begin{aligned}\mathrm{e}\,\Gamma(x+2)^{\frac{1}{x+1}}\Big(1-\frac{2p}{x+1}\Big)^{2p} - x &\geq \mathrm{e}\,\Gamma(2p+2)^{\frac{1}{2p+1}}\Big(1-\frac{2p}{2p+1}\Big)^{2p} - 2p \\ &\geq \left(\frac{2\pi}{(2p+1)^{1+\frac{1}{p}}}\right)^{\frac{1}{4p+2}}(2p+1) - 2p,\end{aligned}$$

The right-hand side is non-negative provided that

$$\frac{2\pi}{(2p+1)^{1+\frac{1}{p}}} \geq \mathrm{e}^{-2},$$

which holds for any $p \geq 18$. We conclude that

$$\mathrm{e}\,\Gamma(x+2)^{\frac{1}{x+1}}\Big(1-\frac{2p}{x+1}\Big)^{2p} - x \geq 0, \qquad 0 \leq p \leq 18, \quad x \geq 2p,$$

which is equivalent to (B.8), thus completing the proof. ∎

B.1. Proof of Proposition 8.6. When we condition on $\boldsymbol{H}$, the mapping $\boldsymbol{G} \mapsto \|\boldsymbol{S}\boldsymbol{G}\boldsymbol{H}^\dagger\|_{\mathrm{F}}$ is Lipschitz with Lipschitz constant $L = \|\boldsymbol{S}\|_{\mathrm{F}}\|\boldsymbol{H}^\dagger\|_{\mathrm{F}}$, and the conditional expectation satisfies

$$\mathbb{E}\big[\|\boldsymbol{S}\boldsymbol{G}\boldsymbol{H}^\dagger\|_{\mathrm{F}} \,|\, \boldsymbol{H}\big] \leq \mathbb{E}\big[\|\boldsymbol{S}\boldsymbol{G}\boldsymbol{H}^\dagger\|_{\mathrm{F}}^2 \,|\, \boldsymbol{H}\big]^{1/2} = \|\boldsymbol{S}\|_{\mathrm{F}}\|\boldsymbol{H}^\dagger\|_{\mathrm{F}}.$$

Using a concentration inequality for Lipschitz functions of Gaussian random fields [16, Thm. 4.5.7], we find for any $u \geq 1$,

$$\mathbb{P}\big\{\|\boldsymbol{S}\boldsymbol{G}\boldsymbol{H}^\dagger\|_{\mathrm{F}} > \sqrt{u}\|\boldsymbol{S}\|_{\mathrm{F}}\|\boldsymbol{H}^\dagger\|_{\mathrm{F}} \mid \boldsymbol{H}\big\} \leq \mathrm{e}^{-(\sqrt{u}-1)^2/2},$$

and the right-hand side is bounded by $\mathrm{e}^{-(u-2)/4}$ for any $u \geq 0$. Hence, we obtain the conditional probability bound

$$\mathbb{P}\left\{\|\boldsymbol{S}\boldsymbol{G}\boldsymbol{H}^\dagger\|_{\mathrm{F}}^2 > \frac{utr}{k-r+1}\|\boldsymbol{S}\|_{\mathrm{F}}^2 \,\middle|\, \|\boldsymbol{H}^\dagger\|_{\mathrm{F}}^2 \leq \frac{tr}{k-r+1}\right\} \leq \mathrm{e}^{-(u-2)/4}, \tag{B.10}$$

while Lemma B.2 implies the probability bound

$$\mathbb{P}\left\{\|\boldsymbol{H}^\dagger\|_{\mathrm{F}}^2 > \frac{tr}{k-r+1}\right\} \leq \sqrt{\pi r}\left(\frac{t}{\mathrm{e}}\right)^{-(k-r+1)/2}. \tag{B.11}$$

Combining (B.10) and (B.11) establishes the probability bound (8.9) for $\|\boldsymbol{SGH}^\dagger\|_{\mathrm{F}}^2$. Additionally, Lemmas B.1 and B.2 allow us to verify the expectation bound (8.10) for $\|\boldsymbol{SGH}^\dagger\|_{\mathrm{F}}^2$, namely,

$$\begin{aligned}\mathbb{E}\|\boldsymbol{SGH}^\dagger\|_{\mathrm{F}}^2 &= \mathbb{E}\,\mathbb{E}\big[\|\boldsymbol{SGH}^\dagger\|_{\mathrm{F}}^2 \mid \boldsymbol{H}^\dagger\big] = \|\boldsymbol{S}\|_{\mathrm{F}}^2\,\mathbb{E}\|\boldsymbol{H}^\dagger\|_{\mathrm{F}}^2 \\ &= \|\boldsymbol{S}\|_{\mathrm{F}}^2\,\mathbb{E}\,\mathrm{tr}(\boldsymbol{H}^\dagger(\boldsymbol{H}^\dagger)^*) = \|\boldsymbol{S}\|_{\mathrm{F}}^2\mathrm{tr}(\mathbb{E}(\boldsymbol{HH}^*)^{-1}) \\ &= \|\boldsymbol{S}\|_{\mathrm{F}}^2\,\frac{r}{k-r-1}.\end{aligned}$$

In the above display, we have applied conditioning and then exchanged the expectation and the trace.

B.2. Proof of Theorem 8.7. Using Lemma 8.3, we assume $\boldsymbol{A}$ is diagonal and psd, with non-increasing diagonal entries. Then, we partition the matrices as

$$\boldsymbol{A} = \begin{bmatrix}\boldsymbol{A}_1 & \\ & \boldsymbol{A}_2\end{bmatrix}, \qquad \boldsymbol{\Omega} = \begin{bmatrix}\boldsymbol{\Omega}_1 \\ \boldsymbol{\Omega}_2\end{bmatrix},$$

so that $\boldsymbol{A}_1$ is a $r \times r$ matrix and $\boldsymbol{\Omega}_1$ is a $r \times k$ matrix. We will systematically derive expectation bounds in the Frobenius norm, spectral norm, and Schatten-4 norm.

To begin, Proposition 8.5 guarantees the error bound

$$\mathbb{E}\|\boldsymbol{A} - \boldsymbol{\Pi}_{\boldsymbol{A\Omega}}\boldsymbol{A}\|_{\mathrm{F}}^2 \le \|\boldsymbol{A}_2\|_{\mathrm{F}}^2 + \mathbb{E}\big[\|\boldsymbol{A}_1\|_{\mathrm{F}}^2 : \|\boldsymbol{A}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\|_{\mathrm{F}}^2\big].$$

Using the concavity of the parallel sum (Lemma 8.4 part 3) and Lemma B.1, we argue

$$\begin{aligned}\mathbb{E}[\|\boldsymbol{A}_1\|_{\mathrm{F}}^2 : \|\boldsymbol{A}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\|_{\mathrm{F}}^2] &= \mathbb{E}\big[\mathbb{E}\big[\|\boldsymbol{A}_1\|_{\mathrm{F}}^2 : \|\boldsymbol{A}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\|_{\mathrm{F}}^2 \,\big|\, \boldsymbol{\Omega}_1\big]\big] \\ &\le \mathbb{E}\big[\|\boldsymbol{A}_1\|_{\mathrm{F}}^2 : \mathbb{E}\big[\|\boldsymbol{A}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\|_{\mathrm{F}}^2 \,\big|, \boldsymbol{\Omega}_1\big]\big] \\ &= \mathbb{E}\big[\|\boldsymbol{A}_1\|_{\mathrm{F}}^2 : \|\boldsymbol{A}_2\|_{\mathrm{F}}^2\|\boldsymbol{\Omega}_1^\dagger\|_{\mathrm{F}}^2\big] \\ &= \|\boldsymbol{A}_2\|_{\mathrm{F}}^2 \cdot \mathbb{E}\left[\frac{\|\boldsymbol{A}_1\|_{\mathrm{F}}^2}{\|\boldsymbol{A}_2\|_{\mathrm{F}}^2} : \|\boldsymbol{\Omega}_1^\dagger\|_{\mathrm{F}}^2\right].\end{aligned}$$

Using Lemma B.2, we bound the last quantity as

$$\mathbb{E}\left[\frac{\|\boldsymbol{A}_1\|_{\mathrm{F}}^2}{\|\boldsymbol{A}_2\|_{\mathrm{F}}^2} : \|\boldsymbol{\Omega}_1^\dagger\|_{\mathrm{F}}^2\right] \le \begin{cases}\dfrac{r}{k-r-1}, & k \ge r+2, \\ r\log\left(1 + \dfrac{\|\boldsymbol{A}_1\|_{\mathrm{F}}^2}{\|\boldsymbol{A}_2\|_{\mathrm{F}}^2}\right), & k = r+1, \\ r\left(\dfrac{\pi\|\boldsymbol{A}_1\|_{\mathrm{F}}^2}{2\|\boldsymbol{A}_2\|_{\mathrm{F}}^2}\right)^{1/2}, & k = r,\end{cases}$$

which confirms the Frobenius–norm error bounds for `RSVD`.

Next, working in the spectral norm, Proposition 8.5 guarantees

$$\mathbb{E}\|\boldsymbol{A} - \boldsymbol{\Pi}_{\boldsymbol{A\Omega}}\boldsymbol{A}\|^2 \le \|\boldsymbol{A}_2\|^2 + \mathbb{E}\|\boldsymbol{A}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\|^2$$

We use Lemmas B.1 to B.3 to calculate

$$\begin{aligned}
\big(\mathbb{E}\|\boldsymbol{A}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\|^2\,\mathbb{E}\big)^{1/2} &= \big(\mathbb{E}\big[\mathbb{E}\big[\|\boldsymbol{A}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\|^2 \,\big|\, \boldsymbol{\Omega}_1\big]\big]\big)^{1/2} \\
&= \big(\mathbb{E}\big|\|\boldsymbol{A}_2\|\|\boldsymbol{\Omega}_1^\dagger\|_{\mathrm{F}} + \|\boldsymbol{\Omega}_1^\dagger\|\|\boldsymbol{A}_2\|_{\mathrm{F}}|^2\big)^{1/2} \\
&\le \|\boldsymbol{A}_2\|\big(\mathbb{E}\|\boldsymbol{\Omega}_1^\dagger\|_{\mathrm{F}}^2\big)^{1/2} + \big(\mathbb{E}\|\boldsymbol{\Omega}_1^\dagger\|^2\big)^{1/2}\|\boldsymbol{A}_2\|_{\mathrm{F}} \\
&\le \left(\frac{r}{k-r-1}\right)^{1/2}\|\boldsymbol{A}_2\| + \left(\frac{\mathrm{e}^2(k+r)}{2(k-r)^2}\right)^{1/2}\|\boldsymbol{A}_2\|_{\mathrm{F}},
\end{aligned}$$

whence

$$\begin{aligned}
\mathbb{E}\|\boldsymbol{A} - \boldsymbol{\Pi}_{\boldsymbol{A}\boldsymbol{\Omega}}\boldsymbol{A}\|^2 &\le \|\boldsymbol{A}_2\|^2 + \frac{2r}{k-r-1}\|\boldsymbol{A}_2\|^2 + \frac{\mathrm{e}^2(k+r)}{(k-r)^2}\|\boldsymbol{A}_2\|_{\mathrm{F}}^2 \\
&\le \left(1 + \frac{2r}{k-r-1}\right)\left(\|\boldsymbol{A}_2\|^2 + \frac{\mathrm{e}^2}{k-r}\|\boldsymbol{A}_2\|_{\mathrm{F}}^2\right),
\end{aligned}$$

which confirms the spectral–norm error bound for `RSVD`.

Last, to obtain fourth moment expectation bounds, we apply Proposition 8.6 and the triangle inequality to yield

$$\begin{aligned}
\big(\mathbb{E}\|\boldsymbol{A} - \boldsymbol{\Pi}_{\boldsymbol{A}\boldsymbol{\Omega}}\boldsymbol{A}\|_p^4\big)^{1/2} &\le \big(\mathbb{E}\big|\|\boldsymbol{A}_2\|_p^2 + \|\boldsymbol{A}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\|_p^2\big|^2\big)^{1/2} \\
&\le \|\boldsymbol{A}_2\|_p^2 + \big(\mathbb{E}\|\boldsymbol{A}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\|_p^4\big)^{1/2}
\end{aligned}$$

and we proceed to bound $\big(\mathbb{E}\|\boldsymbol{A}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\|_p^4\big)^{1/2}$ for $p = 4, \infty$. We use Lemmas B.1 and B.2 to calculate

$$\begin{aligned}
\big(\mathbb{E}\|\boldsymbol{A}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\|_4^4\big)^{1/2} &= \big(\mathbb{E}\big[\mathbb{E}\big[\|\boldsymbol{A}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\|_4^4 \,\big|\, \boldsymbol{\Omega}_1\big]\big]\big)^{1/2} \\
&= \big(\mathbb{E}\big[\|\boldsymbol{A}_2\|_4^4\|\boldsymbol{\Omega}_1^\dagger\|_{\mathrm{F}}^4 + \|\boldsymbol{A}_2\|_4^4\|\boldsymbol{\Omega}_1^\dagger\|_4^4 + \|\boldsymbol{A}_2\|_{\mathrm{F}}^4\|\boldsymbol{\Omega}_1^\dagger\|_4^4\big]\big)^{1/2} \\
&\le \big(\mathbb{E}\big[\|\boldsymbol{\Omega}_1^\dagger\|_{\mathrm{F}}^4 + \|\boldsymbol{\Omega}_1^\dagger\|_4^4\big]\big)^{1/2}\|\boldsymbol{A}_2\|_4^2 + \big(\mathbb{E}\|\boldsymbol{\Omega}_1^\dagger\|_4^4\big)^{1/2}\|\boldsymbol{A}_2\|_{\mathrm{F}}^2 \\
&\le \frac{r+1}{k-r-3}\|\boldsymbol{A}_2\|_4^2 + \left(\frac{r(k-1)}{(k-r)(k-r-1)(k-r-3)}\right)^{1/2}\|\boldsymbol{A}_2\|_{\mathrm{F}}^2 \\
&\le \frac{r+1}{k-r-3}\|\boldsymbol{A}_2\|_4^2 + \frac{k-2}{(k-r-3)\sqrt{k-r}}\|\boldsymbol{A}_2\|_{\mathrm{F}}^2,
\end{aligned}$$

where the last line follows because $r \le k-4$. Thus, we confirm the Schatten-4 norm error bound for `RSVD`. We use Lemmas B.1 to B.3 to calculate

$$\begin{aligned}
&\big(\mathbb{E}\|\boldsymbol{A}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\|^4\big)^{1/4} \\
&= \big(\mathbb{E}\big[\mathbb{E}\big[\|\boldsymbol{A}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\|^4 \,\big|\, \boldsymbol{\Omega}_1\big]\big]\big)^{1/4} \\
&\le \big(\mathbb{E}\big|\|\boldsymbol{A}_2\|(\|\boldsymbol{\Omega}_1^\dagger\|_{\mathrm{F}}^4 + 2\|\boldsymbol{\Omega}_1^\dagger\|_4^4)^{1/4} + \|\boldsymbol{\Omega}_1^\dagger\|(\|\boldsymbol{A}_2\|_{\mathrm{F}}^4 + 2\|\boldsymbol{A}_2\|_4^4)^{1/4}\big|^4\big)^{1/4} \\
&\le \big(\mathbb{E}\big[\|\boldsymbol{\Omega}_1^\dagger\|_{\mathrm{F}}^4 + 2\|\boldsymbol{\Omega}_1^\dagger\|_4^4\big]\big)^{1/4}\|\boldsymbol{A}_2\| + 3^{1/4}\big(\mathbb{E}\|\boldsymbol{\Omega}_1^\dagger\|^4\big)^{1/4}\|\boldsymbol{A}_2\|_{\mathrm{F}} \\
&\le \left(\frac{r+2}{k-r-3}\right)^{1/2}\|\boldsymbol{A}_2\| + \left(\frac{\sqrt{3}\mathrm{e}^2(k+r)}{2(k-r)^2}\right)^{1/2}\|\boldsymbol{A}_2\|_{\mathrm{F}},
\end{aligned}$$

whence

$$\big(\mathbb{E}\|\boldsymbol{A}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\|^4\big)^{1/2} \le \frac{2(r+2)}{k-r-3}\|\boldsymbol{A}_2\|^2 + \frac{\sqrt{3}\mathrm{e}^2(k+r)}{(k-r)^2}\|\boldsymbol{A}_2\|_{\mathrm{F}}^2,$$

which confirms the last error bound for RSVD and completes the proof of Theorem 8.7.

Appendix C. Proof of Lemma 9.3. To confirm part 3 of Lemma 9.3, we first observe

$$2r = \int_0^r 2\,ds \le \int_0^r \frac{2\,ds}{1-s^2} = \log\left(\frac{1+r}{1-r}\right), \tag{C.1}$$

and by exponentiating both sides we obtain $\mathrm{e}^{2r} \le (1+r)/(1-r)$. Next, for $x \ge 1$, we express the Chebyshev polynomial $T_q(x)$ as

$$2T_q(x) = 2\cosh(\log(x + \sqrt{x^2-1})) = \big(x + \sqrt{x^2-1}\big)^q + \big(x + \sqrt{x^2-1}\big)^{-q}, \tag{C.2}$$

where we have used the logarithmic representation $\operatorname{arcosh} x = \log(x + \sqrt{x^2-1})$ [3, Sec. 4.6.21]. By substituting $x = (1+r)/(1-r)$ into (C.2), we establish

$$\mathrm{e}^{2q\sqrt{r}} \le \left(\frac{1+\sqrt{r}}{1-\sqrt{r}}\right)^q \le \left(\frac{1+\sqrt{r}}{1-\sqrt{r}}\right)^q + \left(\frac{1+\sqrt{r}}{1-\sqrt{r}}\right)^{-q} = 2T_q\left(\frac{1+r}{1-r}\right),$$

which gives the stated lower bound for the Chebyshev polynomial T_q.

To confirm part 4 of Lemma 9.3, we write

$$T_q^{-1}(x) = \cosh\big(\tfrac{1}{q}\operatorname{arcosh} x\big) \tag{C.3}$$

and apply the inequalities $\cosh(x) \le \exp(x^2/2)$ [3] and $\operatorname{arcosh} x \le \log(2x)$, which are both valid for $x \ge 1$.

Last, to confirm part 5 of Lemma 9.3, set $\phi(x) = xT_q(x^p)^2$ for $p = 1/2$ or $p = 1/4$. A direct calculation shows the derivative $\phi'(x)$ is increasing for $x \ge 1$ and attains its largest value at the right endpoint of the interval $0 \le x \le 1$, which suffices to show

$$\phi(y) \ge \phi(x) + \phi'(x)(y-x)$$

for any $x \ge 1$ and any $y \ge 0$.

REFERENCES

[1] G. Abraham and M. Inouye, *Fast principal component analysis of large-scale genome-wide data*, PLOS ONE, 9 (2014), pp. 1–5, https://doi.org/10.1371/journal.pone.0093766.

[2] G. Abraham, Y. Qiu, and M. Inouye, *FlashPCA2: Principal component analysis of Biobank-scale genotype datasets*. https://github.com/gabraham/flashpca, 2017.

[3] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, National Bureau of Standards, 1972.

[4] W. Anderson and R. Duffin, *Series and parallel addition of matrices*, Journal of Mathematical Analysis and Applications, 26 (1969), pp. 576–594, https://doi.org/10.1016/0022-247X(69)90200-5.

[5] T. Ando, *Majorizations and inequalities in matrix theory*, Linear Algebra and its Applications, 199 (1994), pp. 17–67, https://doi.org/10.1016/0024-3795(94)90341-7.

[6] A. Bakshi, N. Chepurko, and D. P. Woodruff, *Robust and sample optimal algorithms for PSD low rank approximation*, in IEEE 61st Annual Symposium on Foundations of Computer Science, 2020, https://doi.org/10.1109/FOCS46700.2020.00054.

[7] A. Bakshi, K. L. Clarkson, and D. P. Woodruff, *Low-rank approximation with* $1/\epsilon^{1/3}$ *matrix-vector products*, in Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing, 2022, https://doi.org/10.1145/3519935.3519988.

[8] A. Bakshi and S. Narayanan, *Krylov methods are (nearly) optimal for low-rank approximation*, arXiv:2304.03191, (2023), https://arxiv.org/abs/2304.03191.

[9] O. Balabanov and L. Grigori, *Randomized gram–schmidt process with application to gmres*, SIAM Journal on Scientific Computing, 44 (2022), pp. A1450–A1474, https://doi.org/10.1137/20M138870X.

[10] O. Balabanov and L. Grigori, *Randomized block Gram-Schmidt process for solution of linear systems and eigenvalue problems*, arXiv:2111.14641, (2023), https://arxiv.org/abs/2111.14641.

[11] G. Berkooz, P. Holmes, and J. L. Lumley, *The proper orthogonal decomposition in the analysis of turbulent flows*, Annual Review of Fluid Mechanics, 25 (1993), pp. 539–575, https://doi.org/10.1146/annurev.fl.25.010193.002543.

[12] R. Bhatia, *Matrix Analysis*, Springer, 1997, https://doi.org/10.1007/978-1-4612-0653-8.

[13] R. Bhatia, *Positive Definite Matrices*, Princeton University Press, 2007, https://doi.org/10.1515/9781400827787.

[14] E. K. Bjarkason, *Pass-efficient randomized algorithms for low-rank matrix approximation using any number of views*, SIAM Journal on Scientific Computing, 41 (2019), pp. A2355–A2383, https://doi.org/10.1137/18M118966X.

[15] E. K. Bjarkason, O. J. Maclaren, J. P. O'Sullivan, and M. J. O'Sullivan, *Randomized truncated SVD Levenberg-Marquardt approach to geothermal natural state and history matching*, Water Resources Research, 54 (2018), pp. 2376–2404, https://doi.org/10.1002/2017WR021870.

[16] V. Bogachev, *Gaussian Measures*, American Mathematical Society, 1998, https://doi.org/10.1090/surv/062.

[17] A. Bose, V. Kalantzis, E.-M. Kontopoulou, M. Elkady, P. Paschou, and P. Drineas, *TeraPCA: A fast and scalable software package to study genetic variation in tera-scale genotypes*, Bioinformatics, 35 (2019), pp. 3679–3683, https://doi.org/10.1093/bioinformatics/btz157.

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

[19] N. Bousserez, J. J. Guerrette, and D. K. Henze, *Enhanced parallelization of the incremental 4D-Var data assimilation algorithm using the randomized incremental optimal technique*, Quarterly Journal of the Royal Meteorological Society, 146 (2020), pp. 1351–1371, https://doi.org/10.1002/qj.3740.

[20] C. Boutsidis, A. Gittens, and P. Kambadur, *Spectral clustering via the power method - Provably*, in Proceedings of the 32nd International Conference on Machine Learning, 2015, https://dl.acm.org/doi/10.5555/3045118.3045124.

[21] C. Boutsidis, M. W. Mahoney, and P. Drineas, *An improved approximation algorithm for the column subset selection problem*, in Proceedings of the 2009 Annual ACM-SIAM Symposium on Discrete Algorithms, 2009, https://doi.org/10.1137/1.9781611973068.105.

[22] T. Bui-Thanh, C. Burstedde, O. Ghattas, J. Martin, G. Stadler, and L. C. Wilcox, *Extreme-scale UQ for Bayesian inverse problems governed by PDEs*, in Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, 2012, https://doi.org/10.1109/SC.2012.56.

[23] C.-Y. Chen, S. Pollack, D. J. Hunter, J. N. Hirschhorn, P. Kraft, and A. L. Price, *Improved ancestry inference using weights from external reference panels*, Bioinformatics, 29 (2013), pp. 1399–1406, https://doi.org/10.1093/bioinformatics/btt144.

[24] Y. Chen, E. N. Epperly, J. A. Tropp, and R. J. Webber, *Randomly pivoted Cholesky: Practical approximation of a kernel matrix with few entry evaluations*, arXiv:2207.06503, (2023), https://arxiv.org/abs/2207.06503.

[25] J. Cheng and M. D. Sacchi, *Fast dual-domain reduced-rank algorithm for 3D deblending via randomized QR decomposition*, Geophysics, 81 (2016), pp. V89–V101, https://doi.org/10.1190/geo2015-0292.1.

[26] K. L. Clarkson and D. P. Woodruff, *Numerical linear algebra in the streaming model*, in Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, 2009, https://doi.org/10.1145/1536414.1536445.

[27] A. K. Cline, C. B. Moler, G. W. Stewart, and J. H. Wilkinson, *An estimate for the condition number of a matrix*, SIAM Journal on Numerical Analysis, 16 (1979), pp. 368–375, http://www.jstor.org/stable/2156842 (accessed 2023-06-12).

[28] M. B. Cohen, *Nearly tight oblivious subspace embeddings by trace inequalities*, in Proceedings of the 2016 Annual ACM-SIAM Symposium on Discrete Algorithms, 2016, https://doi.org/10.1137/1.9781611974331.ch21.

[29] P. G. Constantine, *Active Subspaces*, Society for Industrial and Applied Mathematics, 2015, https://doi.org/10.1137/1.9781611973860.

[30] A. Deshpande and S. Vempala, *Adaptive sampling and fast low-rank matrix approximation*, in Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 2006, https://doi.org/10.1007/11830924_28.

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

[32] P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay, *Clustering in large graphs and matrices*, in Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, 1999, https://dl.acm.org/doi/10.5555/314500.314576.

[33] P. Drineas, I. C. F. Ipsen, E.-M. Kontopoulou, and M. Magdon-Ismail, *Structural convergence results for approximation of dominant subspaces from block Krylov spaces*, SIAM Journal on Matrix Analysis and Applications, 39 (2018), pp. 567–586, https://doi.org/10.1137/16M1091745.

[34] P. Drineas, R. Kannan, and M. W. Mahoney, *Fast Monte Carlo algorithms for matrices II: Computing a low-rank approximation to a matrix*, SIAM Journal on Computing, 36 (2006), pp. 158–183, https://doi.org/10.1137/S0097539704442696.

[35] P. Drineas, M. W. Mahoney, and S. Muthukrishnan, *Subspace sampling and relative-error matrix approximation: Column-based methods*, in Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 2006, https://doi.org/10.1007/11830924_30.

[36] A. Edelman, *Eigenvalues and condition numbers of random matrices*, SIAM Journal on Matrix Analysis and Applications, 9 (1988), pp. 543–560, https://doi.org/10.1137/0609045.

[37] E. N. Epperly and J. A. Tropp, *Efficient error and variance estimation for randomized matrix computations*, arXiv:2207.06342, (2023), https://arxiv.org/abs/2207.06342.

[38] E. N. Epperly, J. A. Tropp, and R. J. Webber, *XTrace: Making the most of every sample in stochastic trace estimation*, arXiv:2301.07825, (2023), https://arxiv.org/abs/2301.07825.

[39] S. Foucart and H. Rauhut, *A Mathematical Introduction to Compressive Sensing*, Springer, 2013, https://doi.org/10.1007/978-0-8176-4948-7.

[40] A. Frieze, R. Kannan, and S. Vempala, *Fast Monte-Carlo algorithms for finding low-rank approximations*, in Proceedings of the 39th Annual Symposium on Foundations of Computer Science, 1998, https://doi.org/10.1109/SFCS.1998.743487.

[41] K. J. Galinsky, G. Bhatia, P.-R. Loh, S. Georgiev, S. Mukherjee, N. J. Patterson, and A. L. Price, *Fast principal-component analysis reveals convergent evolution of ADH1B in Europe and East Asia*, The American Journal of Human Genetics, 98 (2016), pp. 456–472, https://doi.org/10.1016/j.ajhg.2015.12.022.

[42] R. Gilmore, *Lie Groups, Physics, and Geometry: An Introduction for Physicists, Engineers and Chemists*, Cambridge University Press, 2008, https://doi.org/10.1017/CBO9780511791390.

[43] A. Glielmo, B. E. Husic, A. Rodriguez, C. Clementi, F. Noé, and A. Laio, *Unsupervised learning methods for molecular simulation data*, Chemical Reviews, 121 (2021), pp. 9722–9758, https://doi.org/10.1021/acs.chemrev.0c01195.

[44] G. Golub, *Numerical methods for solving linear least squares problems*, Numerische Mathematik, 7 (1965), pp. 206–216, https://doi.org/10.1007/bf01436075.

[45] G. Golub and W. Kahan, *Calculating the singular values and pseudo-inverse of a matrix*, Journal of the Society for Industrial and Applied Mathematics Series B Numerical Analysis, 2 (1965), pp. 205–224, https://doi.org/10.1137/0702016.

[46] G. Golub and C. F. V. Loan, *Matrix Computations*, Johns Hopkins University Press, 4 ed., 2013, https://doi.org/10.56021/9781421407944.

[47] G. H. Golub, F. T. Luk, and M. L. Overton, *A block Lanczos method for computing the singular values and corresponding singular vectors of a matrix*, ACM Transactions on Mathematical Software, 7 (1981), p. 149–169, https://doi.org/10.1145/355945.355946.

[48] G. H. Golub and H. A. van der Vorst, *Eigenvalue computation in the 20th century*, Journal of Computational and Applied Mathematics, 123 (2000), pp. 35–65, https://doi.org/10.1016/S0377-0427(00)00413-1.

[49] K. Goto and R. A. v. d. Geijn, *Anatomy of high-performance matrix multiplication*, ACM Transactions on Mathematical Software, 34 (2008), https://doi.org/10.1145/1356052.1356053.

[50] M. Gu, *Subspace iteration randomization and singular value problems*, SIAM Journal on Scientific Computing, 37 (2015), pp. A1139–A1173, https://doi.org/10.1137/130938700.

[51] M. Gu and S. C. Eisenstat, *Efficient algorithms for computing a strong rank-revealing qr factorization*, SIAM Journal on Scientific Computing, 17 (1996), pp. 848–869, https://doi.org/10.1137/0917055.

[52] J. guang Sun, *Backward perturbation analysis of certain characteristic subspaces*, Numerische Mathematik, 65 (1993), pp. 357–382, https://doi.org/10.1007/bf01385757.
[53] N. Halko, P.-G. Martinsson, Y. Shkolnisky, and M. Tygert, *An algorithm for the principal component analysis of large data sets*, SIAM Journal on Scientific Computing, 33 (2011), pp. 2580–2594, https://doi.org/10.1137/100804139.
[54] 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, https://doi.org/10.1137/090771806.
[55] B. Hie, B. Bryson, and B. Berger, *Efficient integration of heterogeneous single-cell transcriptomes using Scanorama*, Nature Biotechnology, 37 (2019), pp. 685–691, https://doi.org/10.1038/s41587-019-0113-3.
[56] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, 2 ed., 2012, https://doi.org/10.1017/CBO9781139020411.
[57] International HapMap 3 Consortium, *Integrating common and rare genetic variation in diverse human populations*, Nature, 467 (2010), pp. 52–58, https://doi.org/10.1038/nature09298.
[58] T. Isaac, N. Petra, G. Stadler, and O. Ghattas, *Scalable and efficient algorithms for the propagation of uncertainty from data through inference to prediction for large-scale problems, with application to flow of the Antarctic ice sheet*, Journal of Computational Physics, 296 (2015), pp. 348–368, https://doi.org/10.1016/j.jcp.2015.04.047.
[59] D. Kershaw, *Some extensions of W. Gautschi's inequalities for the gamma function*, Mathematics of Computation, 41 (1983), pp. 607–611, https://doi.org/10.2307/2007697.
[60] J. Kuczyński and H. Woźniakowski, *Estimating the largest eigenvalue by the power and Lanczos algorithms with a random start*, SIAM Journal on Matrix Analysis and Applications, 13 (1992), pp. 1094–1122, https://doi.org/10.1137/0613066.
[61] R. Kumar, M. Graff, I. Vasconcelos, and F. J. Herrmann, *Target-oriented imaging using extended image volumes: a low-rank factorization approach*, Geophysical Prospecting, 67 (2019), pp. 1312–1328, https://doi.org/10.1111/1365-2478.12779.
[62] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, Journal of Research of the National Bureau of Standards, 45 (1950).
[63] R. M. Larsen, *Lanczos bidiagonalization with partial reorthogonalization*, DAIMI Report Series, 27 (1998), https://doi.org/10.7146/dpb.v27i537.7070.
[64] J. Lee and P. K. Kitanidis, *Large-scale hydraulic tomography and joint inversion of head and tracer data using the principal component geostatistical approach (PCGA)*, Water Resources Research, 50 (2014), pp. 5410–5427, https://doi.org/10.1002/2014WR015483.
[65] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide*, Society for Industrial and Applied Mathematics, 1998, https://doi.org/10.1137/1.9780898719628.
[66] A. Levis, D. Lee, J. A. Tropp, C. F. Gammie, and K. L. Bouman, *Inference of black hole fluid-dynamics from sparse interferometric measurements*, in IEEE/CVF International Conference on Computer Vision, 2021, https://doi.org/10.1109/ICCV48922.2021.00234.
[67] 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), https://doi.org/10.1145/3004053.
[68] R.-C. Li and L.-H. Zhang, *Convergence of the block Lanczos method for eigenvalue clusters*, Numerische Mathematik, 131 (2014), pp. 83–113, https://doi.org/10.1007/s00211-014-0681-6.
[69] T. M. Low, F. D. Igual, T. M. Smith, and E. S. Quintana-Orti, *Analytical modeling is enough for high-performance BLIS*, ACM Transactions on Mathematical Software, 43 (2016), https://doi.org/10.1145/2925987.
[70] P.-G. Martinsson and J. A. Tropp, *Randomized numerical linear algebra: Foundations and algorithms*, Acta Numerica, 29 (2020), p. 403–572, https://doi.org/10.1017/S0962492920000021.
[71] J. Mason and D. C. Handscomb, *Chebyshev Polynomials*, Chapman and Hall/CRC, 2002, https://doi.org/10.1201/9781420036114.
[72] MathWorks Inc., *MATLAB Version 9.9 (R2010a)*, 2022, https://www.mathworks.com.
[73] R. A. Meyer, C. Musco, and C. Musco, *On the unreasonable effectiveness of single vector Krylov methods for low-rank approximation*, arXiv:2305.02535, (2023), https://arxiv.org/abs/2305.02535.
[74] 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, 2015, https://dl.acm.org/doi/10.5555/2969239.2969395.

[75] Y. Nakatsukasa, *Fast and stable randomized low-rank matrix approximation*, arXiv:2009.11392, (2020), https://arxiv.org/abs/2009.11392.

[76] Y. Nakatsukasa and J. A. Tropp, *Fast & accurate randomized algorithms for linear systems and eigenvalue problems*, arXiv:2111.00113, (2021), https://arxiv.org/abs/2111.00113.

[77] F. Nüske, H. Wu, J.-H. Prinz, C. Wehmeyer, C. Clementi, and F. Noé, *Alanine dipeptide*. https://markovmodel.github.io/mdshare/ALA2/.

[78] F. Nüske, H. Wu, J.-H. Prinz, C. Wehmeyer, C. Clementi, and F. Noé, *Markov state models from short non-equilibrium simulations—Analysis and correction of estimation bias*, The Journal of Chemical Physics, 146 (2017), https://doi.org/10.1063/1.4976518.

[79] D. P. O'Leary, G. W. Stewart, and J. S. Vandergraft, *Estimating the largest eigenvalue of a positive definite matrix*, Mathematics of Computation, 33 (1979), pp. 1289–1292, https://doi.org/10.2307/2006463.

[80] S. O'Rourke, V. Vu, and K. Wang, *Random perturbation of low rank matrices: Improving classical bounds*, Linear Algebra and its Applications, 540 (2018), pp. 26–59, https://doi.org/10.1016/j.laa.2017.11.014.

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

[82] C. H. Papadimitriou, H. Tamaki, P. Raghavan, and S. Vempala, *Latent semantic indexing: A probabilistic analysis*, in Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, 1998, https://doi.org/10.1145/275487.275505.

[83] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Society for Industrial and Applied Mathematics, 1998, https://doi.org/10.1137/1.9781611971163.

[84] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Édouard Duchesnay, *Scikit-learn: Machine learning in Python*, Journal of Machine Learning Research, 12 (2011), pp. 2825–2830, http://jmlr.org/papers/v12/pedregosa11a.html.

[85] M. Pilanci and M. J. Wainwright, *Randomized sketches of convex programs with sharp guarantees*, IEEE Transactions on Information Theory, 61 (2015), pp. 5096–5115, https://doi.org/10.1109/TIT.2015.2450722.

[86] S. J. Press, *Applied Multivariate Analysis: Using Bayesian and Frequentist Methods of Inference*, Robert E. Krieger Publishing Co, 2 ed., 1982.

[87] V. Rokhlin, A. Szlam, and M. Tygert, *A randomized algorithm for principal component analysis*, SIAM Journal on Matrix Analysis and Applications, 31 (2010), pp. 1100–1124, https://doi.org/10.1137/080736417.

[88] S. Roweis, *Em algorithms for pca and spca*, in Proceedings of the 10th International Conference on Neural Information Processing Systems, 1997, https://dl.acm.org/doi/10.5555/3008904.3008993.

[89] M. Rudelson and R. Vershynin, *Sampling from large matrices: An approach through geometric functional analysis*, Journal of the ACM, 54 (2007), p. 21–es, https://doi.org/10.1145/1255443.1255449.

[90] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Society for Industrial and Applied Mathematics, 2011, https://doi.org/10.1137/1.9781611970739.

[91] T. Sárlos, *Improved approximation algorithms for large matrices via random projections*, in 2006 47th Annual IEEE Symposium on Foundations of Computer Science, 2006, https://doi.org/10.1109/FOCS.2006.37.

[92] P. J. Schmid, *Dynamic mode decomposition and its variants*, Annual Review of Fluid Mechanics, 54 (2022), pp. 225–254, https://doi.org/10.1146/annurev-fluid-030121-015835.

[93] T. M. Smith, R. v. d. Geijn, M. Smelyanskiy, J. R. Hammond, and F. G. V. Zee, *Anatomy of high-performance many-threaded matrix multiplication*, in IEEE 28th International Parallel and Distributed Processing Symposium, 2014, https://doi.org/10.1109/IPDPS.2014.110.

[94] G. W. Stewart, *Simultaneous iteration for computing invariant subspaces of non-Hermitian matrices*, Numerische Mathematik, 25 (1976), pp. 123–136, https://doi.org/10.1007/bf01462265.

[95] G. W. Stewart, *Perturbation theory for the singular value decomposition*, in SVD and Signal Processing II: Algorithms, Analysis and Applications, R. J. Vaccaro, ed., Elsevier, 1991, pp. 99–109, http://hdl.handle.net/1903/552.

[96] G. W. Stewart, *Matrix Algorithms*, vol. II: Eigensystems, Society for Industrial and Applied Mathematics, 2001, https://doi.org/10.1137/1.9780898718058.

[97] J. A. Tropp, *Improved analysis of the subsampled randomized Hadamard transform*, Advances in Adaptive Data Analysis, 03 (2011), pp. 115–126, https://doi.org/10.1142/S1793536911000787.

[98] J. A. Tropp, *Analysis of randomized block Krylov methods*, ACM Report 2018-02, Caltech, Pasadena, 2018, https://resolver.caltech.edu/CaltechAUTHORS:20210624-180721369.

[99] J. A. Tropp, *Randomized block Krylov methods for approximating extreme eigenvalues*, Numerische Mathematik, 150 (2021), pp. 217–255, https://doi.org/10.1007/s00211-021-01250-3.

[100] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher, *Fixed-rank approximation of a positive-semidefinite matrix from streaming data*, in Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017, https://dl.acm.org/doi/10.5555/3294771.3294888.

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

[102] 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, https://doi.org/10.1137/18M1201068.

[103] K. Tsuyuzaki, H. Sato, K. Sato, and I. Nikaido, *Benchmarking principal component analysis for large-scale single-cell RNA-sequencing*, Genome Biology, 21 (2020), https://doi.org/10.1186/s13059-019-1900-3.

[104] V. Vanhoucke, A. Senior, and M. Z. Mao, *Improving the speed of neural networks on CPUs*, in Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011, 2011, https://research.google/pubs/pub37631/.

[105] R. Vershynin, *High-Dimensional Probability: An Introduction with Applications in Data Science*, Cambridge University Press, 2018, https://doi.org/10.1017/9781108231596.

[106] J. M. Wallace, C. Smith, and C. S. Bretherton, *Singular value decomposition of wintertime sea surface temperature and 500-mb height anomalies*, Journal of Climate, 5 (1992), pp. 561 – 576, https://doi.org/10.1175/1520-0442(1992)005⟨0561:SVDOWS⟩2.0.CO;2.

[107] S. Wang, Z. Zhang, and T. Zhang, *Improved analyses of the randomized power method and block Lanczos method*, arXiv:1508.06429, (2015), https://arxiv.org/abs/1508.06429.

[108] P.-Å. Wedin, *Perturbation bounds in connection with singular value decomposition*, BIT, 12 (1972), pp. 99–111, https://doi.org/10.1007/bf01932678.

[109] 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, 2000, https://dl.acm.org/doi/10.5555/3008751.3008847.

[110] F. Woolfe, E. Liberty, V. Rokhlin, and M. Tygert, *A fast randomized algorithm for the approximation of matrices*, Applied and Computational Harmonic Analysis, 25 (2008), pp. 335–366, https://doi.org/10.1016/j.acha.2007.12.002.

[111] M. Yang, M. Graff, R. Kumar, and F. J. Herrmann, *Low-rank representation of omnidirectional subsurface extended image volumes*, Geophysics, 86 (2021), pp. S165–S183, https://doi.org/10.1190/geo2020-0152.1.

[112] W. Yu, Y. Gu, and Y. Li, *Efficient randomized algorithms for the fixed-precision low-rank matrix approximation*, SIAM Journal on Matrix Analysis and Applications, 39 (2018), pp. 1339–1359, https://doi.org/10.1137/17M1141977.

[113] Q. Yuan, M. Gu, and B. Li, *Superlinear convergence of randomized block Lanczos algorithm*, in IEEE International Conference on Data Mining, 2018, https://doi.org/10.1109/ICDM.2018.00193.