Accepted: 25 October 2024

DOI: 10.1002/cpa.22234

RESEARCH ARTICLE

Randomly pivoted Cholesky: Practical approximation of a kernel matrix with few entry evaluations

Yifan Chen[1] | **Ethan N. Epperly**[2] | **Joel A. Tropp**[2] | **Robert J. Webber**[3]

[1]Courant Institute of Mathematical Sciences, New York University, New York, USA

[2]Computing and Mathematical Sciences, California Institute of Technology, Pasadena, California, USA

[3]Department of Mathematics, University of California San Diego, La Jolla, California, USA

Correspondence
Robert J. Webber, Department of Mathematics, University of California San Diego, La Jolla, CA 92093, USA.
Email: rwebber@ucsd.edu

Funding information
Caltech Kortschak Scholar program; Courant Instructorship; Department of Energy, Grant/Award Number: DE-SC0021110; Office of Naval Research, Grant/Award Number: N00014-18-1-2363; National Science Foundation, Grant/Award Number: 1952777

Abstract

The randomly pivoted Cholesky algorithm (RPCHOLESKY) computes a factorized rank-k approximation of an $N \times N$ positive-semidefinite (psd) matrix. RPCHOLESKY requires only $(k+1)N$ entry evaluations and $\mathcal{O}(k^2N)$ additional arithmetic operations, and it can be implemented with just a few lines of code. The method is particularly useful for approximating a kernel matrix. This paper offers a thorough new investigation of the empirical and theoretical behavior of this fundamental algorithm. For matrix approximation problems that arise in scientific machine learning, experiments show that RPCHOLESKY matches or beats the performance of alternative algorithms. Moreover, RPCHOLESKY provably returns low-rank approximations that are nearly optimal. The simplicity, effectiveness, and robustness of RPCHOLESKY strongly support its use in scientific computing and machine learning applications.

1 | MOTIVATION

Kernel methods [57] are a class of machine learning tools for interpolation, regression, clustering, and summarization of data. For small to medium data sets (say, with fewer than 10^5 points), the literature contains evidence that kernel methods are effective for many learning tasks in scientific computing [10, 48, 52, 53, 57, 61]. For particular problems, carefully designed kernel methods can compete with or exceed the performance of neural networks [6, 7, 42, 49].

Kernel methods distill information about the pairwise similarities of N data points into a dense positive-semidefinite (psd) kernel matrix with dimensions $N \times N$. We must compare two data points to determine each entry of the kernel matrix, so it can be burdensome to compute and store all N^2 entries. To perform data analysis tasks with the kernel matrix, we solve linear systems or least-squares problems, or we perform eigenvalue decompositions. With direct algorithms, these linear algebra primitives require $\mathcal{O}(N^3)$ arithmetic. This scaling makes it prohibitive to apply kernel methods to the largest data sets.

Yet the situation is not hopeless. Even for high-dimensional data, the eigenvalues of the kernel matrix can decay surprisingly quickly [3, 70]. This phenomenon has a profound consequence for computation. **When spectral decay is present, we can replace the full kernel matrix with a low-rank approximation to accelerate kernel methods without much loss of accuracy**. This approach can be used to accelerate kernel interpolation [1], kernel ridge regression [23, 44, 55], and kernel spectral clustering [27]. Indeed, "low-rank kernel methods" can run orders of magnitude faster than direct methods that require a full decomposition of the dense kernel matrix (see, e.g., Section 4.3).

With this context in view, we pose a computational question: **What are the best algorithms for finding a low-rank approximation of a large psd kernel matrix?** Here are some desiderata:

1. **Entry evaluations**. We would like to compute a rank-k approximation after revealing only $\mathcal{O}(kN)$ entries of the kernel matrix, such as the k most salient columns.
2. **Arithmetic and storage**. The method should only expend $\mathcal{O}(k^2N)$ additional arithmetic, which is the cost to orthogonalize k vectors of dimension N. The method should return the approximation in factored form, using only $\mathcal{O}(kN)$ storage.
3. **Approximation quality**. The error in the computed rank-k approximation should be competitive with the best approximation that has rank r, where r is a number not much smaller than k.
4. **Reliability, robustness, simplicity**. The method should have consistent performance, and it should succeed for all inputs. The method should be easy to implement, and it should not require the user to adjust parameters.

The randomly pivoted Cholesky algorithm (RPCHOLESKY) is a fundamental numerical method that enjoys all four of these qualities. RPCHOLESKY enhances the classic pivoted partial Cholesky method with an adaptive, probabilistic rule for selecting the next pivot column. We have found that this simple modification consistently produces excellent low-rank matrix approximations, even when alternative pivot rules fail. See Algorithm 1 for pseudocode.

The RPCHOLESKY algorithm has a subtle history (Section 3), but it is fair to say that this method has never received the attention that it deserves. Our purpose is to bring this powerful algorithm into the light. We offer two main contributions:

ALGORITHM 1 RPCHOLESKY.

Input: Psd matrix $\boldsymbol{A} \in \mathbb{C}^{N\times N}$; approximation rank k

Output: Pivot set $\mathsf{S} = \{s_1, \dots, s_k\}$; matrix $\boldsymbol{F} \in \mathbb{C}^{N\times k}$ defining Nyström approximation $\widehat{\boldsymbol{A}} = \boldsymbol{F}\boldsymbol{F}^*$

Initialize $\boldsymbol{F} \leftarrow \boldsymbol{0}_{N\times k}$ and $\boldsymbol{d} \leftarrow \operatorname{diag} \boldsymbol{A}$ ▷ Evaluate diagonal of input matrix

for $i = 1$ to k **do**

Sample pivot $s_i \sim \boldsymbol{d} / \sum_{j=1}^{N} \boldsymbol{d}(j)$ ▷ With probability proportional to residual diagonal

$\boldsymbol{g} \leftarrow \boldsymbol{A}(:, s_i)$ ▷ Evaluate column s_i of input matrix

$\boldsymbol{g} \leftarrow \boldsymbol{g} - \boldsymbol{F}(:, 1 : i-1)\boldsymbol{F}(s_i, 1 : i-1)^*$ ▷ Remove overlap with previously chosen columns

$\boldsymbol{F}(:, i) \leftarrow \boldsymbol{g} / \sqrt{\boldsymbol{g}(s_i)}$ ▷ Update approximation

$\boldsymbol{d} \leftarrow \boldsymbol{d} - |\boldsymbol{F}(:, i)|^2$ ▷ Track diagonal of residual matrix

$\boldsymbol{d} \leftarrow \max\{\boldsymbol{d}, \boldsymbol{0}\}$ ▷ Ensure diagonal remains nonnegative

end for

1. **Empirical performance**. We provide the first numerical evidence that RPCHOLESKY is a competitive technique for approximating large kernel matrices that arise in scientific machine learning. Compared to alternative algorithms, RPCHOLESKY has greater speed, accuracy, or reliability.
2. **Rigorous error bounds**. We develop a new theoretical analysis that describes the performance of the RPCHOLESKY algorithm, as given in Algorithm 1. Our analysis gives a clear picture of why RPCHOLESKY is effective.

In summary, we present a slate of results to suggest that RPCHOLESKY is the leading method for low-rank approximation of large, psd kernel matrices. The effectiveness, robustness, and simplicity of RPCHOLESKY strongly recommend it for modern applications in scientific computing and machine learning.

1.1 | Plan for paper

The rest of the paper is organized as follows. Section 2 introduces RPCHOLESKY and its basic properties, and Section 3 outlines history and related work. Section 4 applies RPCHOLESKY to kernel ridge regression and kernel spectral clustering problems, and Section 5 establishes error bounds. Section 6 offers some conclusions.

1.2 | Notation

The elements of a matrix $\boldsymbol{A} \in \mathbb{C}^{N\times N}$ are written $[\boldsymbol{A}(i, j)]_{1\le i,j\le N}$, while the submatrices of $\boldsymbol{A}$ are expressed using MATLAB notation. For example, $\boldsymbol{A}(:, i)$ represents the ith column of $\boldsymbol{A}$ and $\boldsymbol{A}(\mathsf{S}, :)$ denotes the submatrix of $\boldsymbol{A}$ with rows indexed by the set S. The conjugate transpose of a (rectangular) matrix $\boldsymbol{F}$ is denoted as $\boldsymbol{F}^*$, and the Moore–Penrose pseudoinverse is $\boldsymbol{F}^\dagger$. We write $\boldsymbol{\Pi}_{\boldsymbol{F}}$ for the orthogonal projector onto the column span of $\boldsymbol{F}$.

The function $\lambda_i(\boldsymbol{A})$ outputs the ith largest eigenvalue of a psd matrix $\boldsymbol{A}$. The symbol $\preceq$ denotes the psd order on Hermitian matrices: $\boldsymbol{H} \preceq \boldsymbol{A}$ if and only if $\boldsymbol{A} - \boldsymbol{H}$ is psd. We say $\boldsymbol{A}$ is rank-r when $\operatorname{rank} \boldsymbol{A} \le r$. The symbol $[\![\boldsymbol{A}]\!]_r$ refers to a best rank-r approximation of a psd matrix

$\boldsymbol{A}$, which can be obtained from an r-truncated eigendecomposition. A best rank-r approximation may not be unique, so we employ this notation only in contexts where it leads to an unambiguous interpretation.

2 | RANDOMLY PIVOTED CHOLESKY

In large-scale kernel methods, it is expensive to evaluate all the entries of the psd kernel matrix. As a cheaper alternative, we can try to approximate the kernel matrix by adaptively evaluating a small number of the columns. In Section 2.1, we introduce the *column Nyström approximation*, which is optimal among all approximations using a given set of columns. In Section 2.2, we describe the *pivoted partial Cholesky algorithm* as an efficient strategy for forming a column Nyström approximation. In Section 2.3, we design a rule for selecting the columns (aka "pivots") in the pivoted partial Cholesky algorithm that leads to the RPCHOLESKY algorithm. In Sections 2.4 and 2.5, we summarize simple numerical experiments with RPCHOLESKY. Last, in Section 2.6, we present a new error bound for RPCHOLESKY that explains why the method is effective.

2.1 | Nyström approximation of a psd matrix

Let $\boldsymbol{A} \in \mathbb{C}^{N \times N}$ be an arbitrary psd matrix (not necessarily arising from a kernel computation). To approximate the matrix using a given subset S of the column indices, we can employ the *column Nyström approximation* [43, §19.2]:

$$\widehat{\boldsymbol{A}} := \boldsymbol{A}(:, \mathsf{S})\boldsymbol{A}(\mathsf{S}, \mathsf{S})^{\dagger}\boldsymbol{A}(\mathsf{S}, :) \quad \text{where} \quad \mathsf{S} \subseteq \{1, \dots, N\}. \tag{2.1}$$

Since psd matrices are self-adjoint, we note that $\boldsymbol{A}(\mathsf{S}, :) = \boldsymbol{A}(:, \mathsf{S})^{*}$. When the set S contains k column indices, the Nyström approximation $\widehat{\boldsymbol{A}}$ yields a rank-k psd approximation with the following desirable properties:

1. The Nyström approximation $\widehat{\boldsymbol{A}}$ agrees with the target matrix $\boldsymbol{A}$ in the distinguished columns. That is, $\widehat{\boldsymbol{A}}(:, \mathsf{S}) = \boldsymbol{A}(:, \mathsf{S})$.
2. The range of the Nyström approximation $\widehat{\boldsymbol{A}}$ coincides with the span of the distinguished columns: $\operatorname{range}(\widehat{\boldsymbol{A}}) = \operatorname{range}(\boldsymbol{A}(:, \mathsf{S}))$.
3. With respect to the psd order, the Nyström approximation $\widehat{\boldsymbol{A}}$ satisfies the bounds $\mathbf{0} \preceq \widehat{\boldsymbol{A}} \preceq \boldsymbol{A}$.

In fact, the Nyström approximation is the maximal psd matrix that satisfies properties (2) and (3). See [5, Theorem 5.3] for a rigorous statement.

We measure the quality of a Nyström approximation $\widehat{\boldsymbol{A}}$ using the *trace-norm error*:

$$\operatorname{tr}(\boldsymbol{A} - \widehat{\boldsymbol{A}}). \tag{2.2}$$

Since $\widehat{\boldsymbol{A}} \preceq \boldsymbol{A}$, the trace-norm error is always nonnegative. Other norms are possible, but the trace-norm error is especially meaningful in the kernel learning context [37, §5.2.4]. Our goal is to find a set S of k column indices that makes the trace-norm error as small as possible.

ALGORITHM 2 Pivoted partial Cholesky algorithm.

Input: Psd matrix $\boldsymbol{A} \in \mathbb{C}^{N\times N}$; approximation rank k

Output: Pivot set $\mathsf{S} = \{s_1, \dots, s_k\}$ and matrix $\boldsymbol{F} \in \mathbb{C}^{N\times k}$ defining Nyström approximation $\widehat{\boldsymbol{A}} = \boldsymbol{F}\boldsymbol{F}^*$

Initialize $\boldsymbol{F} \leftarrow \boldsymbol{0}_{N\times k}$

for $i = 1$ to k **do**

Select a pivot $s_i \in \{1, \dots, N\}$ ▷ See Section 2.3 for pivot rules

$\boldsymbol{g} \leftarrow \boldsymbol{A}(:, s_i)$ ▷ Evaluate the s_i column of the input matrix $\boldsymbol{A}$

$\boldsymbol{g} \leftarrow \boldsymbol{g} - \boldsymbol{F}(:, 1 : (i-1))\boldsymbol{F}(s_i, 1 : (i-1))^*$ ▷ Remove the influence of previously chosen columns

$\boldsymbol{F}(:, i) \leftarrow \boldsymbol{g}/\sqrt{\boldsymbol{g}(s_i)}$

end for

2.2 | The pivoted partial Cholesky algorithm

We can efficiently compute a Nyström approximation (2.1) via the *pivoted partial Cholesky algorithm*, presented as Algorithm 2.

Conceptually, the algorithm begins with an initial approximation $\widehat{\boldsymbol{A}}^{(0)} = \boldsymbol{0}$ and an initial residual $\boldsymbol{A}^{(0)} = \boldsymbol{A}$. At each step $i = 1, 2, 3, \dots, k$, we adaptively select a new column index $s_i \in \{1, \dots, N\}$, called a *pivot*, using some pivot rule (Section 2.3). Then we evaluate the s_i column $\boldsymbol{A}^{(i-1)}(:, s_i)$ of the current residual, and we use this column to update the approximation and the residual:

$$\begin{aligned} \widehat{\boldsymbol{A}}^{(i)} &= \widehat{\boldsymbol{A}}^{(i-1)} + \frac{\boldsymbol{A}^{(i-1)}(:, s_i)\boldsymbol{A}^{(i-1)}(s_i, :)}{\boldsymbol{A}^{(i-1)}(s_i, s_i)}; \\ \boldsymbol{A}^{(i)} &= \boldsymbol{A}^{(i-1)} - \frac{\boldsymbol{A}^{(i-1)}(:, s_i)\boldsymbol{A}^{(i-1)}(s_i, :)}{\boldsymbol{A}^{(i-1)}(s_i, s_i)}. \end{aligned} \tag{2.3}$$

We can also track the diagonal of the residual $\boldsymbol{A}^{(i)}$ using the formula

$$\operatorname{diag}(\boldsymbol{A}^{(i)}) = \operatorname{diag}(\boldsymbol{A}^{(i-1)}) - \frac{1}{\boldsymbol{A}^{(i-1)}(s_i, s_i)} \cdot |\boldsymbol{A}^{(i-1)}(:, s_i)|^2. \tag{2.4}$$

The function $|\cdot|^2 : \mathbb{C}^N \to \mathbb{R}^N_+$ returns the entrywise squared magnitude of a vector. This observation supports stopping rules based on functions of the diagonal entries of the residual, such as its trace.

The practical implementation of pivoted partial Cholesky (Algorithm 2) maintains the approximation in factored form: $\widehat{\boldsymbol{A}}^{(i)} = \boldsymbol{F}(:, 1 : i)\boldsymbol{F}(:, 1 : i)^*$ where $\boldsymbol{F} \in \mathbb{C}^{N\times k}$. We generate the columns of the factor $\boldsymbol{F}$ sequentially. The ith column $\boldsymbol{F}(:, i)$ is obtained by evaluating the s_i column $\boldsymbol{A}(:, s_i)$ of the input matrix and removing the influence of the previously selected columns.

The following classic result [72, pp. 24] connects the pivoted partial Cholesky algorithm with the column Nyström approximation.

Property 2.1 (Pivoted partial Cholesky computes a Nyström approximation). The pivoted partial Cholesky algorithm (Algorithm 2) with psd input matrix $\boldsymbol{A}$ and with pivot set $\mathsf{S} = \{s_1, \dots, s_k\}$ returns the column Nyström approximation $\widehat{\boldsymbol{A}} = \boldsymbol{A}(:, \mathsf{S})\boldsymbol{A}(\mathsf{S}, \mathsf{S})^\dagger \boldsymbol{A}(\mathsf{S}, :)$ in the factorized form $\widehat{\boldsymbol{A}} = \boldsymbol{F}\boldsymbol{F}^*$.

Indeed, at each step of the pivoted partial Cholesky algorithm, $\widehat{\boldsymbol{A}}^{(i)}$ is the Nyström approximation of $\boldsymbol{A}$ using the columns indexed by $\{s_1, \dots, s_i\}$.

What are the computational costs? For k steps of the pivoted partial Cholesky algorithm, we evaluate kN entries of the input matrix $\boldsymbol{A}$, plus any other entries required to implement the pivot rule. We expend $\mathcal{O}(k^2N)$ additional arithmetic operations, and the algorithm needs $\mathcal{O}(kN)$ storage.

In practice, we often run the pivoted partial Cholesky method until the trace-norm error falls below a specified threshold: $\operatorname{tr}(\boldsymbol{A} - \widehat{\boldsymbol{A}}) \leq \eta \cdot \operatorname{tr} \boldsymbol{A}$. This modification requires an uncertain number of steps, but it controls the error level and ensures that $\widehat{\boldsymbol{A}}$ can be reliably used in place of $\boldsymbol{A}$ for downstream computations.

2.3 | Pivot selection rules

In the pivoted partial Cholesky algorithm, we select a new pivot at each step. Although there are several natural strategies, it has long remained unclear how to quickly and reliably select pivots that result in matrix approximations that control the trace-norm error (2.2).

We have already seen that we can track the diagonal of the residual matrix via (2.4). One principled approach for pivot selection is to exploit the information contained in the diagonal. Indeed, the diagonal entries of a psd matrix $\boldsymbol{A} \in \mathbb{C}^{N \times N}$ are nonnegative, and they control the off-diagonal entries via the inequality

$$|\boldsymbol{A}(i, j)| \leq \sqrt{\boldsymbol{A}(i, i)\boldsymbol{A}(j, j)} \quad \text{for each } 1 \leq i, j \leq N.$$

Thus, a large diagonal entry $\boldsymbol{A}(j, j)$ indicates that column j might contain large-magnitude entries. By eliminating such a column, we can hope to substantially reduce the trace of the residual at each step. We outline several established strategies based on this intuition.

2.3.1 | Greedy pivoting

Because of the significance of large diagonal entries, we might be tempted to use a *greedy* pivoting strategy. At step i of the pivoted partial Cholesky method, the greedy method selects the pivot s_i by finding the position of the largest diagonal entry of the residual matrix $\boldsymbol{A}^{(i-1)}$, with ties broken arbitrarily. In symbols, the greedy pivot rule is

$$s_i \in \operatorname{argmin}_{1 \leq j \leq N} \boldsymbol{A}^{(i-1)}(j, j). \tag{2.5}$$

This greedy pivoting strategy has long been used in scientific computing and kernel machine learning, under the name "Cholesky with complete pivoting" [36]. The strategy is entirely based on *exploiting* the large diagonal entries without *exploring* any smaller ones. The overemphasis on exploitation makes the greedy method brittle, as this algorithm is often derailed by the presence of outlier columns.

2.3.2 | Uniform random pivoting

The opposite strategy from greedy pivoting is to sample each pivot uniformly at random:

$$s_i \sim \text{UNIFORM}\{1, \dots, N\} \qquad \text{for each } i = 1, \dots, k.$$

If we select a pivot that we have already seen, we can draw a sample again. Uniform sampling has been popular in kernel machine learning since the works of Williams & Seeger [69] and Drineas & Mahoney [21]. Uniform sampling has the reverse problem from the greedy method: it randomly *explores* without *exploiting* any information from the diagonal. As a result, uniform sampling leads to a poor approximation when there are small sets of columns that are distinct from all the others.

2.3.3 | Adaptive random pivoting

This paper advocates for a third way. The randomly pivoted Cholesky (RPCHOLESKY) algorithm balances *exploration* of the columns with *exploitation* of the information available from the diagonal. At the ith step, RPCHOLESKY adaptively samples the pivot s_i according to the probability distribution that is proportional to the diagonal entries of the current residual $\boldsymbol{A}^{(i-1)}$:

$$\mathbb{P}\{s_i = j\} = \frac{\boldsymbol{A}^{(i-1)}(j,j)}{\operatorname{tr} \boldsymbol{A}^{(i-1)}} \quad \text{for } j = 1, \dots, N. \tag{2.6}$$

Algorithm 1 contains a simple implementation of RPCHOLESKY. See Section 3 for the history and some related algorithms.

2.3.4 | Pivoting with a Gibbs distribution

In retrospect, we realize that the greedy method, uniform sampling, and RPCHOLESKY are connected. Consider the pivot rule that selects the next pivot s_i from a Gibbs probability distribution that is proportional to the diagonal entries of the residual matrix $\boldsymbol{A}^{(i-1)}$, after raising them to a power $\beta \in [0, \infty]$:

$$\mathbb{P}\{s_i = j\} = \frac{\boldsymbol{A}^{(i-1)}(j,j)^\beta}{\sum_{k=1}^N \boldsymbol{A}^{(i-1)}(k,k)^\beta} \quad \text{for each } j = 1, \dots, N. \tag{2.7}$$

The greedy method arises from the $\beta \to \infty$ limit ("zero temperature"), while uniform sampling arises from the $\beta \to 0$ limit ("infinite temperature"). RPCHOLESKY takes the intermediate value $\beta = 1$ ("not too hot, not too cold"). Our analysis (Section 5) proves that RPCHOLESKY ($\beta = 1$) yields rigorous error bounds, while we provide examples (Appendix C) where the extreme strategies ($\beta \to 0$ and $\beta \to \infty$) fail. The literature contains empirical evidence that values $\beta \notin \{0, 1, \infty\}$ can be effective for some matrices [60].

2.4 | Numerical results: Illustrative examples

In this section, we present two stylized examples to highlight the benefits of the RPCHOLESKY method and the potential weaknesses of some other matrix approximation algorithms. Real-world examples appear in Sections 2.5 and 4.

We applied several column Nyström approximation schemes to two kernel matrices:

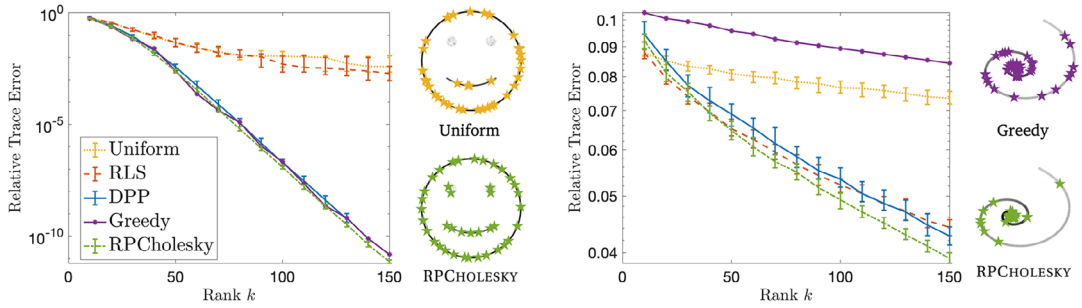

FIGURE 1 Rank-k approximation of Gaussian kernel matrices. Median relative trace-norm error $\operatorname{tr}\left(\boldsymbol{A}-\widehat{\boldsymbol{A}}^{(k)}\right)/\operatorname{tr}\boldsymbol{A}$ and 20% -80% quantile bars for several Nyström-based column approximation methods for **Smile** (*left*) and **Spiral** (*right*) examples. Selected pivots (colored stars) and data points (gray circles) for uniform, greedy, and RPCHOLESKY methods are shown next to each panel.

1. **Smile**: A Gaussian kernel matrix constructed from 10^4 data points depicting a smile in $\mathbb{R}^2$. The smile is located in $[-10,10]\times[-10,10]$ and the kernel bandwidth is $\sigma = 2.0$. The eyes are constructed from 10^2 points, making them easy to miss for certain sampling methods.
2. **Spiral**: A Gaussian kernel matrix constructed from 10^4 data points in $\mathbb{R}^2$ depicting the logarithmic spiral $(e^{0.2t}\cos t, e^{0.2t}\sin t)$ for non-equispaced parameter values $t \in [0,64]$. The kernel bandwidth is 1000, making the outer edge of the spiral a region of outliers in the data.

For each data set, Figure 1 tracks the mean relative trace-norm error $\operatorname{tr}(\boldsymbol{A}-\widehat{\boldsymbol{A}})/\operatorname{tr}(\boldsymbol{A})$ over 100 independent trials. The pivots selected by different Nyström-based column selection methods with $k = 40$ are shown next to each panel. See Appendix A for additional computational details.

These tests bring to light the failure modes of uniform sampling and the greedy method. Uniform sampling fails to select the pivots representing less populated regions of the data space, such as the eyes in the **Smile** example. The greedy method heavily emphasizes outliers, leading to poor approximation accuracy in the **Spiral** example. By contrast, RPCHOLESKY avoids these failure modes and achieves high accuracy for both test problems.

Figure 1 also evaluates two other randomized column selection schemes for Nyström approximation: *determinantal point process* (DPP) sampling [17] and *ridge leverage score* (RLS) sampling [2, 45, 56] (using the RRLS algorithm [45]). These methods offer strong theoretical guarantees, yet existing samplers are complicated and require care in implementation to avoid failure. In practice, they also require far more entry evaluations than RPCHOLESKY to achieve the same approximation quality.

Computational cost is another important difference between various Nyström column selection methods. One simple measure of cost is the total number of entry evaluations required to generate the pivots and then form the Nyström approximation. Uniform sampling is the cheapest Nyström method, requiring just kN entry evaluations. The greedy method and RPCHOLESKY follow closely behind, requiring $(k+1)N$ entry evaluations. For the examples in Figure 1, RLS sampling requires roughly $3kN$ entry evaluations and DPP sampling (using the `vfx` sampler [29]) requires between $80kN$ and $200kN$ entry evaluations, making these latter two methods comparatively expensive.

RLS sampling and DPP sampling exhibit other performance issues in addition to the high cost. RLS sampling often fails to provide a high-quality approximation for the **Smile** kernel

matrix. With an approximation rank of $k = 40$, there is a 95% chance of missing both eyes, which is better than the 99.6% chance of missing both eyes with uniform sampling, but still indicates a failure mode for the RLS algorithm. DPP sampling using the `vfx` algorithm [29] often fails to produce any output, generating an error that the matrix is close to rank-deficient. With existing software, it is necessary to switch to a slower `GS` sampler [29] based on a complete eigendecomposition of the matrix $\boldsymbol{A}$. Even the `GS` sampler fails for the **Smile** kernel matrix for $k \geq 140$.

2.5 | Numerical results: Real data

To confirm that the findings of Section 2.4 extend to real data sets, we compare different Nyström methods on a testbed of kernel matrices formed from twenty data sets in the LIBSVM [15], OpenML [64], and UCI [22] repositories. These examples are cataloged in [23, Table 1]. We subsample each data set to $N = 10^4$ points, standardize each feature, and form the Gaussian kernel matrix with bandwidth $\sigma = \sqrt{d}$, where d is the number of features. (See Section 4.1 for a primer on kernel matrices.) We omit DPP sampling due to its high computational cost, and we include the block RPCholesky method (introduced below in Section 3.4) with block size $T = 100$. For each method, we report the median relative trace error over ten trials.

Results are shown in Table 1. RPCholesky achieves the lowest trace error on every problem in the testbed. It achieves 8×10^7 and 5×10^3 times lower error than uniform and RLS sampling for the data set `COMET_MC_SAMPLE` with the fastest spectral decay. RPCholesky achieves a more modest 3× improvement over the greedy method in this example.

2.6 | Theoretical results

Given the excellent empirical performance of the RPCholesky algorithm, it is natural to seek a more rigorous explanation. We present the first proof that RPCholesky (as given in Algorithm 1) attains error bounds that are nearly optimal within the class of column Nyström approximations. See Section 3.6 and Appendix C for prior theoretical work.

Let $\boldsymbol{A}$ be a psd target matrix, and let $\widehat{\boldsymbol{A}}$ be a rank-k column Nyström approximation of the form (2.1). Then it is appropriate to compare the approximation error, $\operatorname{tr}(\boldsymbol{A} - \widehat{\boldsymbol{A}})$, against the error $\operatorname{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r)$ attained by a best rank-r psd approximation $[\![\boldsymbol{A}]\!]_r$ where the parameter $r \leq k$.

A Nyström approximation $\widehat{\boldsymbol{A}}$ using k randomly chosen columns is called an (r, ε)-*approximation* of the target matrix $\boldsymbol{A}$ when

$$\mathbb{E} \operatorname{tr}(\boldsymbol{A} - \widehat{\boldsymbol{A}}) \leq (1 + \varepsilon) \cdot \operatorname{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r) \tag{2.8}$$

for parameters $r \in \mathbb{N}$ and $\varepsilon > 0$. The expectation averages over the random choice of columns (e.g., the random pivots in RPCholesky). Our theory addresses the following question:

Question 2.2. How many columns k are sufficient to guarantee that a randomized column Nyström method attains an (r, ε)-approximation (2.8) for every $N \times N$ psd input matrix?

TABLE 1 Comparison of Nyström methods.

	Unif	RLS	Greedy	B-RPCHOL	RPCHOL	Opt
sensit_vehicle	1.57e-1	1.40e-1	2.07e-1	**1.37e-1**	**1.37e-1**	8.77e-2
yolanda	1.46e-1	1.39e-1	2.08e-1	1.37e-1	**1.36e-1**	8.41e-2
YearPredictionMSD	1.30e-1	1.20e-1	1.73e-1	1.18e-1	**1.16e-1**	6.79e-2
w8a	1.42e-1	1.17e-1	1.91e-1	1.14e-1	**1.05e-1**	6.09e-2
MNIST	1.21e-1	1.10e-1	1.67e-1	1.07e-1	**1.06e-1**	5.83e-2
jannis	1.11e-1	1.10e-1	1.28e-1	**1.09e-1**	**1.09e-1**	5.45e-2
HIGGS	6.73e-2	6.31e-2	8.51e-2	6.14e-2	**6.07e-2**	2.96e-2
connect_4	5.81e-2	5.07e-2	6.48e-2	4.85e-2	**4.81e-2**	2.25e-2
volkert	5.35e-2	4.42e-2	5.92e-2	4.26e-2	**4.17e-2**	1.99e-2
creditcard	4.83e-2	3.71e-2	5.77e-2	3.26e-2	**3.01e-2**	1.31e-2
Medical_Appointment	1.74e-2	1.43e-2	1.92e-2	1.31e-2	**1.29e-2**	4.59e-3
sensorless	1.20e-2	7.78e-3	8.70e-3	8.23e-3	**5.80e-3**	2.11e-3
ACSIncome	9.93e-3	5.55e-3	8.35e-3	4.26e-3	**4.02e-3**	1.27e-3
Airlines_DepDelay_1M	4.19e-3	2.37e-3	2.64e-3	1.93-3	**1.78e-3**	5.08e-4
covtype_binary	9.10e-3	2.12e-3	1.41e-3	1.26e-3	**1.04e-3**	2.97e-4
diamonds	1.31e-3	2.40e-4	1.12e-4	1.70e-4	**5.85e-5**	1.30e-5
hls4ml_lhc_jets_hlf	3.78e-4	7.30e-5	6.68e-5	5.36e-5	**4.38e-5**	1.04e-5
ijcnn1	3.91e-5	3.09e-5	2.86e-5	2.62e-5	**2.13e-5**	4.67e-6
cod_rna	6.00e-4	1.36e-5	9.19e-6	1.37e-5	**5.05e-6**	9.86e-7
COMET_MC_SAMPLE	3.65e-3	2.44e-7	1.2e-10	6.43e-5	**4.3e-11**	3.5e-12

Note: Relative trace error of rank-1000 Nyström approximation of kernel matrices produced by the RLS, uniform, RPCHOLESKY, greedy, and block RPCHOLESKY methods on a testbed of examples. The error of the optimal rank-1000 approximation is shown for reference, and we report the median error of ten trials. RPCHOLESKY achieves the smallest approximation error for every test problem.

Abbreviations: RLS, ridge leverage score; RPCholesky, randomly pivoted Cholesky algorithm.

To achieve an (r, ε)-approximation for a worst-case matrix, a column Nyström approximation must use at least $k \geq r/\varepsilon$ columns. For example, see Theorem C.1. We will establish that RPCHOLESKY achieves an (r, ε)-approximation for every psd matrix after accessing only a little more than r/ε columns. Here is a partial statement of our main result (Theorem 5.1):

Theorem 2.3 (Randomly pivoted Cholesky: simplified bound). *Fix $r \in \mathbb{N}$ and $\varepsilon > 0$, and let $\boldsymbol{A}$ be a psd matrix. The column Nyström approximation $\widehat{\boldsymbol{A}}^{(k)}$ produced by* RPCHOLESKY *(Algorithm 1) attains the error bound (2.8) provided that the number of columns, k, satisfies*

$$k \geq \frac{r}{\varepsilon} + r \log\left(\frac{1}{\varepsilon\eta}\right) \quad \textit{where} \quad \eta := \frac{\operatorname{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r)}{\operatorname{tr}(\boldsymbol{A})}. \tag{2.9}$$

For comparison, Table 2 presents the best available upper bounds on the number k of columns for RPCHOLESKY and other column Nyström approximation methods to achieve (2.8). These bounds are expressed in terms of r, ε, N, and the relative approximation error η defined in (2.9). DPP sampling satisfies the strongest error bounds of all the methods in Table 2. To achieve an

TABLE 2 Bounds for column Nyström approximations.

Method	Number k of columns to achieve (2.8)	Reference
Greedy method	$(1-(1+\varepsilon)\eta)N$	Theorem C.2
Uniform sampling*	$(r-1)/(\varepsilon\eta)+1/\varepsilon$	[28], Theorem C.3
DPP sampling	$r/\varepsilon+r-1$	[11, 35], Theorem C.4
RLS sampling†	$65(r/\varepsilon+r)\log((4/\eta)(r/\varepsilon+r))$	Theorem C.6
RPCholesky	$r/\varepsilon+r\log(1/(\varepsilon\eta))$	Theorem 5.1
RPCholesky	$r/\varepsilon+r+r\log_+(2^r/\varepsilon)$	Theorem 5.1
Lower bound	r/ε	[19, 35], Theorem C.1

Note: Upper bounds and a lower bound on the number of columns that several Nyström approximation schemes use to produce an (r,ε)-approximation (2.8). The parameter η is the relative error (see Theorem 2.3). All logarithms have base e, and $\log_+(x) := \max\{\log x, 0\}$ for $x > 0$. *The result for uniform sampling assumes the diagonal entries of $\boldsymbol{A}$ are all equal. †See Section C.5 for discussion.

(r,ε)-approximation with DPP sampling, it suffices to take the number k of columns as

$$k \geq \frac{r}{\varepsilon} + r - 1. \tag{2.10}$$

RPCholesky satisfies the second strongest error bound (2.9). The main difference between the DPP result (2.10) and the RPCholesky result (2.9) is the multiplicative factor $\log(1/\eta)$ present in the latter. However, because the relative error η appears inside the logarithm, this factor has only a modest impact on the computational scaling. Indeed, $\log(1/\eta)$ is between 2.4 and 26 for all twenty examples in Table 1. We establish Theorem 2.3 in Section 5, which contains additional results and discussion.

The bound on the approximation rank for RPCholesky is at least 65× smaller than the bound for RLS sampling. However, this quantitative dependence may be a result of the proof technique that overstates the difference between the methods. See Section C.5 for the proof of the RLS error bounds and related discussion.

The error bounds for RPCholesky are also *significantly stronger* than the bounds for the greedy method and uniform sampling. For a worst-case matrix $\boldsymbol{A}$, the greedy method requires $\Theta(N)$ columns to approach the best rank-r approximation error (Section C.2), while the uniform sampling method requires $\Theta(r/\eta)$ columns (Section C.3). In contrast, RPCholesky uses a number of columns that is independent of the dimension N and depends only logarithmically on the relative error η. These results help explain why RPCholesky does not exhibit the same failure modes as the greedy method and uniform sampling.

3 | HISTORY, RELATED WORK, AND EXTENSIONS

To understand the history of the RPCholesky algorithm, we must reinterpret it as a randomly pivoted QR algorithm. Indeed, almost all of the existing theoretical and numerical work that is relevant to RPCholesky is framed in terms of randomly pivoted QR algorithms, which—unlike RPCholesky—require reading all entries of the input matrix. In this section, we will explore this connection, discuss prior work, and describe related algorithms.

3.1 | Nyström approximations and projection approximations

We begin with an alternative perspective on the column Nyström approximation. Let $\boldsymbol{A} \in \mathbb{C}^{N\times N}$ be a psd matrix, and select any factorization $\boldsymbol{A} = \boldsymbol{B}^*\boldsymbol{B}$ where the factor $\boldsymbol{B} \in \mathbb{C}^{M\times N}$. For any subset $\mathsf{S} \subseteq \{1, \dots, N\}$ of column indices, the Nyström approximation $\widehat{\boldsymbol{A}} = \boldsymbol{A}(:, \mathsf{S})\boldsymbol{A}(\mathsf{S}, \mathsf{S})^\dagger \boldsymbol{A}(\mathsf{S}, :)$ of the target matrix $\boldsymbol{A}$ admits the representation

$$\widehat{\boldsymbol{A}} = \boldsymbol{B}^*\boldsymbol{\Pi}_{\boldsymbol{B}(:,\mathsf{S})}\boldsymbol{B},$$

where $\boldsymbol{\Pi}_{\boldsymbol{M}}$ denotes the orthogonal projector onto range$(\boldsymbol{M})$. To check this claim, set $\boldsymbol{M} = \boldsymbol{B}(:, \mathsf{S})$ and decompose the projector as $\boldsymbol{\Pi}_{\boldsymbol{M}} = \boldsymbol{M}(\boldsymbol{M}^*\boldsymbol{M})^\dagger\boldsymbol{M}^*$.

Equivalently, the Nyström approximation takes the form

$$\widehat{\boldsymbol{A}} = \widehat{\boldsymbol{B}}^*\widehat{\boldsymbol{B}} \quad \text{where} \quad \widehat{\boldsymbol{B}} = \boldsymbol{\Pi}_{\boldsymbol{B}(:,\mathsf{S})}\boldsymbol{B}.$$

We call $\widehat{\boldsymbol{B}}$ a *column projection approximation* of $\boldsymbol{B}$ with respect to the column index set S. The trace-norm error in the Nyström approximation can be expressed in terms of the projection approximation:

$$\operatorname{tr}(\boldsymbol{A} - \widehat{\boldsymbol{A}}) = \|\boldsymbol{B} - \widehat{\boldsymbol{B}}\|_{\mathrm{F}}^2.$$

Therefore, the problem of finding a set S of k columns to minimize the trace-norm error in the Nyström approximation of $\boldsymbol{A}$ is the same as the problem of finding a set S of k columns to minimize the squared Frobenius-norm error in the projection approximation of the factor $\boldsymbol{B}$. Additionally, the projection approximation $\widehat{\boldsymbol{B}}$ is the best Frobenius-norm approximation of $\boldsymbol{B}$ with the same range as $\boldsymbol{B}(:, \mathsf{S})$.

3.2 | Partial Cholesky and partial QR

Just as we compute the column Nyström approximation by means of the pivoted partial Cholesky algorithm, we can compute a column projection approximation via the classical column-pivoted partial QR algorithm (Algorithm B.1 in the appendix).

Here is a conceptual description. For an input matrix $\boldsymbol{B}$, the column-pivoted partial QR algorithm initializes the approximation $\widehat{\boldsymbol{B}}^{(0)} = \boldsymbol{0}$ and the residual $\boldsymbol{B}^{(0)} = \boldsymbol{B}$. At each step i, we choose a column index s_i using some pivot rule (Section 3.3). The updated approximation $\widehat{\boldsymbol{B}}^{(i)}$ is the projection approximation of $\boldsymbol{B}$ with respect to the selected columns $\{s_1, \dots, s_i\}$. The updated residual is $\boldsymbol{B}^{(i)} = \boldsymbol{B} - \widehat{\boldsymbol{B}}^{(i)}$. We repeat for k steps or until we trigger a stopping criterion.

The relationship between pivoted partial QR and pivoted partial Cholesky is classical [36, §5.2]:

Property 3.1 (Pivoted partial Cholesky and pivoted partial QR). Assume that $\boldsymbol{A} = \boldsymbol{B}^*\boldsymbol{B}$. Suppose pivoted partial Cholesky (Algorithm 2) selects pivot set S and outputs an approximation $\widehat{\boldsymbol{A}}$, while partial QR (Algorithm B.1) selects the same pivot set S and outputs an approximation $\widehat{\boldsymbol{B}}$. Then the approximations satisfy $\widehat{\boldsymbol{A}} = \widehat{\boldsymbol{B}}^*\widehat{\boldsymbol{B}}$.

While pivoted partial QR and pivoted partial Cholesky are algebraically related, these procedures have different computational costs. To construct a rank-k approximation of a rectangular matrix $\boldsymbol{B} \in \mathbb{C}^{M \times N}$, pivoted partial QR reads all MN entries of $\boldsymbol{B}$ and expends $\mathcal{O}(kMN)$ additional arithmetic operations. By contrast, pivoted partial Cholesky constructs a rank-k approximation of a psd matrix $\boldsymbol{A} \in \mathbb{C}^{N \times N}$ by looking at just kN matrix entries (ignoring the pivot rule) and expending $\mathcal{O}(k^2 N)$ additional operations.

3.3 | Pivot rules

The goal of column-pivoted partial QR is to identify the "most important" columns of a rectangular matrix. There is an extensive literature on pivot selection rules for QR decompositions (see [34] and the references therein). The most sophisticated methods, called strong rank-revealing QR algorithms, enjoy powerful approximation guarantees but are intricate and computationally expensive.

We can also consider simpler strategies that are computationally cheaper, akin to the diagonal pivot rules used in pivoted partial Cholesky. If $\boldsymbol{A} = \boldsymbol{B}^* \boldsymbol{B}$, then the diagonal entries of $\boldsymbol{A}$ agree with the squared column norms of $\boldsymbol{B}$. That is,

$$\boldsymbol{A}(j, j) = \|\boldsymbol{B}(:, j)\|_2^2 \quad \text{for each } j = 1, \dots, N.$$

This correspondence allows us to equip partial QR with analogs of the partial Cholesky pivoting strategies. In particular, greedy pivot rules are classical [36, §1].

From our current vantage, it is natural to consider a *randomly pivoted QR method* (Algorithm B.2 in the appendix with $T = 1$) that is akin to randomly pivoted Cholesky (Algorithm 1). At each step i of this method, we sample the next pivot s_i in proportion to the squared column norms of the residual $\boldsymbol{B}^{(i-1)}$.

3.4 | Blocking

A standard strategy for accelerating (column-pivoted) QR methods is to select a block of columns to eliminate at each step [33, Section 5.2.3]. In particular, we can consider a blocked variant of randomly pivoted QR (Algorithm B.2 in the appendix with $T > 1$). Let $T \geq 1$ be a block size parameter. At each step i, we sample T pivot columns independently with probability proportional to the squared column norms of the residual $\boldsymbol{B}^{(i-1)}$. We project out these columns and repeat.

Analogously, we can develop a blocked version of RPCHOLESKY (Algorithm 3 with $T > 1$). At each step i, we sample T pivot columns independently with probability proportional to the diagonal entries of the residual $\boldsymbol{A}^{(i-1)}$. We eliminate these columns and repeat.

Both of these methods require careful implementation to manage potential issues with numerical stability. In particular, block RPCHOLESKY repeatedly computes a full Cholesky decomposition of a square matrix that might be numerically rank-deficient. We can address this issue by adding a positive multiple of the identity to this matrix; see Algorithm 3 for details. Block randomly pivoted QR, on the other hand, can suffer from loss of orthogonality in the computed

ALGORITHM 3 RPCHOLESKY: Block variant.

Input: Psd matrix $\boldsymbol{A} \in \mathbb{C}^{N\times N}$; block size T; tolerance η or approximation rank k which is a multiple of T

Output: Pivot set S; matrix $\boldsymbol{F}$ defining Nyström approximation $\widehat{\boldsymbol{A}} = \boldsymbol{F}\boldsymbol{F}^*$

Initialize $\boldsymbol{F} \leftarrow \boldsymbol{0}_{N\times k}$, $\mathsf{S} \leftarrow \emptyset$, and $\boldsymbol{d} \leftarrow \operatorname{diag} \boldsymbol{A}$ ▷ Evaluate diagonal of input matrix

for $i = 0$ to $k/T - 1$ **do** ▷ Alternatively, run until $\sum_{j=1}^{N} \boldsymbol{d}(j) \le \eta \operatorname{tr} \boldsymbol{A}$

Sample $s_{iT+1}, \dots, s_{iT+T} \overset{\text{iid}}{\sim} \boldsymbol{d}/\sum_{j=1}^{N} \boldsymbol{d}(j)$ ▷ Probability prop. to diagonal elements of residual

$\mathsf{S}' \leftarrow \mathit{Unique}(\{s_{iT+1}, \dots, s_{iT+T}\})$

$\mathsf{S} \leftarrow \mathsf{S} \cup \mathsf{S}'$

$\boldsymbol{G} \leftarrow \boldsymbol{A}(:, \mathsf{S}')$ ▷ Evaluate columns S' of input matrix

$\boldsymbol{G} \leftarrow \boldsymbol{G} - \boldsymbol{F}\boldsymbol{F}(\mathsf{S}', :)^*$ ▷ Remove overlap with previously chosen columns

$\boldsymbol{R} \leftarrow \mathit{Chol}(\boldsymbol{G}(\mathsf{S}', :) + \varepsilon_{\text{mach}} \operatorname{tr}(\boldsymbol{G}(\mathsf{S}', :))\mathbf{I})$ ▷ Stabilized Cholesky $\boldsymbol{G}(\mathsf{S}', :) \approx \boldsymbol{R}^*\boldsymbol{R}$

$\boldsymbol{F}(:, iT + 1 : iT + |\mathsf{S}'|) \leftarrow \boldsymbol{G}\boldsymbol{R}^{-1}$ ▷ Update approximation

$\boldsymbol{d} \leftarrow \boldsymbol{d} - \mathit{SquaredRowNorms}(\boldsymbol{G}\boldsymbol{R}^{-1})$ ▷ Track diagonal of residual matrix

$\boldsymbol{d} \leftarrow \max\{\boldsymbol{d}, \boldsymbol{0}\}$ ▷ Ensure diagonal remains nonnegative

end for

Remove zero columns from $\boldsymbol{F}$

$\boldsymbol{Q}$ matrix; see a standard numerical linear algebra reference (e.g., [33, Chapter 5]) for discussion on stably computing a QR decomposition.

How does block RPCHOLESKY compare to simple RPCHOLESKY? Usually, the block RPCHOLESKY method is faster. When block RPCHOLESKY with block size $T = 50$ is applied to a $10^5 \times 10^5$ kernel matrix (see Section 4.2), we obtain a 5× speedup using the ℓ_1 Laplace kernel (4.4) and a 20× speedup after switching to the Gaussian kernel. As demonstrated in Table 1, the approximation quality of block RPCHOLESKY and simple RPCHOLESKY is similar for many inputs. However, block RPCHOLESKY can produce a significantly worse approximation on some inputs, leading to a 10^6× higher trace-norm error when applied to the `COMET_MC_SAMPLE` example, for example. The theoretical and empirical properties of block RPCHOLESKY are studied in the follow-up paper [24], which introduces an "accelerated RPCHOLESKY" method that remedies the deficits of block RPCHOLESKY.

3.5 | Randomly pivoted QR: Origins

We believe that the randomly pivoted QR method was first proposed in the theoretical computer science literature on column subset selection problems. In 2004, Frieze et al. [28] studied projection approximations where a set of column indices is chosen randomly by sampling in proportion to the squared column norms of the input matrix. In 2006, Deshpande et al. [19, 20] described a procedure that applies the Frieze et al. approximation iteratively, projecting out the contributions of previously selected columns at each step. They called the resulting method "adaptive sampling," in contrast to the one-shot sampling method of Frieze et al. Their approach is essentially the same as block randomly pivoted QR, modulo implementation details. As we will explain, the blocking is central to their proposal. We have chosen to use the terminology "(block) randomly pivoted QR" in this work because it clarifies the relationship with standard linear algebra algorithms.

3.6 | Randomly pivoted QR: Theory

In 2006, the original papers [19, 20] on randomly pivoted QR (i.e., adaptive sampling) focused on proving approximation guarantees. Here is a typical result.

Proposition 3.2 (Deshpande et al. [20, Theorem 1.2]). *Fix a matrix $\boldsymbol{B} \in \mathbb{R}^{M\times N}$, a target approximation rank r, and a tolerance $\varepsilon \in (0,1)$. Set the block size $T \geq r/\varepsilon$. After s steps, the block randomly pivoted QR method (Algorithm B.2) produces a random projection approximation $\widehat{\boldsymbol{B}}$ with rank $k = s \cdot T$ that satisfies*

$$\mathbb{E}\|\boldsymbol{B} - \widehat{\boldsymbol{B}}\|_{\mathrm{F}}^2 \leq (1-\varepsilon)^{-1}\|\boldsymbol{B} - [\![\boldsymbol{B}]\!]_r\|_{\mathrm{F}}^2 + \varepsilon^s\|\boldsymbol{B}\|_{\mathrm{F}}^2.$$

Using Property 3.1, we obtain a parallel result for block RPCHOLESKY (Algorithm 3). Fix a psd input matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$, an approximation rank r, and a tolerance $\varepsilon \in (0,1)$. Set the block size $T \geq r/\varepsilon$. After choosing k/T blocks of columns, block RPCHOLESKY produces a random Nyström approximation $\widehat{\boldsymbol{A}}$ with rank k that satisfies

$$\mathbb{E}\,\mathrm{tr}(\boldsymbol{A} - \widehat{\boldsymbol{A}}) \leq (1-\varepsilon)^{-1}\,\mathrm{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r) + \varepsilon^{k/T}\,\mathrm{tr}(\boldsymbol{A}).$$

Assuming $\varepsilon \leq 1/2$, this result guarantees an $(r, 3\varepsilon)$-approximation with rank

$$k = \frac{r}{\varepsilon} + r \cdot \frac{\log(1/\eta)}{\varepsilon \log(1/\varepsilon)} \quad \text{where} \quad \eta = \frac{\mathrm{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r)}{\mathrm{tr}(\boldsymbol{A})}.$$

For comparison, our result Theorem 2.3 guarantees an (r,ε)-approximation with approximation rank $k = r/\varepsilon + r\log(1/(\eta\varepsilon))$, which is always a stronger bound.

More seriously, Proposition 3.2 requires the user to fix the approximation rank r and error tolerance ε in advance. The block size T is adapted to both parameters. Furthermore, the statement is vacuous in case the block size $T = 1$. In other words, the existing theory is silent about the versions of these algorithms that we recommend for use in practice, which have fixed block size and stopping rules that depend on the observed error.

Our paper provides the first analysis of RPCHOLESKY that does not require unrealistic parameter choices and that addresses the fundamental case where the block size $T = 1$. See Section 5 for the details, including a new error bound (Corollary 5.2) for randomly pivoted QR with $T = 1$. In the followup paper [24], we extend our analysis to handle the case of a fixed block size $T > 1$.

3.7 | Randomly pivoted QR: Empirical work

In the period from 2009 to 2013, researchers in applied machine learning explored the empirical performance of randomly pivoted QR for approximating kernel matrices. In sharp contrast to our findings for RPCHOLESKY (Section 4), their conclusions were pessimistic.

We must stress that the existing numerical work applies randomly pivoted QR to a kernel matrix to obtain a column projection approximation at a cost of $\mathcal{O}(kN^2)$ operations (they run Algorithm B.2 with input $\boldsymbol{A}$). **These studies do not apply RPCHOLESKY to obtain a column Nyström approximation at the lesser cost of $\mathcal{O}(k^2N)$ operations**.

In their review of kernel approximation, Kumar et al. reported [41, Table 3] that randomly pivoted QR (Algorithm B.2) was 60× to 800× slower than the Nyström approximation (2.1) with uniformly sampled columns. For data sets with $N > 4000$ data points, they did not even run the randomly pivoted QR algorithm because they considered it impractical. Summarizing their findings, they emphasized "the computational and storage burdens" (pp. 990) and they stated that the algorithm "requires a full pass through [the kernel matrix] $\boldsymbol{K}$ at each iteration and is thus inefficient for large $\boldsymbol{K}$" (pp. 989).

The literature describes some attempts [39–41, 67] to improve randomly pivoted QR, but the algorithm fell into disuse over the subsequent decade.

3.8 | Randomly pivoted Cholesky: Origins

In 2017, Musco & Woodruff [46, pp. 3] briefly noted that one can perform "adaptive sampling" more efficiently, given access to the Gram matrix. Their observation suggests an algorithm similar to block RPCHOLESKY (Algorithm 3). This paper does not include an implementation or report any numerical experiments.

Randomly pivoted Cholesky also appears in a 2020 paper of Poulson [47]. Rather than using RPCHOLESKY for low-rank approximation, Poulson uses RPCHOLESKY to sample from a projection DPP. In Poulson's work, the input matrix $\boldsymbol{A}$ is always a rank-k orthoprojector, RPCHOLESKY is always run for exactly k steps, and the computational output is the set S of pivots; the factor $\boldsymbol{F}$ is discarded.

To the best of our knowledge, the papers [46, 47] are the sole references to RPCHOLESKY in the literature. Neither paper documents numerical experiments or provides a theoretical analysis of RPCHOLESKY for the low-rank approximation task.

3.9 | Comparison with random Fourier features

Random Fourier features (RFF) [50] is a popular Monte Carlo method for the low-rank approximation of a psd kernel matrix (Section 4.1) associated with a translationally invariant kernel. The main disadvantage of RFF that the approximation error $\boldsymbol{A} \approx \widehat{\boldsymbol{A}}$ converges at the Monte Carlo rate $\mathcal{O}(k^{-1/2})$, whereas RPCHOLESKY is *spectrally accurate*, producing approximations comparable to a best low-rank approximation. For applications where a high-accuracy approximation is important, RFF fares significantly worse than Nyström methods like RPCHOLESKY [23]. However, RFF can be useful in applications where a fast, crude approximation of $\boldsymbol{A}$ is sufficient.

4 | APPLICATIONS TO KERNEL MACHINE LEARNING

In this section, we undertake a numerical study to evaluate the performance of RPCHOLESKY on benchmark kernel computations from scientific machine learning. Section 4.2 treats a kernel ridge regression problem that arises in quantum chemistry, and Section 4.3 discusses a kernel spectral clustering problem from molecular biophysics.

4.1 | Kernel methods: Basics

Kernel methods [57] are designed for analyzing data in a general domain $\mathcal{X}$, equipped with a *kernel function* $K : \mathcal{X} \times \mathcal{X} \to \mathbb{C}$. We interpret the kernel function as a measure of similarity between a pair of data points. Suppose that $\boldsymbol{x}_1, \dots, \boldsymbol{x}_N \in \mathcal{X}$ is a list of N data points. We can form a *kernel matrix* $\boldsymbol{A} \in \mathbb{C}^{N \times N}$ that tabulates the pairwise similarities:

$$\boldsymbol{A}(i, j) := K(\boldsymbol{x}_i, \boldsymbol{x}_j) \quad \text{for } 1 \le i, j \le N.$$

We say that the kernel function K is *positive definite* if the kernel matrix $\boldsymbol{A}$ is psd for every family of N data points in $\mathcal{X}$ and every natural number N. For example, the inner-product kernel and the Gaussian kernel are both positive-definite kernels on $\mathbb{C}^d$:

$$K(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^* \boldsymbol{y} \qquad \text{(inner-product kernel);}$$

$$K(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{1}{2\sigma^2} \|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \right) \qquad \text{(Gaussian kernel).}$$

The parameter $\sigma > 0$ is called the *bandwidth* of the Gaussian kernel. Kernel methods reduce data analysis tasks in $\mathcal{X}$ to linear algebra computations on the kernel matrix $\boldsymbol{A}$.

We have seen that RPCHOLESKY quickly and reliably finds a low-rank approximation of a psd kernel matrix. Equivalently, the pivots S of RPCHOLESKY identify a modest number of data points $\{\boldsymbol{x}_s : s \in \mathsf{S}\}$ that can be used to summarize the data set. Therefore, we can incorporate RPCHOLESKY into the computational pipeline to obtain more scalable algorithms for kernel machine learning. We will elaborate on this idea in the next two sections.

4.2 | Kernel ridge regression

One powerful application for RPCHOLESKY is to accelerate *kernel ridge regression* (KRR) [57, §4.9.1]. We will study an application in quantum chemistry.

4.2.1 | Functional regression

KRR is a nonlinear extension of least-squares regression that approximates an unknown input–output map using a positive-definite kernel function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{C}$ and input–output pairs $(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_N, y_N) \in \mathcal{X} \times \mathbb{C}$. As output, KRR provides a prediction function of the form

$$f(\cdot\,; \boldsymbol{\beta}) := \sum_{i=1}^{N} \beta_i K(\boldsymbol{x}_i, \cdot), \tag{4.1}$$

with the coefficient vector $\boldsymbol{\beta}$ chosen to minimize a regularized least-squares loss:

$$\min_{\boldsymbol{\beta} \in \mathbb{C}^N} \frac{1}{N} \sum_{j=1}^{N} \left| f(\boldsymbol{x}_j\,; \boldsymbol{\beta}) - y_j \right|^2 + \lambda \sum_{i,j=1}^{N} \beta_i \beta_j K(\boldsymbol{x}_i, \boldsymbol{x}_j).$$

ALGORITHM 4 RPCHOLESKY-accelerated kernel ridge regression.

Input: Data points $\mathsf{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\} \subseteq \mathbb{C}^d$; output values $\boldsymbol{y} \in \mathbb{C}^N$; approximation rank k; regularization parameter $\lambda > 0$

Output: Pivots S and coefficients $\widehat{\boldsymbol{\beta}}$ defining a prediction function $\widehat{f}(\cdot)$ by (4.3)

$\boldsymbol{A} \leftarrow \mathit{KernelMatrix}(\mathsf{X})$

$(\sim, \mathsf{S}) \leftarrow \mathit{RPCholesky}(\boldsymbol{A}, k)$

$\widehat{\boldsymbol{\beta}} \leftarrow \big(\boldsymbol{A}(\mathsf{S}, :)\boldsymbol{A}(:, \mathsf{S}) + \lambda N\, \boldsymbol{A}(\mathsf{S}, \mathsf{S})\big)^{-1} \boldsymbol{A}(\mathsf{S}, :)\boldsymbol{y}$

Explicitly, the vector $\boldsymbol{\beta}$ is the solution to a linear system

$$\boldsymbol{\beta} = (\boldsymbol{A} + \lambda N\, \mathbf{I})^{-1} \boldsymbol{y}, \tag{4.2}$$

where $\boldsymbol{A}$ is the $N \times N$ kernel matrix induced by the input data $\boldsymbol{x}^{(i)}$ and $\boldsymbol{y}$ is the vector of output values.

4.2.2 | Restricted KRR via RPCHOLESKY

Directly computing the vector $\boldsymbol{\beta}$ via (4.2) would require solving a dense linear system at $\mathcal{O}(N^3)$ cost. As a faster alternative, we can solve a *restricted* version of the KRR problem at $\mathcal{O}(k^2 N)$ cost, where k is a user-definable parameter. Restricted KRR was proposed by Smola and Bartlett [59] and developed in subsequent works [23, 55, 56].

In restricted KRR, we first identify a set $\mathsf{S} = \{s_1, \dots, s_k\} \subseteq \{1, \dots, N\}$ of k *landmarks* that achieve good coverage over the data set. Next, we find a restricted prediction function

$$\widehat{f}\big(\cdot\,; \widehat{\boldsymbol{\beta}}\big) = \sum_{i=1}^{k} \widehat{\beta}_i K\big(\boldsymbol{x}_{s_i}, \cdot\big) \tag{4.3}$$

by solving the regularized least-squares problem

$$\min_{\widehat{\boldsymbol{\beta}} \in \mathbb{C}^k} \frac{1}{N} \sum_{j=1}^{N} \left| \widehat{f}\big(\boldsymbol{x}_j\,; \widehat{\boldsymbol{\beta}}\big) - y_j \right|^2 + \lambda \sum_{i,j=1}^{k} \widehat{\beta}_i \widehat{\beta}_j K\big(\boldsymbol{x}_{s_i}, \boldsymbol{x}_{s_j}\big).$$

The coefficient vector $\widehat{\boldsymbol{\beta}} \in \mathbb{C}^k$ is given by a smaller linear system

$$\widehat{\boldsymbol{\beta}} = \big(\boldsymbol{A}(\mathsf{S}, :)\boldsymbol{A}(:, \mathsf{S}) + \lambda N\, \boldsymbol{A}(\mathsf{S}, \mathsf{S})\big)^{-1} \boldsymbol{A}(\mathsf{S}, :)\boldsymbol{y}.$$

involving a $k \times k$ matrix, which is relatively inexpensive to solve. Forming and solving this system requires $\mathcal{O}(k^2 N)$ operations. Evaluating the prediction function (4.3) for restricted KRR requires just $\mathcal{O}(k)$ operations, which improves on the $\mathcal{O}(N)$ cost of evaluating (4.1).

In past work, the landmark set S has been selected by uniform random sampling [23, 55], ridge leverage score sampling [56], or greedy procedures [59]. To improve on these approaches, we propose choosing the landmarks S to be the pivot set chosen by RPCHOLESKY, resulting in the RPCHOLESKY-accelerated KRR method shown in Algorithm 4. We demonstrate below that the

RPCHOLESKY-based approach leads to improved out-of-sample prediction accuracy compared to previous landmark selection approaches.

4.2.3 | QM9 data

To showcase the effectiveness of RPCHOLESKY-accelerated KRR, we use Algorithm 4 to predict the highest occupied molecular orbital (HOMO) energy of organic molecules from the QM9 data set [51, 54]. The HOMO energy quantifies the electron-donating capacity of a molecule, and it is traditionally obtained using expensive first-principles calculations. As a modern alternative, a recent "Editor's Pick" journal article [61] proposes applying KRR to predict HOMO energies, and the authors train their prediction function on the QM9 data set due to its large size and molecular diversity. Here, we evaluate the RPCHOLESKY-accelerated KRR method on the HOMO prediction task with the QM9 data.

To represent molecules as vectors for KRR, we use a standard feature set based on the Coulomb repulsions between the atomic nuclei and the nuclear charges [61, §III.A]. We standardize the data and, following [61], evaluate the similarity between data points using the positive-definite ℓ_1 Laplace kernel:

$$K(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{1}{\sigma}\sum\nolimits_{j=1}^{d}|\boldsymbol{x}(j) - \boldsymbol{y}(j)|\right). \tag{4.4}$$

We divide the data into 100,000 data points for training and roughly 33,000 data points for testing and set the bandwidth σ and the ridge parameter λ using cross-validation.

Forming and storing the full kernel matrix for the QM9 data set would require 40 GB and 4 trillion arithmetic operations (20,000× the operation count for the **Smile** matrix from Section 2.4). Because of this high computational cost, previous authors [61] applied KRR using a random subsample of $N \leq 64{,}000$ training points, but they remarked that the approximation quality improves with the size of the data (see their Figure 8). In contrast, our RPCHOLESKY-accelerated computational approach allows us to use all $N = 100{,}000$ training points at a modest computational cost (5 min on a laptop computer for $k = 1000$).

Figure 2 displays the results for the test portion of the data set ($N_{\text{test}} = 3.3 \times 10^4$ data points). The out-of-sample prediction errors are measured using the symmetric mean absolute percentage error:

$$\text{SMAPE} = \frac{1}{N_{\text{test}}}\sum_{i=1}^{N_{\text{test}}}\frac{\left|y_i^{\text{test}} - \widehat{f}\left(\boldsymbol{x}_i^{\text{test}}; \widehat{\boldsymbol{\beta}}\right)\right|}{\frac{1}{2}\left|y_i^{\text{test}}\right| + \frac{1}{2}\left|\widehat{f}\left(\boldsymbol{x}_i^{\text{test}}; \widehat{\boldsymbol{\beta}}\right)\right|}. \tag{4.5}$$

The SMAPE for RPCHOLESKY, uniform sampling, and RLS sampling are similar, with uniform sampling being the slight favorite. The greedy method has notably worse performance than the three other methods, and we were unable to use DPP sampling because of the large values $N = 10^5$ and $k = 10^3$.

Nevertheless, RPCHOLESKY leads to higher accuracy than uniform sampling for the *extremal* molecules in the test set which have the greatest number of atoms (29 atoms). Across these nine

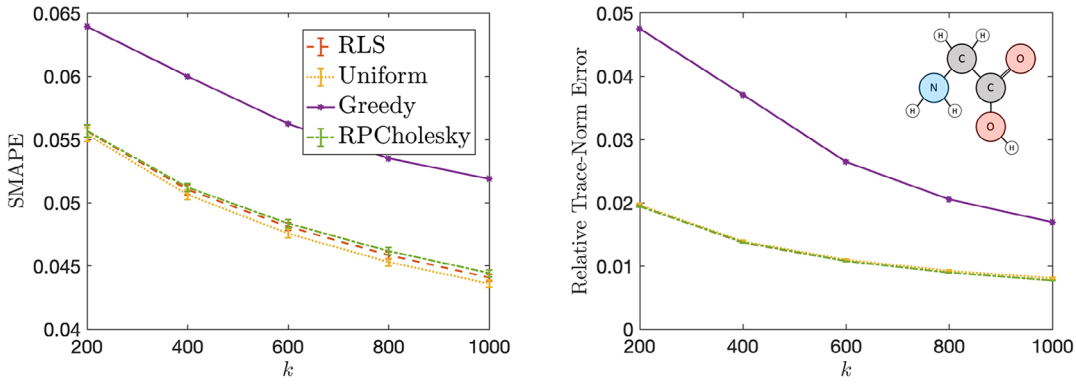

FIGURE 2 Kernel ridge regression for QM9 data. *Left*: Prediction error (4.5) for several Nyström algorithms. *Right*: Relative trace-norm error.

TABLE 3 Out-of sample prediction for QM9 data.

Compound #	Composition	Uniform	Greedy	RLS	RPCHOLESKY
1996	CC(C)CC1CO1	0.481	**0.034**	0.036	0.035
8664	CC(CO)(C=O)C=O	0.475	0.039	**0.030**	0.033
13,812	CC(O)C1CC1CO	0.473	0.045	0.023	**0.022**
64,333	CC1(C)CCOC(=N)O1	0.538	**0.032**	0.084	0.077
81,711	OC1C2CC1OCCO2	0.536	**0.007**	0.019	0.018
109,816	CCC1C(O)C1CC#C	0.516	0.062	0.018	**0.016**
118,229	COCC(C)OC(C)C	0.529	0.051	0.018	**0.017**
122,340	CC1C(CCCO)N1C	0.533	**0.002**	0.035	0.027
131,819	OCCCN1C=NC=N1	0.486	0.081	0.028	**0.026**
Average		0.507	0.039	0.033	**0.030**

Note: Symmetric absolute percentage error for the nine largest molecules in the test portion of the QM9 data set. The smallest errors in each row are marked in bold.
Abbreviations: RLS, ridge leverage score; RPCholesky, randomly pivoted Cholesky algorithm.

molecules, Table 3 shows that RPCHOLESKY is 10%–30% more accurate than RLS and greedy pivoting, and RPCHOLESKY achieves **17× smaller prediction errors** than uniform sampling. This observation suggests that RPCHOLESKY is more effective at representing less populated regions of data space, as seen earlier in the **Smile** example (Section 2.4). The importance of sampling diverse data points in kernel ridge regression to boost outlier predictive performance was also emphasized in the recent work [25].

4.3 | Kernel spectral clustering

We can also use RPCHOLESKY to accelerate *kernel spectral clustering* [58, 66]. We will study an application in molecular biophysics.

4.3.1 | Kernel clustering

In kernel spectral clustering, we use a positive-definite kernel function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{C}$ to compute similarities between data points $\boldsymbol{x}_1, \dots, \boldsymbol{x}_N$. Then we find a low-dimensional embedding $\boldsymbol{V} \in \mathbb{C}^{N \times m}$ of the N data points into m-dimensional Euclidean space that preserves the kernel-based similarities as well as possible. Afterward, we apply the conventional k-means algorithm [8] to cluster the rows of $\boldsymbol{V}$.

Specifically, the embedding matrix $\boldsymbol{V}$ is chosen to minimize the kernel-based distortion

$$\frac{1}{2} \sum_{i,j=1}^{N} K(\boldsymbol{x}_i, \boldsymbol{x}_j) \|\boldsymbol{V}(i,:) - \boldsymbol{V}(j,:)\|^2$$

while also satisfying the isotropy condition:

$$\sum_{i=1}^{N} \left(\sum_{j=1}^{N} K(\boldsymbol{x}_i, \boldsymbol{x}_j) \right) \boldsymbol{V}(i,:)^* \boldsymbol{V}(i,:) = \mathbf{I}.$$

The exact solution is described in [12]. We construct the symmetrically scaled transition matrix $\boldsymbol{H} = \boldsymbol{D}^{-1/2} \boldsymbol{A} \boldsymbol{D}^{-1/2}$, where $\boldsymbol{A} \in \mathbb{C}^{N \times N}$ is the kernel matrix and $\boldsymbol{D} \in \mathbb{C}^{N \times N}$ is the diagonal matrix that lists the row sums of $\boldsymbol{A}$. Then we calculate the m dominant eigenvectors $\boldsymbol{U} = \begin{bmatrix} \boldsymbol{u}_1 & \cdots & \boldsymbol{u}_m \end{bmatrix} \in \mathbb{C}^{N \times m}$ of the transition matrix $\boldsymbol{H}$. The optimal embedding matrix $\boldsymbol{V}$ is obtained from the diagonal rescaling $\boldsymbol{V} = \boldsymbol{D}^{-1/2} \boldsymbol{U}$. We note that there are several nonequivalent versions of spectral clustering; we use the present version because it is effective and widely used in biochemistry [32, 53] and amenable to acceleration by RPCHOLESKY.

4.3.2 | Accelerated kernel clustering via RPCHOLESKY

Directly computing the eigendecomposition of $\boldsymbol{H} = \boldsymbol{D}^{-1/2} \boldsymbol{A} \boldsymbol{D}^{-1/2}$ would require $\mathcal{O}(N^3)$ operations. However, there is a faster approach due to Fowlkes et al. [27] that requires just $\mathcal{O}(k^2 N)$ operations, where k is a parameter.

In this approach, we replace the kernel matrix $\boldsymbol{A}$ with a rank-k approximation $\widehat{\boldsymbol{A}}^{(k)}$ and replace the diagonal matrix $\boldsymbol{D}$ with the diagonal matrix $\widehat{\boldsymbol{D}}$ listing the row sums of $\widehat{\boldsymbol{A}}^{(k)}$. We form the approximate transition matrix $\widehat{\boldsymbol{H}} = \widehat{\boldsymbol{D}}^{-1/2} \widehat{\boldsymbol{A}}^{(k)} \widehat{\boldsymbol{D}}^{-1/2}$ and compute its m dominant eigenvectors $\widehat{\boldsymbol{U}} = \begin{bmatrix} \widehat{\boldsymbol{u}}_1 & \dots & \widehat{\boldsymbol{u}}_m \end{bmatrix}$. Just as before, we obtain an embedding $\widehat{\boldsymbol{V}} = \widehat{\boldsymbol{D}}^{-1/2} \widehat{\boldsymbol{U}}$, and we apply k-means clustering to the rows of $\widehat{\boldsymbol{V}}$.

Fowlkes et al. [27] used Nyström approximation with uniform sampling to obtain the low-rank approximation $\widehat{\boldsymbol{A}}^{(k)}$. We propose to replace uniform sampling with RPCHOLESKY, and we will demonstrate that this modification can significantly enhance the clustering accuracy. Our RPCHOLESKY-accelerated spectral clustering algorithm is presented in Algorithm 5.

4.3.3 | Alanine dipeptide trajectories

Kernel spectral clustering has become a popular approach for interpreting molecular dynamics (MD) data sets in computational biochemistry [32]. A typical MD data set consists of the

ALGORITHM 5 RPCHOLESKY-accelerated spectral clustering.

Input: Data points $\mathsf{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\} \subseteq \mathbb{C}^d$; eigenvector count m; approximation rank k

Output: Partition of labels $\{1, \dots, N\}$ into clusters $\mathsf{C}_1 \cup \cdots \cup \mathsf{C}_c$

$\boldsymbol{A} \leftarrow \mathit{KernelMatrix}(\mathsf{X})$

$\boldsymbol{F} \leftarrow \mathit{RPCholesky}(\boldsymbol{A}, k)$

$\widehat{\boldsymbol{D}} \leftarrow \mathrm{diag}(\boldsymbol{F}(\boldsymbol{F}^* \mathbb{1}))$

$\boldsymbol{G} \leftarrow \widehat{\boldsymbol{D}}^{-1/2} \boldsymbol{F}$

$(\boldsymbol{U}, \sim, \sim) \leftarrow \mathit{SVD}(\boldsymbol{G}, \texttt{'econ'})$

$\widehat{\boldsymbol{V}} \leftarrow \widehat{\boldsymbol{D}}^{-1/2} \boldsymbol{U}$

$\mathsf{C}_1, \dots, \mathsf{C}_c \leftarrow$ k-means$(\widehat{\boldsymbol{V}}(:, 1 : m))$ ▷ Cluster the rows of $\widehat{\boldsymbol{V}}(:, 1 : m)$

(x, y, z)-positions of the backbone (non-hydrogen) atoms in a simulated biomolecule. Spectral clustering is used to identify the metastable (long-lived) conformations of the molecule, which help determine the molecular functionality.

Alanine dipeptide ($CH_3 - CO - NH - C_\alpha HCH_3 - CO - NH - CH_3$) is a commonly studied molecule which has emerged as a benchmark when developing and testing numerical methods. The metastable states of alanine dipeptide are well-described by the dihedral angles ϕ between C, N, C_α, and C and ψ between N, C_α, C, and N. Yet even without access to this ϕ–ψ feature space, we can identify the metastable states by applying spectral clustering using the (x, y, z)-positions of the backbone atoms.

We downloaded one of the 250ns alanine dipeptide trajectories documented in [68], which includes the positions of the backbone atoms at intervals of 1ps, leading to $N = 2.5 \times 10^5$ data points in $\mathbb{R}^{30}$. Because of the large size of the data set, it would be extremely expensive to apply spectral clustering directly. To make these computations tractable, the biochemistry literature often applies subsampling to the data and then runs spectral clustering codes for a day or more on high-performance workstations [53, Section 4]. Here we document an alternative approach using RPCHOLESKY-accelerated spectral clustering, which uses all $N = 2.5 \times 10^5$ data points while retaining a modest computational cost (20 s on a laptop computer for $k = 150$).

In our approach, we first quantify the similarity between alanine dipeptide configurations using a Gaussian kernel with bandwidth $\sigma = 0.1$ nm. We then apply Algorithm 5 to form a low-dimensional embedding and find clusters in the data. Because the first three eigenvalues of $\widehat{\boldsymbol{D}}^{-1/2} \widehat{\boldsymbol{A}} \widehat{\boldsymbol{D}}^{-1/2}$ are much larger than the rest, we cluster based on the dominant $m = 3$ eigenvectors. We use the k-means algorithm to identify $c = 4$ clusters and present the results in Figure 3.

To measure the error, we obtain reference clusters by running RPCHOLESKY with $k = 1000$ columns. These reference clusters are consistent with [68, Figure 4]. For each method and each approximation rank $10 \le k \le 200$, we calculate the fraction of incorrect labelings for the best permutation of the cluster labels. We average over 1000 independent trials and plot the resulting errors in Figure 3 (top left). We find that RPCHOLESKY reliably produces a near-perfect clustering (top right panel) after reading just $k = 150$ of the $N = 2.5 \times 10^5$ columns. In contrast, given the same approximation rank, uniform, RLS, and greedy sampling frequently produce an incorrect clustering (bottom panels). With $k = 150$ columns, **RPCHOLESKY gives a 9× to 14× smaller misclassification rate** than uniform, RLS, or greedy sampling.

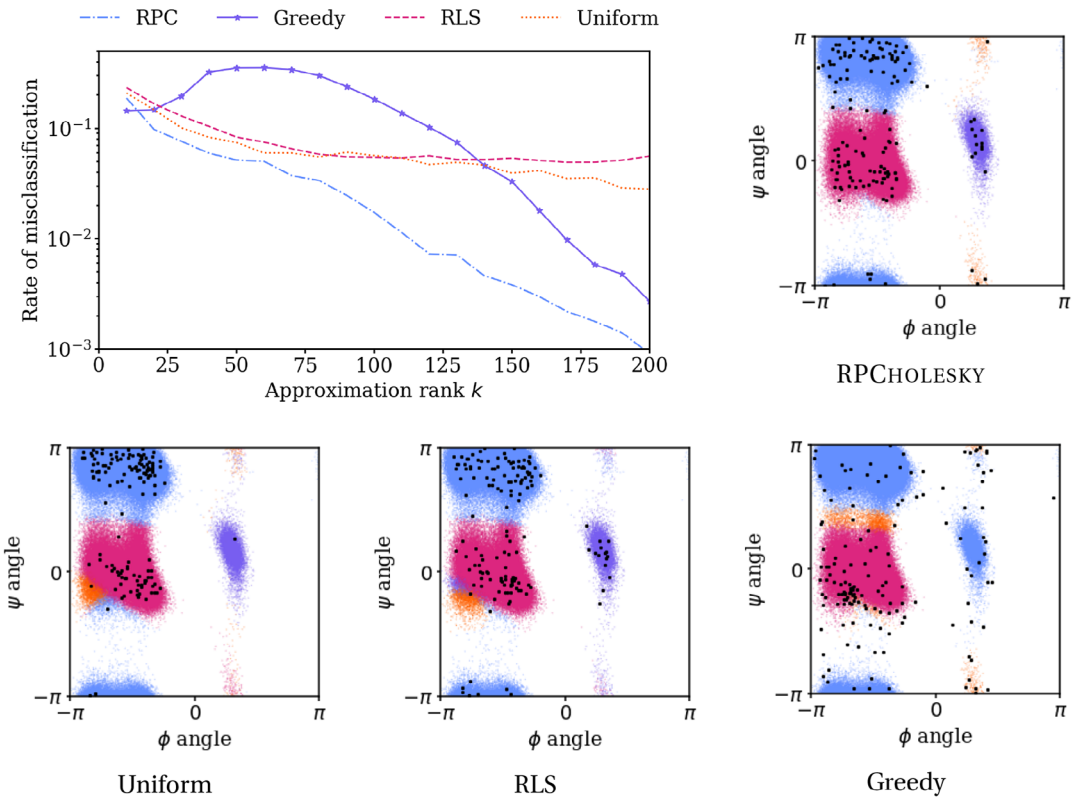

FIGURE 3 Spectral clustering for alanine dipeptide trajectories. *Top left*: Misclassification rate, averaged over 1000 independent trials. *Top right*: Example of correct clustering (< 0.2% misclassification) produced by RPCHOLESKY with rank $k = 150$. *Bottom*: Incorrect clusterings (> 2% misclassification) produced by uniform, RLS and greedy sampling with rank $k = 150$. Black dots mark data points selected as pivots. RLS, ridge leverage score; RPCholesky, randomly pivoted Cholesky algorithm.

4.3.4 | Comparison with neural networks

The recent journal article [6] compared an RPCHOLESKY-accelerated kernel method with a conventional neural network for the semisupervised clustering of alanine dipeptide trajectories. Given a fixed number of parameters, the RPCHOLESKY-accelerated approach was found to improve the accuracy, speed, and interpretability.

5 | THEORETICAL ANALYSIS OF RPCHOLESKY

Given the appealing computational profile and empirical performance of the RPCHOLESKY algorithm, we would like to understand when it is guaranteed to produce an accurate low-rank approximation. In this section, we will prove the following new result.

Theorem 5.1 (Randomly pivoted Cholesky). *Let $\boldsymbol{A}$ be a psd matrix. Fix $r \in \mathbb{N}$ and $\varepsilon > 0$. The rank-k column Nyström approximation $\widehat{\boldsymbol{A}}^{(k)}$ produced by k steps of* RPCHOLESKY *(Algorithm 1) attains*

the bound

$$\mathbb{E}\,\mathrm{tr}\big(\boldsymbol{A} - \widehat{\boldsymbol{A}}^{(k)}\big) \le (1+\varepsilon)\cdot \mathrm{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r),$$

provided that the number k of columns satisfies

$$k \ge \frac{r}{\varepsilon} + \min\left\{ r\log\left(\frac{1}{\varepsilon\eta}\right), r + r\log_+\left(\frac{2^r}{\varepsilon}\right)\right\}. \tag{5.1}$$

The relative error η *is defined by* $\eta := \mathrm{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r)/\mathrm{tr}(\boldsymbol{A})$. *As usual,* $\log_+(x) := \max\{\log x, 0\}$ *for* $x > 0$, *and the logarithm has base* e.

Let us emphasize that we do not need any prior knowledge to run RPCHOLESKY and attain this approximation guarantee. In fact, for any number k of steps, the error bound is valid for any pair (r, ε) that satisfies the relation (5.1).

The major takeaway from Theorem 5.1 is that RPCHOLESKY must produce an (r, ε)-approximation as soon as the number k of columns satisfies

$$k \ge \frac{r}{\varepsilon} + r\log\left(\frac{1}{\varepsilon\eta}\right).$$

The logarithmic factor is typically a modest constant. For example, $\log(1/\eta)$ is between 2.4 and 26 for all the examples in Table 1. Therefore, this bound is comparable with the minimal cost of $k \ge r/\varepsilon$ columns (Theorem C.1). Turning back to Table 2, we see that RPCHOLESKY improves on the uniform sampling method, where the number k of columns can depend *linearly* on the relative error $1/\eta$ (Section C.2). It also improves on the greedy method, in which the number k of columns may be proportional to the dimension N in the worst case (Section C.3).

Theorem 5.1 includes a second bound that demonstrates that RPCHOLESKY produces an (r, ε)-approximation when the number k of columns satisfies

$$k \ge \frac{r}{\varepsilon} + r + r\log_+\left(\frac{2^r}{\varepsilon}\right). \tag{5.2}$$

This alternative error bound is significant because it completely eliminates the dependence on the relative error η, at the cost of a quadratic dependence on r. We believe this quadratic dependence is a limitation of the proof technique, rather than a feature of the algorithm.

In the special case that $\mathrm{rank}(\boldsymbol{A}) \le r$, our analysis (Lemma 5.5) ensures that $\boldsymbol{A} = \widehat{\boldsymbol{A}}^{(k)}$ for any $k \ge r$. We note that both greedy and DPP sampling achieve a perfect approximation quality when $k \ge \mathrm{rank}(\boldsymbol{A})$, but the same is not true for uniform sampling or for RLS sampling.

Due to the close relationship between RPCHOLESKY and randomly pivoted QR (Section 3.1), our analysis also leads to the following new error bound:

Corollary 5.2 (Randomly pivoted QR). *Fix a matrix* $\boldsymbol{B} \in \mathbb{C}^{M\times N}$, *a target approximation rank r, and a tolerance* $\varepsilon > 0$. *The rank-k column projection approximation* $\widehat{\boldsymbol{B}}^{(k)}$ *produced by k steps of randomly pivoted QR (Algorithm B.2 with block size* $T = 1$*) attains the bound*

$$\mathbb{E}\|\boldsymbol{B} - \widehat{\boldsymbol{B}}^{(k)}\|_{\mathrm{F}}^2 \le (1+\varepsilon)\cdot \|\boldsymbol{B} - [\![\boldsymbol{B}]\!]_r\|_{\mathrm{F}}^2,$$

provided that the number k of columns satisfies

$$k \geq \frac{r}{\varepsilon} + \min\left\{ r \log\left(\frac{1}{\varepsilon\eta}\right), r + r \log_+\left(\frac{2^r}{\varepsilon}\right) \right\}.$$

The relative error η is defined by $\eta := \|\boldsymbol{B} - [\![\boldsymbol{B}]\!]_r\|_{\mathrm{F}}^2 / \|\boldsymbol{B}\|_{\mathrm{F}}^2$.

In follow-up work [24], we extended these results to address block randomly pivoted QR (Algorithm B.2) and block RPCHOLESKY (Algorithm 3) with a fixed block size $T > 1$. This analysis is more complicated and requires several additional ideas.

5.1 | Proof of Theorem 5.1

Let $\boldsymbol{A}$ be a psd input matrix. Recall the definition (2.3) of the approximation $\widehat{\boldsymbol{A}}^{(i)}$ and residual $\boldsymbol{A}^{(i)}$ matrices generated by the pivoted partial Cholesky algorithm. Recall that RPCHOLESKY samples each pivot from the distribution (2.6).

The proof of Theorem 5.1 is based on the properties of the *expected residual function*:

$$\boldsymbol{\Phi}(\boldsymbol{A}) := \mathbb{E}\left[\boldsymbol{A}^{(1)} \,\middle|\, \boldsymbol{A}\right] = \boldsymbol{A} - \frac{\boldsymbol{A}^2}{\operatorname{tr} \boldsymbol{A}}. \tag{5.3}$$

This function returns the expectation of the residual $\boldsymbol{A}^{(1)}$ after applying one step of RPCHOLESKY to the psd matrix $\boldsymbol{A}$. The equality (5.3) follows from a short computation using the sampling distribution (2.6) and the definition (2.3) of the residual. Note that $\boldsymbol{\Phi}$ is defined on psd matrices of any dimension.

The first lemma describes some basic facts about the expected residual function $\boldsymbol{\Phi}$. We postpone the proof to Section 5.2.

Lemma 5.3 (Expected residual). *The expected residual map $\boldsymbol{\Phi}$ defined in (5.3) is positive, monotone, and concave with respect to the psd order. That is, for all psd $\boldsymbol{A}, \boldsymbol{H}$ with the same dimensions,*

$$\mathbf{0} \preceq \boldsymbol{\Phi}(\boldsymbol{A}) \preceq \boldsymbol{\Phi}(\boldsymbol{A} + \boldsymbol{H}); \tag{5.4}$$

$$\theta\boldsymbol{\Phi}(\boldsymbol{A}) + (1-\theta)\boldsymbol{\Phi}(\boldsymbol{H}) \preceq \boldsymbol{\Phi}(\theta\boldsymbol{A} + (1-\theta)\boldsymbol{H}) \quad \text{for all } \theta \in [0,1]. \tag{5.5}$$

The second lemma describes how the trace of the residual declines after multiple steps of the RPCHOLESKY procedure. This is the key new ingredient in our argument. The proof appears in Section 5.3.

Lemma 5.4 (Contraction rate). *Consider the k-fold composition of the expected residual (5.3):*

$$\boldsymbol{\Phi}^{\circ k} := \underbrace{\boldsymbol{\Phi} \circ \boldsymbol{\Phi} \circ \cdots \circ \boldsymbol{\Phi}}_{k \text{ times}} \quad \text{for each } k \in \mathbb{N}.$$

Fix $r \in \mathbb{N}$. For each psd matrix $\boldsymbol{H}$ and each $\Delta > 0$,

$$\operatorname{tr} \boldsymbol{\Phi}^{\circ k}(\boldsymbol{H}) \leq \operatorname{tr}(\boldsymbol{H} - [\![\boldsymbol{H}]\!]_r) + \Delta \operatorname{tr} \boldsymbol{H} \quad \textit{when} \quad k \geq \frac{r \operatorname{tr}(\boldsymbol{H} - [\![\boldsymbol{H}]\!]_r)}{\Delta \operatorname{tr} \boldsymbol{H}} + r \log_+\left(\frac{1}{\Delta}\right).$$

Last, we present a bound which shows that the error after k steps of RPCHOLESKY is comparable with the error in the best rank-k approximation. This lemma improves on an earlier result [19, Proposition 2] by reducing a $(k+1)!$ factor to 2^k, but the proof is entirely different. This bound is responsible for the final term in eq. (5.2), and the proof appears in Section 5.4.

Lemma 5.5 (Error doubling). *For each psd matrix $\boldsymbol{A}$, the residual matrix $\boldsymbol{A}^{(k)}$ after applying k steps of* RPCHOLESKY *satisfies*

$$\mathbb{E} \operatorname{tr} \boldsymbol{A}^{(k)} \leq 2^k \operatorname{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_k) \quad \textit{for each } k \in \mathbb{N}.$$

With these results at hand, we quickly establish the main error bound for RPCHOLESKY.

Proof of Theorem 5.1. Fix a psd matrix $\boldsymbol{A}$. By the concavity (5.5) of the expected residual map (5.3) and a matrix version of Jensen's inequality [14, Theorem 4.16], the residual matrices satisfy

$$\mathbb{E}\boldsymbol{A}^{(j)} = \mathbb{E}\big[\mathbb{E}\big[\boldsymbol{A}^{(j)} \,\big|\, \boldsymbol{A}^{(j-1)}\big]\big] = \mathbb{E}\boldsymbol{\Phi}(\boldsymbol{A}^{(j-1)}) \preceq \boldsymbol{\Phi}(\mathbb{E}\boldsymbol{A}^{(j-1)}) \quad \text{for each } j \in \{1, \dots, k\}.$$

Next, by monotonicity (5.4) of $\boldsymbol{\Phi}$ and the last display,

$$\mathbb{E}\boldsymbol{A}^{(k)} \preceq \boldsymbol{\Phi}(\mathbb{E}\boldsymbol{A}^{(k-1)}) \preceq \boldsymbol{\Phi} \circ \boldsymbol{\Phi}(\mathbb{E}\boldsymbol{A}^{(k-2)}) \preceq \cdots \preceq \boldsymbol{\Phi}^{\circ k}(\boldsymbol{A}).$$

Using the fact that the trace is linear and preserves the psd order,

$$\mathbb{E} \operatorname{tr}\big(\boldsymbol{A} - \widehat{\boldsymbol{A}}^{(k)}\big) = \mathbb{E} \operatorname{tr} \boldsymbol{A}^{(k)} \leq \operatorname{tr} \boldsymbol{\Phi}^{\circ k}(\boldsymbol{A}).$$

Next, we apply Lemma 5.4 with $\boldsymbol{H} = \boldsymbol{A}$ and $\Delta = \varepsilon\eta$ to see that

$$\mathbb{E} \operatorname{tr}\big(\boldsymbol{A} - \widehat{\boldsymbol{A}}^{(k)}\big) \leq (1+\varepsilon) \cdot \operatorname{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r) \quad \text{when} \quad k \geq \frac{r}{\varepsilon} + r \log_+\left(\frac{1}{\varepsilon\eta}\right),$$

where we have recognized the relative error η. Last, we can replace $\log_+$ with log because the statement holds trivially for any Nyström approximation with any number of columns if $\varepsilon\eta \geq 1$.

By a similar argument,

$$\mathbb{E} \operatorname{tr}\big(\boldsymbol{A} - \widehat{\boldsymbol{A}}^{(k)}\big) = \mathbb{E} \operatorname{tr} \boldsymbol{A}^{(k)} \leq \operatorname{tr} \boldsymbol{\Phi}^{\circ(k-r)}\big(\mathbb{E}\boldsymbol{A}^{(r)}\big).$$

We can apply Lemma 5.4 with $\boldsymbol{H} = \mathbb{E}\boldsymbol{A}^{(r)}$ and $\Delta = \varepsilon\eta \cdot \operatorname{tr}\boldsymbol{A} / \mathbb{E} \operatorname{tr} \boldsymbol{A}^{(r)}$ to see that

$$\mathbb{E} \operatorname{tr}\big(\boldsymbol{A} - \widehat{\boldsymbol{A}}^{(k)}\big) \leq \operatorname{tr}\big(\mathbb{E}\boldsymbol{A}^{(r)} - [\![\mathbb{E}\boldsymbol{A}^{(r)}]\!]_r\big) + \varepsilon \operatorname{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r) \tag{5.6}$$

provided the number of columns satisfies

$$k - r \geq \frac{r \operatorname{tr}\big(\mathbb{E}\boldsymbol{A}^{(r)} - [\![\mathbb{E}\boldsymbol{A}^{(r)}]\!]_r\big)}{\varepsilon \operatorname{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r)} + r \log_+\left(\frac{\mathbb{E} \operatorname{tr} \boldsymbol{A}^{(r)}}{\varepsilon \operatorname{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r)}\right). \tag{5.7}$$

This bound can be simplified as follows. Observe that the (random) residual matrix $\boldsymbol{A}^{(r)}$ is a Schur complement of $\boldsymbol{A}$, so it satisfies $\mathbb{E}\boldsymbol{A}^{(r)} \preceq \boldsymbol{A}$. By the Ky Fan variational principle [71, Theorem 8.17], the best rank-r approximation error in the trace norm is monotone with respect to the psd order, so

$$\operatorname{tr}\bigl(\mathbb{E}\boldsymbol{A}^{(r)} - [\![\mathbb{E}\boldsymbol{A}^{(r)}]\!]_r\bigr) \le \operatorname{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r).$$

Additionally, Lemma 5.5 guarantees that $\mathbb{E}\operatorname{tr}\boldsymbol{A}^{(r)} \le 2^r \operatorname{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r)$. Using these facts, we can simplify (5.6)–(5.7) to show that

$$\mathbb{E}\operatorname{tr}\bigl(\boldsymbol{A} - \widehat{\boldsymbol{A}}^{(k)}\bigr) \le (1+\varepsilon)\cdot\operatorname{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r) \quad \text{when} \quad k \ge r + \frac{r}{\varepsilon} + r\log_+\left(\frac{2^r}{\varepsilon}\right),$$

This completes the proof of the second bound. □

5.2 | Proof of Lemma 5.3

Let $\boldsymbol{A}, \boldsymbol{H}$ be psd matrices, and recall the definition (5.3) of the expected residual map $\boldsymbol{\Phi}$. First, to prove that $\boldsymbol{\Phi}$ is positive, note that

$$\boldsymbol{\Phi}(\boldsymbol{H}) = \left(\mathbf{I} - \frac{\boldsymbol{H}}{\operatorname{tr}\boldsymbol{H}}\right)\boldsymbol{H} \succeq \mathbf{0}.$$

Next, to establish concavity, we choose $\theta \in [0,1]$, set $\overline{\theta} := 1-\theta$, and make the calculation

$$\boldsymbol{\Phi}(\theta\boldsymbol{A} + \overline{\theta}\boldsymbol{H}) - \theta\boldsymbol{\Phi}(\boldsymbol{A}) - \overline{\theta}\boldsymbol{\Phi}(\boldsymbol{H}) = \frac{\theta\overline{\theta}}{\theta\operatorname{tr}\boldsymbol{A} + \overline{\theta}\operatorname{tr}\boldsymbol{H}}\left(\sqrt{\frac{\operatorname{tr}\boldsymbol{H}}{\operatorname{tr}\boldsymbol{A}}}\boldsymbol{A} - \sqrt{\frac{\operatorname{tr}\boldsymbol{A}}{\operatorname{tr}\boldsymbol{H}}}\boldsymbol{H}\right)^2 \succeq \mathbf{0}.$$

Last, to establish monotonicity, observe that $\boldsymbol{\Phi}$ is positive homogeneous; that is, $\boldsymbol{\Phi}(\tau\boldsymbol{A}) = \tau\boldsymbol{\Phi}(\boldsymbol{A})$ for $\tau \ge 0$. Invoking the concavity property (5.5) with $\theta = 1/2$,

$$\boldsymbol{\Phi}(\boldsymbol{A}+\boldsymbol{H}) = 2\,\boldsymbol{\Phi}\left(\frac{\boldsymbol{A}+\boldsymbol{H}}{2}\right) \succeq \boldsymbol{\Phi}(\boldsymbol{A}) + \boldsymbol{\Phi}(\boldsymbol{H}) \succeq \boldsymbol{\Phi}(\boldsymbol{A}).$$

We have used the positivity of $\boldsymbol{\Phi}(\boldsymbol{H})$ in the last step.

5.3 | Proof of Lemma 5.4

We break the proof into several steps.

5.3.1 | Step 1: Reduction to diagonal case

First, we show that it suffices to consider the case of a diagonal matrix. Let $\boldsymbol{H}$ be an $N \times N$ psd matrix with eigendecomposition $\boldsymbol{H} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^*$. The definition (5.3) of the expected residual map implies that

$$\boldsymbol{\Phi}(\boldsymbol{H}) = \boldsymbol{V}\boldsymbol{\Phi}(\boldsymbol{\Lambda})\boldsymbol{V}^*.$$

By iteration, the same relation holds with $\boldsymbol{\Phi}^{\circ k}$ in place of $\boldsymbol{\Phi}$. In particular, $\operatorname{tr}\boldsymbol{\Phi}^{\circ k}(\boldsymbol{H}) = \operatorname{tr}\boldsymbol{\Phi}^{\circ k}(\boldsymbol{\Lambda})$. Therefore, we may restrict our attention to the diagonal case where $\boldsymbol{H} = \boldsymbol{\Lambda}$.

5.3.2 | Step 2: Identification of worst-case matrix

Second, we obtain an upper bound on $\operatorname{tr}\boldsymbol{\Phi}^{\circ k}(\boldsymbol{\Lambda})$. We accomplish this goal by identifying the worst-case set of eigenvalues. Because the map $\boldsymbol{\Lambda} \mapsto \operatorname{tr}\boldsymbol{\Phi}^{\circ k}(\boldsymbol{\Lambda})$ is concave and invariant under permutations of its arguments, averaging together some of the eigenvalues $\lambda_1, \dots, \lambda_N$ of $\boldsymbol{\Lambda}$ can only increase the value of $\operatorname{tr}\boldsymbol{\Phi}^{\circ k}(\boldsymbol{\Lambda})$. Therefore, by introducing the function

$$f_k(a, b, r, q) := \operatorname{tr}\boldsymbol{\Phi}^{\circ k}\left(\operatorname{diag}\left(\underbrace{\frac{a}{r}, \dots, \frac{a}{r}}_{r \text{ times}}, \underbrace{\frac{b}{q}, \dots, \frac{b}{q}}_{q \text{ times}}\right)\right),$$

we obtain the upper bound

$$\operatorname{tr}\boldsymbol{\Phi}^{\circ k}(\boldsymbol{\Lambda}) \le f_k(\operatorname{tr}[\![\boldsymbol{\Lambda}]\!]_r, \operatorname{tr}(\boldsymbol{\Lambda} - [\![\boldsymbol{\Lambda}]\!]_r), r, N - r). \tag{5.8}$$

For further reference, we note that $f_k(a, b, r, q)$ is weakly increasing as a function of q. Indeed, for every tuple (a, b, r, q),

$$\begin{aligned} f_k(a, b, r, q) &= \operatorname{tr}\boldsymbol{\Phi}^{\circ k}\left(\operatorname{diag}\left(\underbrace{\frac{a}{r}, \dots, \frac{a}{r}}_{r \text{ times}}, \underbrace{\frac{b}{q}, \dots, \frac{b}{q}}_{q \text{ times}}\right)\right) = \operatorname{tr}\boldsymbol{\Phi}^{\circ k}\left(\operatorname{diag}\left(\underbrace{\frac{a}{r}, \dots, \frac{a}{r}}_{r \text{ times}}, \underbrace{\frac{b}{q}, \dots, \frac{b}{q}}_{q \text{ times}}, 0\right)\right) \\ &\le \operatorname{tr}\boldsymbol{\Phi}^{\circ k}\left(\operatorname{diag}\left(\underbrace{\frac{a}{r}, \dots, \frac{a}{r}}_{r \text{ times}}, \underbrace{\frac{b}{q+1}, \dots, \frac{b}{q+1}}_{q+1 \text{ times}}\right)\right) = f_k(a, b, r, q+1). \end{aligned}$$

We have exploited the fact that $\boldsymbol{\Phi}$ is defined for matrices of every dimension.

5.3.3 | Step 3: Dynamics of the error

Next, we derive a worst-case expression for the error $f_k(a, b, r, q)$. We accomplish this task by identifying a discrete-time dynamical system that describes the evolution of the residuals. For each $k = 0, 1, 2, \dots$, define the nonnegative quantities $a^{(k)}$ and $b^{(k)}$ via the relation

$$\boldsymbol{\Phi}^{\circ k}\left(\operatorname{diag}\left(\underbrace{\frac{a}{r}, \dots, \frac{a}{r}}_{r \text{ times}}, \underbrace{\frac{b}{q}, \dots, \frac{b}{q}}_{q \text{ times}}\right)\right) =: \operatorname{diag}\left(\underbrace{\frac{a^{(k)}}{r}, \dots, \frac{a^{(k)}}{r}}_{r \text{ times}}, \underbrace{\frac{b^{(k)}}{q}, \dots, \frac{b^{(k)}}{q}}_{q \text{ times}}\right).$$

By the definition (5.3) of the expected residual map $\boldsymbol{\Phi}$, the quantities $a^{(k)}$ and $b^{(k)}$ satisfy the recurrence relations

$$a^{(k)} - a^{(k-1)} = -\frac{\left(a^{(k-1)}\right)^2}{r\left(a^{(k-1)} + b^{(k-1)}\right)} \quad \text{and} \quad b^{(k)} - b^{(k-1)} = -\frac{\left(b^{(k-1)}\right)^2}{q\left(a^{(k-1)} + b^{(k-1)}\right)}$$

with initial conditions $a^{(0)} = a$ and $b^{(0)} = b$. This construction guarantees $f_k(a, b, r, q) = a^{(k)} + b^{(k)}$. Additionally, the quantities $a^{(k)}$ and $b^{(k)}$ converge as $q \to \infty$ to limiting values $\overline{a}^{(k)}$ and $\overline{b}^{(k)} \equiv b$, where the sequence $\overline{a}^{(k)}$ satisfies

$$\overline{a}^{(k)} - \overline{a}^{(k-1)} = -\frac{\big(\overline{a}^{(k-1)}\big)^2}{r\big(\overline{a}^{(k-1)} + b\big)} \quad \text{with initial condition } \overline{a}^{(0)} = a.$$

It follows that

$$f_k(a, b, r, q) \le \overline{a}^{(k)} + b \le a + b.$$

We have used the facts that $f_k(a, b, r, q)$ is increasing in q and $\overline{a}^{(k)}$ is decreasing in k.

5.3.4 | Step 4: Comparison with continuous-time dynamics

All that remains is to determine how quickly $\overline{a}^{(k)}$ decreases as a function of k. To that end, we pass from discrete time to continuous time. At each instant $t = 0, 1, 2, \ldots$, the discrete-time process $\overline{a}^{(t)}$ is bounded from above by the continuous-time process $x(t)$ satisfying the ODE

$$\frac{\mathrm{d}}{\mathrm{d}t} x(t) = -\frac{x(t)^2}{r(x(t) + b)} \quad \text{with initial condition } x(0) = a.$$

The comparison between discrete- and continuous-time processes holds because $x \mapsto -x^2/(rx + rb)$ is decreasing over the range $x \in (0, \infty)$. Next, assuming $\Delta(a + b) \le a$, we can use separation of variables to solve for the time $t_\star$ at which $x(t_\star) = \Delta(a + b)$:

$$t_\star = \int_{x=a}^{\Delta(a+b)} -\frac{r(x+b)}{x^2}\,\mathrm{d}x = rb\left(\frac{1}{\Delta(a+b)} - \frac{1}{a}\right) + r\log\left(\frac{a}{\Delta(a+b)}\right) \le \frac{rb}{\Delta(a+b)} + r\log\left(\frac{1}{\Delta}\right).$$

It follows (even for $\Delta(a + b) > a$) that

$$\overline{a}^{(k)} \le \Delta(a + b) \quad \text{when} \quad k \ge \frac{rb}{\Delta(a+b)} + r\log_+\left(\frac{1}{\Delta}\right).$$

To conclude, substitute $a = \operatorname{tr}[\![\boldsymbol{\Lambda}]\!]_r$ and $b = \operatorname{tr}(\boldsymbol{\Lambda} - [\![\boldsymbol{\Lambda}]\!]_r)$ and combine with (5.8).

5.4 | Proof of Lemma 5.5

Let $\boldsymbol{P}$ denote the orthogonal projection onto the k dominant eigenvectors of $\boldsymbol{A}$ and set $\boldsymbol{P}_\perp := \mathbf{I} - \boldsymbol{P}$. Apply the Ky Fan variational principle [71, Theorem 8.17] to write

$$\operatorname{tr}\big(\boldsymbol{A}^{(1)} - [\![\boldsymbol{A}^{(1)}]\!]_{k-1}\big) = \min\left\{ \sum_{j=k}^{N} \boldsymbol{u}_j^* \boldsymbol{A}^{(1)} \boldsymbol{u}_j \;:\; \boldsymbol{u}_k, \ldots, \boldsymbol{u}_N \text{ orthonormal} \right\}.$$

Choose the vectors $\boldsymbol{u}_{k+1}, \dots, \boldsymbol{u}_N$ to be unit-norm eigenvectors $\boldsymbol{v}_{k+1}(\boldsymbol{A}), \dots, \boldsymbol{v}_N(\boldsymbol{A})$ associated with the smallest eigenvalues. To choose the last vector $\boldsymbol{u}_k$, let us separately consider the cases $\boldsymbol{P}(s_1, s_1) = 0$ and $\boldsymbol{P}(s_1, s_1) > 0$, where $\boldsymbol{P}(i, i)$ denotes the (i, i)-entry of $\boldsymbol{P}$.

On the event $\boldsymbol{P}(s_1, s_1) = 0$, choose $\boldsymbol{u}_k := \boldsymbol{v}_1(\boldsymbol{A})$ and apply the crude bound

$$\begin{aligned} \operatorname{tr}\big(\boldsymbol{A}^{(1)} - [\![\boldsymbol{A}^{(1)}]\!]_{k-1}\big) &\le \sum_{j=k}^{N} \boldsymbol{u}_j^* \boldsymbol{A}^{(1)} \boldsymbol{u}_j \\ &\le \sum_{j=k}^{N} \boldsymbol{u}_j^* \boldsymbol{A} \boldsymbol{u}_j = \operatorname{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_k) + \lambda_1(\boldsymbol{A}). \end{aligned} \tag{5.9}$$

This relation holds because $\boldsymbol{A}^{(1)} \preceq \boldsymbol{A}$ and $\sum_{j=k+1}^{N} \lambda_j(\boldsymbol{A}) = \operatorname{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_k)$.

On the event $\boldsymbol{P}(s_1, s_1) > 0$, choose $\boldsymbol{u}_k := \boldsymbol{P}\mathbf{e}_{s_1} / \|\boldsymbol{P}\mathbf{e}_{s_1}\|$, and observe that $\boldsymbol{u}_k$ is orthonormal to the other vectors by the choice of $\boldsymbol{P}$. This gives the bound

$$\begin{aligned} \operatorname{tr}\big(\boldsymbol{A}^{(1)} - [\![\boldsymbol{A}^{(1)}]\!]_{k-1}\big) &\le \sum_{j=k+1}^{N} \boldsymbol{u}_j^* \boldsymbol{A}^{(1)} \boldsymbol{u}_j + \frac{\mathbf{e}_{s_1}^* \boldsymbol{P} \boldsymbol{A}^{(1)} \boldsymbol{P} \mathbf{e}_{s_1}}{\mathbf{e}_{s_1}^* \boldsymbol{P} \mathbf{e}_{s_1}} \\ &\le \operatorname{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_k) + \frac{(\boldsymbol{P}\boldsymbol{A}^{(1)}\boldsymbol{P})(s_1, s_1)}{\boldsymbol{P}(s_1, s_1)}. \end{aligned} \tag{5.10}$$

Using the facts that $\boldsymbol{A}^{(1)} = \boldsymbol{A} - \boldsymbol{A}\mathbf{e}_{s_1}\mathbf{e}_{s_1}^*\boldsymbol{A}/\boldsymbol{A}(s_1, s_1)$ and $\boldsymbol{P}\boldsymbol{A} = \boldsymbol{A}\boldsymbol{P}$, it follows

$$(\boldsymbol{P}\boldsymbol{A}^{(1)}\boldsymbol{P})(s_1, s_1) = (\boldsymbol{A}\boldsymbol{P})(s_1, s_1) - \frac{((\boldsymbol{A}\boldsymbol{P})(s_1, s_1))^2}{\boldsymbol{A}(s_1, s_1)} = \frac{(\boldsymbol{A}\boldsymbol{P})(s_1, s_1)(\boldsymbol{A}\boldsymbol{P}_\perp)(s_1, s_1)}{\boldsymbol{A}(s_1, s_1)}.$$

Combine the bounds (5.9) and (5.10) and sum over the selection probabilities $\mathbb{P}\{s_1 = i\} = \boldsymbol{A}(i, i)/\operatorname{tr}\boldsymbol{A}$ to evaluate

$$\begin{aligned} \mathbb{E}\big[\operatorname{tr}\big(\boldsymbol{A}^{(1)} - [\![\boldsymbol{A}^{(1)}]\!]_{k-1}\big)\big] &\le \operatorname{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_k) + \sum_{\boldsymbol{P}(i,i)=0} \frac{\boldsymbol{A}(i,i)}{\operatorname{tr}\boldsymbol{A}} \cdot \lambda_1(\boldsymbol{A}) + \sum_{\boldsymbol{P}(i,i)>0} \frac{\boldsymbol{A}(i,i)}{\operatorname{tr}\boldsymbol{A}} \cdot \frac{(\boldsymbol{A}\boldsymbol{P})(i,i)(\boldsymbol{A}\boldsymbol{P}_\perp)(i,i)}{\boldsymbol{A}(i,i)\boldsymbol{P}(i,i)} \\ &\le \operatorname{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_k) + \sum_{i=1}^{N} \frac{(\boldsymbol{A}\boldsymbol{P}_\perp)(i,i)}{\operatorname{tr}\boldsymbol{A}} \cdot \lambda_1(\boldsymbol{A}) \\ &= \left(1 + \frac{\lambda_1(\boldsymbol{A})}{\operatorname{tr}\boldsymbol{A}}\right) \operatorname{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_k). \end{aligned} \tag{5.11}$$

The middle line uses the inequalities

$$\boldsymbol{P}\mathbf{e}_i = \mathbf{0} \Rightarrow \boldsymbol{A}(i, i) = (\boldsymbol{A}\boldsymbol{P})(i, i) + (\boldsymbol{A}\boldsymbol{P}_\perp)(i, i) = (\boldsymbol{A}\boldsymbol{P}_\perp)(i, i).$$

and $(\boldsymbol{A}\boldsymbol{P})(i, i) \le \lambda_1(\boldsymbol{A})\boldsymbol{P}(i, i)$.

Since $\lambda_1(\boldsymbol{A}) \le \operatorname{tr}\boldsymbol{A}$, the bound (5.11) can be weakened to give

$$\mathbb{E}\operatorname{tr}\big(\boldsymbol{A}^{(1)} - [\![\boldsymbol{A}^{(1)}]\!]_{k-1}\big) \le 2\operatorname{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_k).$$

By iterating, we conclude that

$$\mathbb{E}\,\mathrm{tr}\,\boldsymbol{A}^{(k)} \le 2\mathbb{E}\,\mathrm{tr}\big(\boldsymbol{A}^{(k-1)} - [\![\boldsymbol{A}^{(k-1)}]\!]_1\big) \le 4\mathbb{E}\,\mathrm{tr}\big(\boldsymbol{A}^{(k-2)} - [\![\boldsymbol{A}^{(k-2)}]\!]_2\big) \le \cdots \le 2^k\,\mathrm{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_k).$$

This estimate is the statement of the doubling bound.

6 | CONCLUSION

This work has demonstrated the utility of RPCHOLESKY for low-rank approximation of a psd matrix $\boldsymbol{A} \in \mathbb{C}^{N\times N}$. RPCHOLESKY allows us to accelerate many kernel algorithms, such as kernel ridge regression and kernel spectral clustering. RPCHOLESKY reduces the computational cost of these kernel methods from $\mathcal{O}(N^3)$ operations to just $\mathcal{O}(k^2N)$ operations, where the approximation rank k can be much smaller than the matrix dimension N.

Numerical experiments suggest that RPCHOLESKY improves over other column Nyström approximation algorithms in terms of arithmetic operations and memory footprint. Given a fixed approximation rank k, RPCHOLESKY requires a very small number of entry evaluations, just $(k+1)N$. RPCHOLESKY typically produces approximations that match or improve on the greedy method, uniform sampling, RLS sampling, and DPP sampling. Moreover, theoretical error bounds guarantee that RPCHOLESKY converges nearly as fast as possible in the expected trace norm.

Taken as a whole, this work paves the way for greater use of RPCHOLESKY in the future. Additionally, we believe RPCHOLESKY can be pushed even further, for example, through combinations with accelerated methods for prediction, clustering, and other learning tasks. The future is indeed bright for this simple yet surprisingly effective algorithm.

ACKNOWLEDGMENTS

We thank Mateo Díaz, Zachary Frangella, Marc Gilles, Eitan Levin, Eliza O'Reilly, Jonathan Weare, and Aaron Dinner for helpful discussions and corrections. Y.C. acknowledges support from the Caltech Kortschak Scholar program and the Courant Instructorship. E.N.E acknowledges support from the Department of Energy through Award DE-SC0021110. J.A.T and R.J.W acknowledge support from the Office of Naval Research through BRC Award N00014-18-1-2363 and from the National Science Foundation through FRG Award 1952777.

REFERENCES

1. A. Abedsoltan, M. Belkin, and P. Pandit, *Toward large kernel models*, Proceedings of the 40th International Conference on Machine Learning, JMLR.org, 2023, pp. 61–78. URL https://dl.acm.org/doi/10.5555/3618408.3618412.
2. A. Alaoui and M. W. Mahoney, *Fast randomized kernel ridge regression with statistical guarantees*, Proceedings of the 28th International Conference on Neural Information Processing Systems, MIT Press, 2015, 775–783. URL https://dl.acm.org/doi/10.5555/2969239.2969326.
3. J. M. Altschuler and P. A. Parrilo, *Kernel approximation on algebraic varieties*, SIAM J. Appl. Algebra Geom. **7** (2023), no. 1, 1–28, DOI 10.1137/21M1425050.
4. N. Anari, S. Oveis Gharan, and A. Rezaei, *Monte Carlo Markov chain algorithms for sampling strongly Rayleigh distributions and determinantal point processes*, 29th Annual Conference on Learning Theory, PMLR, 2016, 49:103-115. URL https://proceedings.mlr.press/v49/anari16.html.
5. T. Ando, Schur complements and matrix inequalities: Operator-theoretic approach, F. Zhang (ed.), The Schur Complement and Its Applications, Numerical Methods and Algorithms, Springer, 2005, pp. 137–162, DOI 10.1007/0-387-24273-2_6.

6. D. Aristoff, M. Johnson, G. Simpson, and R. J. Webber, *The fast committor machine: Interpretable prediction with kernels*, J. Chem. Phys. **161** (2024), no. 8, 084113, DOI 10.1063/5.0222798.
7. S. Arora, S. S. Du, Z. Li, R. Salakhutdinov, R. Wang, and D. Yu, *Harnessing the power of infinitely wide deep nets on small-data tasks*, International Conference on Learning Representations, 2020, URL https://openreview.net/forum?id=rkl8sJBYvH.
8. D. Arthur and S. Vassilvitskii, *k-means++: The advantages of careful seeding*, Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2007, URL https://dl.acm.org/doi/10.5555/1283383.1283494.
9. F. R. Bach and M. I. Jordan, *Predictive low-rank decomposition for kernel methods*, Proceedings of the 22nd International Conference on Machine Learning, 2005, DOI 10.1145/1102351.1102356.
10. P. Batlle, M. Darcy, B. Hosseini, and H. Owhadi, *Kernel methods are competitive for operator learning*, J. Comput. Phys. **496** (2024), 112549, DOI 10.1016/j.jcp.2023.112549.
11. M.-A. Belabbas and P. J. Wolfe, *Spectral methods in machine learning and new strategies for very large datasets*, Proc. Natl. Acad. Sci. **106** (2009), no. 2, 369–374, DOI 10.1073/pnas.0810600105.
12. M. Belkin and P. Niyogi, *Laplacian eigenmaps for dimensionality reduction and data representation*, Neural Comput. **15** (2003), no. 6, 1373–1396, DOI 10.1162/089976603321780317.
13. D. Calandriello, A. Lazaric, and M. Valko, *Distributed adaptive sampling for kernel matrix approximation*, Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, 2017, URL https://proceedings.mlr.press/v54/calandriello17a.html.
14. E. Carlen, Trace inequalities and quantum entropy: An introductory course, R. Sims and D. Ueltschi (eds.), Entropy and the Quantum, Contemporary Mathematics, vol. 529, American Mathematical Society, 2010, pp. 73–140, DOI 10.1090/conm/529.
15. C.-C. Chang and C.-J. Lin, *LIBSVM: A library for support vector machines*, ACM Trans. Intell. Syst. Technol. **2** (2011), no. 3, 1–27, DOI 10.1145/1961189.1961199.
16. D. E. Crabtree and E. V. Haynsworth, *An identity for the Schur complement of a matrix*, Proc. Amer. Math. Soc. **22** (1969), no. 2, 364–366. URL http://www.jstor.org/stable/2037057.
17. M. Dereziński and M. W. Mahoney, *Determinantal point processes in randomized numerical linear algebra*, Notices Amer. Math. Soc. **68** (2021), no. 01, 1, DOI 10.1090/noti2202.
18. M. Derezinski, D. Calandriello, and M. Valko, *Exact sampling of determinantal point processes with sublinear time preprocessing*, Proceedings of the 33rd International Conference on Neural Information Processing Systems, 2019, URL https://dl.acm.org/doi/10.5555/3454287.3455323.
19. A. Deshpande and S. Vempala, Adaptive sampling and fast low-rank matrix approximation, J. Díaz, K. Jansen, J. D. P. Rolim, and U. Zwick (eds.), Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, Lecture Notes in Computer Science, Springer, 2006, pp. 292–303, DOI 10.1007/11830924_28.
20. A. Deshpande, L. Rademacher, S. S. Vempala, and G. Wang, *Matrix approximation and projective clustering via volume sampling*, Theory Comput. **2** (2006), no. 12, 225–247, DOI 10.4086/toc.2006.v002a012.
21. P. Drineas and M. W. Mahoney, *On the Nyström method for approximating a Gram matrix for improved kernel-based learning*, J. Mach. Learn. Res. **6** (2005), no. 72, 2153–2175.
22. D. Dua and C. Graff, *UCI machine learning repository*, 2017, URL http://archive.ics.uci.edu/ml.
23. M. Díaz, E. N. Epperly, Z. Frangella, J. A. Tropp, and R. J. Webber, *Robust, randomized preconditioning for kernel ridge regression*, arXiv preprint arXiv:2304.12465, 2024, URL https://arxiv.org/abs/2304.12465v4.
24. E. N. Epperly, J. A. Tropp, and R. J. Webber, *Embrace rejection: Kernel matrix approximation by accelerated randomly pivoted Cholesky*, arXiv preprint arXiv:2410.03969, 2024.
25. M. Fanuel, J. Schreurs, and J. Suykens, *Diversity sampling is an implicit regularization for kernel methods*, SIAM J. Math. Data Sci. **3** (2021), no. 1, 280–297, DOI 10.1137/20M1320031.
26. S. Fine and K. Scheinberg, *Efficient SVM training using low-rank kernel representations*, J. Mach. Learn. Res. **2** (2002), 243–264. URL https://dl.acm.org/doi/10.5555/944790.944812.
27. C. Fowlkes, S. Belongie, F. Chung, and J. Malik, *Spectral grouping using the Nyström method*, IEEE Trans. Pattern Anal. Mach. Intell. **26** (2004), no. 2, 214–225, DOI 10.1109/TPAMI.2004.1262185.
28. A. Frieze, R. Kannan, and S. Vempala, *Fast Monte-Carlo algorithms for finding low-rank approximations*, J. ACM **51** (2004), no. 6, 1025–1041, DOI 10.1145/1039488.1039494.

29. G. Gautier, G. Polito, R. Bardenet, and M. Valko, *DPPy: DPP sampling with Python*, J. Mach. Learn. Res. **20** (2019), no. 180, 1–7. URL http://jmlr.org/papers/v20/19-179.html.
30. J. Gillenwater, *Approximate Inference for Determinantal Point Processes*, PhD thesis, University of Pennsylvania, 2014, URL https://jgillenw.com/thesis.pdf.
31. A. Gittens, *The spectral norm error of the naive Nyström extension*, 2011, URL https://arxiv.org/abs/1110.5305v1.
32. A. Glielmo, B. E. Husic, A. Rodriguez, C. Clementi, F. Noé, and A. Laio, *Unsupervised learning methods for molecular simulation data*, Chem. Rev. **121** (2021), no. 16, 9722–9758, DOI 10.1021/acs.chemrev.0c01195.
33. G. H. Golub and C. F. Van Loan, *Matrix Computations*, 4th ed., Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 2013, DOI 10.56021/9781421407944.
34. M. G. Gu and S. C. Eisenstat, *Efficient algorithms for computing a strong rank-revealing QR factorization*, SIAM J. Sci. Comput. **17** (1996), no. 4, 848–869, DOI 10.1137/0917055.
35. V. Guruswami and A. K. Sinop, *Optimal column-based low-rank matrix reconstruction*, Proceedings of the 2012 Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, 2012, pp. 1207–1214, DOI 10.1137/1.9781611973099.95.
36. N. J. Higham, Analysis of the Cholesky decomposition of a semi-definite matrix, M. G. Cox and S. J. Hammarling (eds.), Reliable Numerical Computation, Oxford University Press, 1990, pp. 161–185. URL https://eprints.maths.manchester.ac.uk/id/eprint/1193.
37. D. Holzmüller, V. Zaverkin, J. Kästner, and I. Steinwart, *A framework and benchmark for deep batch active learning for regression*, J. Mach. Learn. Res. **24** (2023), no. 164, 1–81. URL http://jmlr.org/papers/v24/22-0937.html.
38. A. Kulesza and B. Taskar, *Determinantal point processes for machine learning*, Found. Trends Mach. Learn. **5** (2012), no. 2-3, 123–286, DOI 10.1561/2200000044.
39. S. Kumar, M. Mohri, and A. Talwalkar, *Sampling techniques for the Nyström method*, Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics, PMLR, 2009, 304–311. URL https://proceedings.mlr.press/v5/kumar09a.html.
40. S. Kumar, M. Mohri, and A. Talwalkar, *On sampling-based approximate spectral decomposition*, Proceedings of the 26th Annual International Conference on Machine Learning, Association for Computing Machinery, 2009, pp. 553–560, DOI 10.1145/1553374.1553446.
41. S. Kumar, M. Mohri, and A. Talwalkar, *Sampling methods for the Nyström method*, J. Mach. Learn. Res. **13** (2012), no. 34, 981–1006. URL http://jmlr.org/papers/v13/kumar12a.html.
42. J. Lee, S. Schoenholz, J. Pennington, B. Adlam, L. Xiao, R. Novak, and J. Sohl-Dickstein, *Finite versus infinite neural networks: An empirical study*, Proceedings of the 34th International Conference on Neural Information Processing Systems, Curran Associates Inc., 2020, 15156–15172. URL https://dl.acm.org/doi/10.5555/3495724.3496995.
43. P.-G. Martinsson and J. A. Tropp, *Randomized numerical linear algebra: Foundations and algorithms*, Acta Numer. **29** (2020), 403–572, DOI 10.1017/S0962492920000021.
44. G. Meanti, L. Carratino, L. Rosasco, and A. Rudi, *Kernel methods through the roof: Handling billions of points efficiently*, Proceedings of the 34th International Conference on Neural Information Processing Systems, Curran Associates Inc., 2020, 14410–14422. URL https://dl.acm.org/doi/abs/10.5555/3495724.3496932.
45. C. Musco and C. Musco, *Recursive sampling for the Nyström method*, Proceedings of the 31st International Conference on Neural Information Processing Systems, Curran Associates Inc., 2017, 3836–3848. URL https://dl.acm.org/doi/10.5555/3294996.3295140.
46. C. Musco and D. P. Woodruff, *Sublinear time low-rank approximation of positive semidefinite matrices*, IEEE 58th Annual Symposium on Foundations of Computer Science, IEEE, 2017, DOI 10.1109/FOCS.2017.68.
47. J. Poulson, *High-performance sampling of generic determinantal point processes*, Philos. Trans. R. Soc. London, Ser. A **378** (2020), no. 2166, 20190059, DOI 10.1098/rsta.2019.0059.
48. A. Radhakrishnan, G. Stefanakis, M. Belkin, and C. Uhler, *Simple, fast, and flexible framework for matrix completion with infinite width neural networks*, Proc. Natl. Acad. Sci. **119** (2022), no. 16, e2115064119, DOI 10.1073/pnas.2115064119.
49. A. Radhakrishnan, D. Beaglehole, P. Pandit, and M. Belkin, *Mechanism for feature learning in neural networks and backpropagation-free machine learning models*, Science **383** (2024), no. 6690, 1461–1467, DOI 10.1126/science.adi5639.

50. A. Rahimi and B. Recht, *Random features for large-scale kernel machines*, Proceedings of the 20th International Conference on Neural Information Processing Systems, Curran Associates Inc, 2007, 1177–1184. URL https://dl.acm.org/doi/10.5555/2981562.2981710.
51. R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. von Lilienfeld, *Quantum chemistry structures and properties of 134 kilo molecules*, Sci. Data **1** (2014), no. 1, 140022, DOI 10.1038/sdata.2014.22.
52. C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, The MIT Press, 2005, DOI 10.7551/mitpress/3206.001.0001.
53. M. A. Rohrdanz, W. Zheng, M. Maggioni, and C. Clementi, *Determination of reaction coordinates via locally scaled diffusion map*, J. Chem. Phys. **134** (2011), no. 12, 124116, DOI 10.1063/1.3569857.
54. L. Ruddigkeit, R. van Deursen, L. C. Blum, and J.-L. Reymond, *Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17*, J. Chem. Inf. Model. **52** (2012), no. 11, 2864–2875, DOI 10.1021/ci300415d.
55. A. Rudi, L. Carratino, and L. Rosasco, *FALKON: An optimal large scale kernel method*, Proceedings of the 31st International Conference on Neural Information Processing Systems, Curran Associates Inc., 2017, 3891–3901. URL https://dl.acm.org/doi/10.5555/3294996.3295145.
56. A. Rudi, D. Calandriello, L. Carratino, and L. Rosasco, *On fast leverage score sampling and optimal learning*, Proceedings of the 32nd International Conference on Neural Information Processing Systems, Curran Associates Inc., 2018, 5677–5687 URL https://dl.acm.org/doi/10.5555/3327345.3327470.
57. B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, The MIT Press, 2018, DOI 10.7551/mitpress/4175.001.0001.
58. J. Shi and J. Malik, *Normalized cuts and image segmentation*, IEEE Trans. Pattern Anal. Mach. Intell. **22** (2000), no. 8, 888–905, DOI 10.1109/34.868688.
59. A. Smola and P. Bartlett, *Sparse greedy Gaussian process regression*, Proceedings of the 13th International Conference on Neural Information Processing Systems, MIT Press, 2000, 598–604. URL https://dl.acm.org/doi/10.5555/3008751.3008838.
60. S. Steinerberger, *Randomly pivoted partial Cholesky: Random how?*, 2024. URL https://arxiv.org/abs/2404.11487v1.
61. A. Stuke, M. Todorović, M. Rupp, C. Kunkel, K. Ghosh, L. Himanen, and P. Rinke, *Chemical diversity in molecular orbital energy predictions with kernel ridge regression*, J. Chem. Phys. **150** (2019), no. 20, 204121, DOI 10.1063/1.5086105.
62. J. A. Tropp, *An introduction to matrix concentration inequalities*, Found. Trends Mach. Learn. **8** (2015), no. 1-2, 1–230, DOI 10.1561/2200000048.
63. A. Vanraes, *Python implementation of the RLS-Nyström method*, 2019. URL https://github.com/axelv/recursive-nystrom.
64. J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo, *OpenML: Networked science in machine learning*, ACM Special Interest Group on Knowledge Discovery in Data Explorations Newsletter **15** (2013), no. 2, 49–60, DOI 10.1145/2641190.2641198.
65. R. Vershynin, *High-Dimensional Probability: An Introduction with Applications in Data Science*, Cambridge University Press, 2018, DOI 10.1017/9781108231596.
66. U. von Luxburg, *A tutorial on spectral clustering*, Stat. Comput. **17** (2007), no. 4, 395–416, DOI 10.1007/s11222-007-9033-z.
67. S. Wang and Z. Zhang, *Improving CUR matrix decomposition and the Nyström approximation via adaptive sampling*, J. Mach. Learn. Res. **14** (2013), no. 47, 2729–2769. URL http://jmlr.org/papers/v14/wang13c.html.
68. C. Wehmeyer and F. Noé, *Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics*, J. Chem. Phys. **148** (2018), no. 24, 241703, DOI 10.1063/1.5011399.
69. C. Williams and M. Seeger, *Using the Nyström method to speed up kernel machines*, Proceedings of the 13th International Conference on Neural Information Processing Systems, MIT Press, 2000, 661–667. URL https://dl.acm.org/doi/10.5555/3008751.3008847.
70. C. K. I. Williams and M. Seeger, *The effect of the input density distribution on kernel-based classifiers*, Proceedings of the Seventeenth International Conference on Machine Learning, Morgan Kaufmann Publishers Inc., 2000, 1159–1166. URL https://dl.acm.org/doi/10.5555/645529.756511.
71. F. Zhang, *Matrix Theory: Basic Results and Techniques*, 2nd ed., Springer, 2011, DOI 10.1007/978-1-4614-1099-7.

72. F. Zhang and R. Horn, Basic properties of the Schur complement, F. Zhang (ed.), The Schur Complement and Its Applications, Numerical Methods and Algorithms, Springer, 2005, pp. 17–46, DOI 10.1007/0-387-24273-2_2.

APPENDIX A: DETAILS OF NUMERICAL EXPERIMENTS

The numerical experiments in Sections 2.4 and 4 are implemented in Python with code available at https://github.com/eepperly/Randomly-Pivoted-Cholesky. Below we provide further implementation details for RPCHOLESKY, DPP sampling, RLS sampling, and greedy pivoting:

RPCHOLESKY. All the RPCHOLESKY results in Sections 2.4 and 4 use the simple RPCHOLESKY method (Algorithm 1) with block size $T = 1$.

DPP sampling. We use the DPP samplers from the DPPy Python package [29]. However, the samplers from this package all produce error messages when tried on certain inputs, particularly when k is large and $\boldsymbol{A}$ is nearly low-rank. The theoretically fastest `alpha` sampler cannot be used for our numerical comparisons since it has error messages for both the **Smile** and **Spiral** examples. For this reason, we used the `vfx` sampler [18] when assessing the entry access cost of DPP sampling, and we produced Figure 1 using the comparatively slow `GS` sampler which requires a full eigendecomposition of $\boldsymbol{A}$. All these DPP samplers generate exact samples from the k-DPP distribution (C.1); we did not test with inexact samplers based on MCMC [4].

RLS sampling. Several RLS sampling algorithms have been introduced in the literature, including recursive ridge leverage scores (RRLS) [45], SQUEAK [13], and BLESS [56]. The analysis and experiments in this paper are based on the RRLS algorithm [45], which is implemented in the `recursiveNystrom` method [63] for Python. For comparison, we have also implemented RLS sampling by directly forming the matrix $\boldsymbol{A}(\boldsymbol{A} + \lambda \mathbf{I})^{-1}$ and sampling k indices with probability weighted defined by the ridge leverage scores (C.5). Even with the ridge leverage scores computed exactly in this way, the relative trace-norm error for the **Smile** example with $k = 100$ is still roughly 10^{-2}, five orders of magnitude larger than the error due to RPCHOLESKY. This indicates that the poor performance of RLS on this example is not due to errors in the approximation of the ridge leverage scores.

Greedy pivoting. Following the original release of this work as an arXiv preprint, S. Steinerberger [60] pointed out that the greedy pivoting method exhibits degenerate behavior if many diagonal entries of the residual matrix are tied for the largest value, up to floating point errors. In case of a tie, existing implementations of the greedy algorithm select columns according to the default ordering, always choosing the first column with the maximal diagonal value. The impact of the ordering can be removed by randomly pre-shuffling the data $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, which we did for the **Smile** and **Spiral** data sets in figure 1. Pre-shuffling slightly improved the performance of greedy pivoting for the **Spiral** data. We note that existing implementations of greedy pivoting, such as LAPACK's `pstrf`, do not perform this shuffling and process the data in the given order.

APPENDIX B: PSEUDOCODE FOR QR METHODS

Pseucode for column-pivoted partial QR and block randomly pivoted QR are provided as Algorithm B.1 and Algorithm B.2.

APPENDIX C: COMPARISON WITH OTHER METHODS

In this section, we give an example which shows that any Nyström method needs at least r/ε columns to guarantee a (r, ε)-approximation (Section C.1). Then, we analyze how many columns are needed to obtain a (r, ε)-approximation using the greedy method (Section C.2), uniform sampling (Section C.3), DPP sampling (Section C.4), and RLS sampling (Section C.5). The proofs in this

ALGORITHM B.1 Column-pivoted partial QR.

Input: Matrix $\boldsymbol{B} \in \mathbb{C}^{M\times N}$; approximation rank k

Output: Pivot set $\mathsf{S} = \{s_1, \dots, s_k\}$; matrices $\boldsymbol{Q} \in \mathbb{C}^{M\times k}$ and $\boldsymbol{R} \in \mathbb{C}^{k\times N}$ defining approximation $\widehat{\boldsymbol{B}} = \boldsymbol{QR}$

Initialize $\boldsymbol{Q} \leftarrow \mathbf{0}_{M\times k}$ and $\boldsymbol{R} \leftarrow \mathbf{0}_{k\times N}$
for $i = 1$ to k **do**
  Select a pivot column $s_i \in \{1, \dots, N\}$ ▷ See Section 3.3
  $\boldsymbol{g} \leftarrow \boldsymbol{B}(:, s_i)$ ▷ Extract s_i column of input matrix
  $\boldsymbol{g} \leftarrow \boldsymbol{g} - \boldsymbol{Q}(:, 1 : i-1)\boldsymbol{R}(1 : i-1, s_i)$ ▷ Remove projection on previously chosen columns
  $\boldsymbol{g} \leftarrow \boldsymbol{g}/\|\boldsymbol{g}\|$ ▷ Normalize
  $\boldsymbol{Q}(:, i) \leftarrow \boldsymbol{g}$ ▷ Update approximation
  $\boldsymbol{R}(i, :) \leftarrow \boldsymbol{g}^*\boldsymbol{B}$
end for

ALGORITHM B.2 Block randomly pivoted partial QR (aka adaptive sampling [20]).

Input: Matrix $\boldsymbol{B} \in \mathbb{C}^{M\times N}$; block size T; approximation rank k which is a multiple of T

Output: Pivot set S; matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ defining approximation $\widehat{\boldsymbol{B}} = \boldsymbol{QR}$

Initialize $\boldsymbol{Q} \leftarrow \mathbf{0}_{M\times k}$, $\boldsymbol{R} \leftarrow \mathbf{0}_{k\times N}$, and $\mathsf{S} \leftarrow \emptyset$
Initialize $\boldsymbol{p} \leftarrow \mathit{SquaredColumnNorms}(\boldsymbol{B})$
for $i = 0$ to $k/T - 1$ **do**
  Sample $s_{iT+1}, \dots, s_{iT+T} \overset{\text{iid}}{\sim} \boldsymbol{p}/\sum_{j=1}^{N} \boldsymbol{p}(j)$ ▷ Probability prop. to squared column norms of residual
  $\mathsf{S}' \leftarrow \mathit{Unique}(\{s_{iT+1}, \dots, s_{iT+T}\})$
  $\mathsf{S} \leftarrow \mathsf{S} \cup \mathsf{S}'$
  $\boldsymbol{G} \leftarrow \boldsymbol{B}(:, \mathsf{S}')$ ▷ Evaluate columns S' of input matrix
  $\boldsymbol{G} \leftarrow \boldsymbol{G} - \boldsymbol{QR}(:, \mathsf{S}')$ ▷ Remove projections on previously chosen columns
  $\boldsymbol{G} \leftarrow \mathit{Orth}(\boldsymbol{G})$ ▷ Orthonormalize
  $\boldsymbol{Q}(:, (iT+1) : (iT + |\mathsf{S}'|)) \leftarrow \boldsymbol{G}$ ▷ Update approximation
  $\boldsymbol{R}((iT+1) : (iT + |\mathsf{S}'|), :) \leftarrow \boldsymbol{G}^*\boldsymbol{B}$
  $\boldsymbol{p} \leftarrow \boldsymbol{p} - \mathit{SquaredColumnNorms}(\boldsymbol{G}^*\boldsymbol{B})$ ▷ Track squared column norms of residual
  $\boldsymbol{p} \leftarrow \max\{\boldsymbol{p}, \mathbf{0}\}$ ▷ Ensure $\boldsymbol{p}$ remains nonnegative
end for
Remove zero columns from $\boldsymbol{Q}$ and zero rows from $\boldsymbol{R}$

section consolidate the existing literature, and we have attempted to streamline the derivations and obtain sharper constants.

C.1 | Lower bound

Here, we prove a lower bound on the number of columns needed for any Nyström method to achieve a (r, ε)-approximation of a worst-case matrix.

Theorem C.1 (Nyström lower bound [19, 35]). *Fix $r \geq 1$ and $\varepsilon > 0$. There exists a psd matrix $\boldsymbol{A} \in \mathbb{C}^{N\times N}$ such that any rank-k Nyström approximation with*

$$k < r/\varepsilon$$

columns has error $\operatorname{tr}\left(\boldsymbol{A} - \boldsymbol{A}^{(k)}\right) > (1+\varepsilon) \cdot \operatorname{tr}\left(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r\right)$.

Proof. We adapt the proof of [35, Lemma 6.2]. We consider the matrix

$$\boldsymbol{A} = \underbrace{\begin{pmatrix} \boldsymbol{B} & & & \\ & \boldsymbol{B} & & \\ & & \ddots & \\ & & & \boldsymbol{B} \end{pmatrix}}_{Mr \times Mr}, \quad \text{where} \quad \boldsymbol{B} = \underbrace{\begin{pmatrix} 1 & \delta & \cdots & \delta \\ \delta & 1 & & \vdots \\ \vdots & & \ddots & \\ \delta & \cdots & & 1 \end{pmatrix}}_{r \times r}.$$

We consider the rank-k Nyström approximation $\widehat{\boldsymbol{A}}^{(k)}$ that selects k_1 columns from the first block, k_2 columns from the second block, and so forth. The approximation error satisfies

$$\frac{\operatorname{tr}(\boldsymbol{A} - \widehat{\boldsymbol{A}}^{(k)})}{\operatorname{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r)} = \frac{1}{r} \sum_{i=1}^{r} \frac{M - k_i}{M - 1} \left(1 + \frac{1}{\delta^{-1} + k_i - 1}\right) \geq \frac{M - k}{M - 1} \cdot \frac{1}{r} \sum_{i=1}^{r} \left(1 + \frac{1}{\delta^{-1} + k_i - 1}\right)$$

We use the convexity of $f(x) = \frac{1}{x}$ to calculate

$$\frac{1}{r} \sum_{i=1}^{r} \left(1 + \frac{1}{\delta^{-1} + k_i - 1}\right) \geq 1 + \frac{1}{\delta^{-1} + k/r - 1}.$$

Last, we choose M large enough and δ close enough to 1 so that

$$\frac{\operatorname{tr}(\boldsymbol{A} - \widehat{\boldsymbol{A}}^{(k)})}{\operatorname{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r)} > 1 + \varepsilon$$

for each $k < \frac{r}{\varepsilon}$. □

C.2 | The greedy method

The *greedy method* (2.5) is a column Nyström approximation with a long history in numerical analysis [36] under the name *complete pivoting* or *diagonal pivoting*. The papers [9, 26] popularized the method in the context of kernel computations. Despite its popularity, however, the greedy method is known to fail when applied to certain input matrices [36, Ex. 2.1], and the greedy method exhibits poor performance for most of the kernel matrices appearing in Sections 2.4, 4.2, and 4.3. Below in Theorem C.2, we construct a worst-case matrix $\boldsymbol{A}$ that is approximated at a slow $1 - k/N$ rate using the greedy method.

Theorem C.2 (Greedy method). *Fix $r \geq 1$ and $\varepsilon > 0$. Then, the greedy method has the following properties:*

(a) For any psd input matrix $\boldsymbol{A} \in \mathbb{C}^{N \times N}$, the greedy method with

$$k \geq (1 - (1 + \varepsilon)\eta)N$$

columns produces an approximation satisfying $\mathbb{E} \operatorname{tr}\big(\boldsymbol{A} - \boldsymbol{A}^{(k)}\big) \leq (1 + \varepsilon) \cdot \operatorname{tr}\big(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r\big)$.

(b) There exists a psd matrix $\boldsymbol{A} \in \mathbb{C}^{N \times N}$ such that the greedy method with

$$k < (1 - (1 + \varepsilon)\eta)N$$

columns has error $\mathbb{E} \operatorname{tr}\big(\boldsymbol{A} - \boldsymbol{A}^{(k)}\big) > (1 + \varepsilon) \cdot \operatorname{tr}\big(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r\big)$.

As usual, we have defined the relative error $\eta := \operatorname{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r) / \operatorname{tr}(\boldsymbol{A})$.

Proof. To prove part (a), observe at each iteration $1 \le i \le k$ there are at most $N - i + 1$ nonzero entries in the diagonal of the residual matrix $\boldsymbol{A}^{(i-1)}$, and the largest entry is incorporated into the pivot set S. Consequently,

$$\operatorname{tr} \boldsymbol{A}^{(i)} \le \left(1 - \frac{1}{N-i+1}\right) \operatorname{tr} \boldsymbol{A}^{(i-1)} = \frac{N-i}{N-i+1} \operatorname{tr} \boldsymbol{A}^{(i-1)}.$$

By induction, it follows that $\operatorname{tr} \boldsymbol{A}^{(k)} \le \frac{N-k}{N} \operatorname{tr} \boldsymbol{A}$ and

$$\frac{\operatorname{tr} \boldsymbol{A}^{(k)}}{\operatorname{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r)} \le \left(1 - \frac{k}{N}\right) \frac{\operatorname{tr} \boldsymbol{A}}{\operatorname{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r)} = \frac{1 - \frac{k}{N}}{\eta}.$$

If $k \ge (1 - (1+\varepsilon)\eta)N$, the right-hand side is bounded by $1 + \varepsilon$, establishing part (a).

To prove part (b), consider the matrix

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{B} & & & \\ & \boldsymbol{C} & & \\ & & \ddots & \\ & & & \boldsymbol{C} \end{bmatrix}, \quad \text{where} \quad \boldsymbol{B} = \underbrace{\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}}_{(N-rM)\times(N-rM)}, \quad \boldsymbol{C} = \underbrace{\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}}_{M\times M}.$$

In lieu of a good tie-breaking rule, the greedy method chooses entries from the $\boldsymbol{B}$ block before the $\boldsymbol{C}$ blocks. An explicit calculation shows

$$\frac{\operatorname{tr}(\boldsymbol{A} - \widehat{\boldsymbol{A}}^{(k)})}{\operatorname{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r)} = \frac{N-k}{N-M-r+1} = \frac{1 - \frac{k}{N}}{\eta}$$

for each $k \le N - M$. This relative error strictly exceeds $1 + \varepsilon$ as long as $k < (1 - (1+\varepsilon)\eta)N$, which establishes part (b). □

C.3 | Uniform sampling

Another popular column Nyström approximation method is *uniform sampling* [21, 69]. In this method, k columns are selected uniformly at random, either with or without replacement. Theoretically and empirically, the accuracy is higher using uniform sampling without replacement, which avoids the issue of duplicate column selections [39]. Uniform sampling leads to accurate approximations when the dominant r eigenvectors have mass that is spread out equally over all the coordinates (a property known as "incoherence") [31]. However, uniform sampling leads to inaccurate approximations when the dominant eigenvectors have mass that is highly concentrated on a subset of the vertices.

Uniform sampling is typically applied to a kernel matrix $\boldsymbol{A}$ with ones on the diagonal, but *diagonal sampling* [28] is a more general method that randomly selects pivots with probabilities proportional to $\operatorname{diag} \boldsymbol{A}$. Diagonal sampling is guaranteed to produce a (r, ε)-approximation when the number of columns satisfies $k \ge (r-1)/(\varepsilon\eta) + 1/\varepsilon$, as we will prove in Theorem C.3. This bound shows that diagonal sampling accurately approximates the dominant rank-one component of a psd matrix, but it can produce inaccurate approximations of the dominant rank-r component for $r > 1$. A better option for approximating the dominant rank-r component with

$r > 1$ is RPCHOLESKY, which is equivalent to performing diagonal sampling iteratively on the residual matrix.

Theorem C.3 (Diagonal sampling [28]). *Fix $r \geq 1$ and $\varepsilon > 0$. Then, diagonal sampling has the following error properties:*

(a) For any psd input matrix $\boldsymbol{A} \in \mathbb{C}^{N\times N}$, diagonal sampling with

$$k \geq \frac{r-1}{\eta}\varepsilon^{-1} + \varepsilon^{-1}$$

columns produces an approximation satisfying $\mathbb{E}\,\mathrm{tr}\big(\boldsymbol{A} - \boldsymbol{A}^{(k)}\big) \leq (1+\varepsilon)\cdot \mathrm{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r)$.

(b) If $r \geq 2$, there exists a psd matrix $\boldsymbol{A} \in \mathbb{C}^{N\times N}$ such that diagonal sampling with

$$k < \frac{r-1}{\eta}(\sqrt{\varepsilon^{-1}+1} - 1)^2$$

columns has error $\mathbb{E}\,\mathrm{tr}\big(\boldsymbol{A} - \boldsymbol{A}^{(k)}\big) > (1+\varepsilon)\cdot \mathrm{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r)$.

As usual, we have defined the relative error $\eta := \mathrm{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r)/\,\mathrm{tr}(\boldsymbol{A})$.

Proof. Part (a) improves on the earlier error bound [28, Equation (4)], and it is proved using a more detailed argument with the same technique. We assume the sampling is conducted with replacement, which leads to higher error. The trace-norm error takes the form

$$\mathrm{tr}\big(\boldsymbol{A} - \boldsymbol{A}(:,\mathsf{S})\boldsymbol{A}(\mathsf{S},\mathsf{S})^{\dagger}\boldsymbol{A}(\mathsf{S},:)\big) = \mathrm{tr}\big(\boldsymbol{A}^{1/2}\big(\mathbf{I} - \boldsymbol{\Pi}_{\boldsymbol{A}^{1/2}(:,\mathsf{S})}\big)\boldsymbol{A}^{1/2}\big) = \big\|\boldsymbol{A}^{1/2}\big(\mathbf{I} - \boldsymbol{\Pi}_{\boldsymbol{A}^{1/2}(:,\mathsf{S})}\big)\big\|_{\mathrm{F}}^2,$$

where $\boldsymbol{\Pi}_{\boldsymbol{A}^{1/2}(:,\mathsf{S})}$ is the orthogonal projector onto the range of $\boldsymbol{A}^{1/2}(:,\mathsf{S})$. Since $\boldsymbol{A}^{1/2}\boldsymbol{\Pi}_{\boldsymbol{A}^{1/2}(:,\mathsf{S})}$ is the optimal Frobenius norm approximation of $\boldsymbol{A}^{1/2}$ in the row space of $\boldsymbol{A}^{1/2}(\mathsf{S},:)$, we observe the inequality

$$\big\|\boldsymbol{A}^{1/2}\big(\mathbf{I} - \boldsymbol{\Pi}_{\boldsymbol{A}^{1/2}(:,\mathsf{S})}\big)\big\|_{\mathrm{F}}^2 \leq \left\|\boldsymbol{A}^{1/2} - \left(\sum_{j=1}^{r}\boldsymbol{v}_j\boldsymbol{v}_j^*\right)\left(\frac{\mathrm{tr}\,\boldsymbol{A}}{k}\sum_{j=1}^{k}\frac{\mathbf{e}_{s_j}\mathbf{e}_{s_j}^*}{\boldsymbol{A}(s_j,s_j)}\right)\boldsymbol{A}^{1/2}\right\|_{\mathrm{F}}^2,$$

where $\mathbf{e}_{s_i}$ is the unit vector in the direction of the ith random pivot and $\boldsymbol{v}_i$ denotes the ith eigenvector of $\boldsymbol{A}$. Taking transposes and using the orthonormal basis of eigenvectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N$, we calculate

$$\begin{aligned}
&\left\|\boldsymbol{A}^{1/2} - \left(\sum_{j=1}^{r}\boldsymbol{v}_j\boldsymbol{v}_j^*\right)\left(\frac{\mathrm{tr}\,\boldsymbol{A}}{k}\sum_{j=1}^{k}\frac{\mathbf{e}_{s_j}\mathbf{e}_{s_j}^*}{\boldsymbol{A}(s_j,s_j)}\right)\boldsymbol{A}^{1/2}\right\|_{\mathrm{F}}^2 \\
&= \sum_{i=1}^{r}\left\|\boldsymbol{A}^{1/2}\left(\mathbf{I} - \frac{\mathrm{tr}\,\boldsymbol{A}}{k}\sum_{j=1}^{k}\frac{\mathbf{e}_{s_j}\mathbf{e}_{s_j}^*}{\boldsymbol{A}(s_j,s_j)}\right)\boldsymbol{v}_i\right\|^2 + \sum_{i=r+1}^{N}\big\|\boldsymbol{A}^{1/2}\boldsymbol{v}_i\big\|^2
\end{aligned}$$

Using the characterization of $\mathrm{tr}\big(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r\big)$ in terms of the eigenvalues $\lambda_1(\boldsymbol{A}) \geq \cdots \geq \lambda_N(\boldsymbol{A})$, we find

$$\sum_{i=r+1}^{N}\big\|\boldsymbol{A}^{1/2}\boldsymbol{v}_i\big\|^2 = \sum_{i=r+1}^{N}\lambda_i(\boldsymbol{A}) = \mathrm{tr}\big(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r\big)$$

Next, observe that the random vectors

$$\boldsymbol{A}^{1/2}\left(\operatorname{tr}\boldsymbol{A}\frac{\mathbf{e}_{s_j}\mathbf{e}_{s_j}^*}{\boldsymbol{A}(s_j,s_j)}\boldsymbol{v}_i\right)$$

are independent for $i = 1, \dots, k$, and each vector has mean $\boldsymbol{A}^{1/2}\boldsymbol{v}_i$ and expected square norm $\operatorname{tr}\boldsymbol{A}$. This allows us to calculate

$$\mathbb{E}\left\|\boldsymbol{A}^{1/2}\left(\mathbf{I}-\frac{\operatorname{tr}\boldsymbol{A}}{k}\sum_{j=1}^{k}\frac{\mathbf{e}_{s_j}\mathbf{e}_{s_j}^*}{\boldsymbol{A}(s_j,s_j)}\right)\boldsymbol{v}_i\right\|^2=\frac{\operatorname{tr}\boldsymbol{A}-\lambda_i(\boldsymbol{A})}{k}.$$

Summing over $i = 1, \dots, r$ guarantees the error bound

$$\mathbb{E}\operatorname{tr}\big(\boldsymbol{A}-\widehat{\boldsymbol{A}}^{(k)}\big)\le\frac{r-1}{k}\operatorname{tr}\boldsymbol{A}+\left(1+\frac{1}{k}\right)\operatorname{tr}(\boldsymbol{A}-[\![\boldsymbol{A}]\!]_r),$$

which completes part (a) of the theorem.

To prove part (b), we assume the sampling is conducted without replacement and consider the $N \times N$ matrix

$$\boldsymbol{A}=\begin{bmatrix}\boldsymbol{B}&&&\\&\ddots&&\\&&\boldsymbol{B}&\\&&&\boldsymbol{C}\end{bmatrix},\quad\text{where}\quad\boldsymbol{B}=\underbrace{\begin{bmatrix}1&\delta&\cdots&\delta\\\delta&1&\cdots&\delta\\\vdots&\vdots&\ddots&\vdots\\\delta&\delta&\cdots&1\end{bmatrix}}_{M\times M},\quad\boldsymbol{C}=\underbrace{\begin{bmatrix}1&1&\cdots&1\\1&1&\cdots&1\\\vdots&\vdots&\ddots&\vdots\\1&1&\cdots&1\end{bmatrix}}_{N-M(r-1)\times N-M(r-1)}.$$

Next, consider the rank-k Nyström approximation $\widehat{\boldsymbol{A}}^{(k)}$ that selects k_1 columns from the first block, k_2 columns from the second block, and so forth. The Schur complement of $\boldsymbol{B}$ with respect to any k_i distinct columns has trace

$$(M-k_i)\left(1+\frac{1}{\delta^{-1}+k_i-1}\right)(1-\delta).$$

Using the fact $\operatorname{tr}(\boldsymbol{A}-[\![\boldsymbol{A}]\!]_r)=(1-\delta)(r-1)(M-1)$, we calculate

$$\frac{\operatorname{tr}(\boldsymbol{A}-\widehat{\boldsymbol{A}}^{(k)})}{\operatorname{tr}(\boldsymbol{A}-[\![\boldsymbol{A}]\!]_r)}\ge\frac{1}{r-1}\sum_{i=1}^{r-1}\frac{M-k_i}{M-1}\left(1+\frac{1}{\delta^{-1}+k_i-1}\right).$$

We use the convexity of $f(x)=\frac{1}{x}$ and the fact that $\mathbb{E}k_i=\frac{Mk}{N}$ for $1\le i\le r-1$, calculate

$$\frac{\mathbb{E}\operatorname{tr}(\boldsymbol{A}-\widehat{\boldsymbol{A}}^{(k)})}{\operatorname{tr}(\boldsymbol{A}-[\![\boldsymbol{A}]\!]_r)}\ge\frac{M-k}{M-1}\left(1+\frac{1}{\delta^{-1}+\frac{Mk}{N}-1}\right).$$

The worst case occurs when we take $\delta=\sqrt{\frac{\varepsilon}{\varepsilon+1}}$ and let the dimensions M and N grow to infinity, with fixed aspect ratio. Then, we use the identity $\eta N=(r-1)(M-1)(1-\delta)$ to show that the

right-hand side converges

$$\frac{M-k}{M-1}\left(1+\frac{1}{\delta^{-1}+\frac{Mk}{N}-1}\right)\to 1+\frac{1}{\delta^{-1}+\frac{k\eta}{(r-1)(1-\delta)}-1}$$

To make this quantity smaller than $1+\varepsilon$, uniform sampling requires at least

$$k\geq\frac{(r-1)(1-\delta)}{\eta}\left[\varepsilon^{-1}+1-\delta^{-1}\right]=\frac{r-1}{\eta}(\sqrt{\varepsilon^{-1}+1}-1)^2$$

columns. This completes the proof of part (b). □

C.4 | Determinantal point process sampling

Determinantal point process (DPP) sampling [17] is a column Nyström approximation method that selects a pivot set of cardinality $|\mathsf{S}|=k$ according to the distribution

$$\mathbb{P}\{\mathsf{S}=\{s_1,\dots,s_k\}\}=\frac{\det \boldsymbol{A}(\mathsf{S},\mathsf{S})}{\sum_{|\mathsf{S}'|=k}\det \boldsymbol{A}(\mathsf{S}',\mathsf{S}')}. \tag{C.1}$$

DPP sampling has nearly optimal (r,ε)-approximation properties, as we will show in Theorem C.4. However, implementing DPP sampling for large k values remains expensive relative to peer methods [4, 18].

In the DPP sampling literature, there is a surprising connection between RPCHOLESKY and k-DPP sampling: when we apply k steps of RPCHOLESKY to a rank-k orthogonal projection matrix, we obtain *exactly* the same distribution as k-DPP sampling [30, 47]. This observation leads to one of the standard strategies for k-DPP sampling, based on a reduction to rank-k orthogonal projection matrices [38]:

1. Calculate the full eigendecomposition of the target matrix.
2. Randomly select a set of k eigenvectors with probability proportional to the product of the k associated eigenvalues.
3. Form the orthogonal projection matrix using the k eigenvectors.
4. Apply RPCHOLESKY to the projection matrix to obtain the set S.

In step 1, the full eigendecomposition requires $\mathcal{O}(N^3)$ operations. It is much cheaper to apply RPCHOLESKY directly, which is equivalent to performing 1-DPP sampling iteratively on the residual matrix.

Theorem C.4 (k-DPP sampling [11, 35]). *Fix $r\geq 1$ and $\varepsilon>0$. Then, the Nyström approximation produced by k-DPP sampling has the following properties:*

(a) For any psd input matrix $\boldsymbol{A}\in\mathbb{C}^{N\times N}$, k-DPP sampling with

$$k\geq r/\varepsilon+r-1$$

columns produces an approximation satisfying $\mathbb{E}\,\mathrm{tr}\big(\boldsymbol{A}-\boldsymbol{A}^{(k)}\big)\leq(1+\varepsilon)\cdot\mathrm{tr}\big(\boldsymbol{A}-[\![\boldsymbol{A}]\!]_r\big)$.

(b) There exists a psd matrix $\boldsymbol{A}\in\mathbb{C}^{N\times N}$ such that k-DPP sampling with

$$k<r/\varepsilon+r-1$$

columns leads to error $\mathbb{E}\,\mathrm{tr}\big(\boldsymbol{A}-\boldsymbol{A}^{(k)}\big)>(1+\varepsilon)\cdot\mathrm{tr}\big(\boldsymbol{A}-[\![\boldsymbol{A}]\!]_r\big)$.

Proof. These results were essentially proved in [11, Theorem 1] and [35], but for completeness we provide a streamlined derivation here. Let $\boldsymbol{A}/\boldsymbol{A}(\mathsf{S},\mathsf{S})$ denote the Schur complement of $\boldsymbol{A}$ with respect to the coordinates $\mathsf{S} = \{s_1, \dots, s_k\}$. Recall the Crabtree–Haynsworth determinant identity [16, Lem. 1]:

$$(\boldsymbol{A}/\boldsymbol{A}(\mathsf{S},\mathsf{S}))(i,j) = \frac{\det \boldsymbol{A}(\mathsf{S}\cup\{i\},\mathsf{S}\cup\{j\})}{\det \boldsymbol{A}(\mathsf{S},\mathsf{S})} \tag{C.2}$$

for $i, j \notin \mathsf{S}$. Also recall the determinant identity [35, Lem. 2.1]:

$$\sum_{|\mathsf{S}|=k} \det \boldsymbol{A}(\mathsf{S},\mathsf{S}) = e_k(\lambda_1(\boldsymbol{A}), \dots, \lambda_N(\boldsymbol{A})), \tag{C.3}$$

where

$$e_k(\lambda_1(\boldsymbol{A}), \dots, \lambda_N(\boldsymbol{A})) = \sum_{|\mathsf{S}|=k} \prod_{i\in\mathsf{S}} \lambda_i(\boldsymbol{A})$$

is the kth elementary symmetric polynomial evaluated on the eigenvalues of $\boldsymbol{A}$. Using (C.2) and (C.3), we can calculate the error of k-DPP sampling exactly:

$$\begin{aligned}\mathbb{E}\,\mathrm{tr}(\boldsymbol{A} - \widehat{\boldsymbol{A}}^{(k)}) &= \frac{\sum_{|\mathsf{S}|=k} \det \boldsymbol{A}(\mathsf{S},\mathsf{S})\,\mathrm{tr}(\boldsymbol{A}/\boldsymbol{A}(\mathsf{S},\mathsf{S}))}{\sum_{|\mathsf{S}'|=k} \det \boldsymbol{A}(\mathsf{S}',\mathsf{S}')} = \frac{\sum_{|\mathsf{S}|=k}\sum_{i\notin\mathsf{S}} \det \boldsymbol{A}(\mathsf{S}\cup\{i\},\mathsf{S}\cup\{i\})}{\sum_{|\mathsf{S}'|=k} \det \boldsymbol{A}(\mathsf{S}',\mathsf{S}')} \\ &= (k+1)\frac{\sum_{|\mathsf{S}|=k+1} \det \boldsymbol{A}(\mathsf{S},\mathsf{S})}{\sum_{|\mathsf{S}'|=k} \det \boldsymbol{A}(\mathsf{S}',\mathsf{S}')} = (k+1)\frac{e_{k+1}(\lambda_1(\boldsymbol{A}), \dots, \lambda_N(\boldsymbol{A}))}{e_k(\lambda_1(\boldsymbol{A}), \dots, \lambda_N(\boldsymbol{A}))},\end{aligned}$$

Remarkably, this error is the same for a diagonal and non-diagonal matrix, so we might as well assume $\boldsymbol{A}$ is diagonal. Next, as noted by [35], the function

$$f(x_1, \dots, x_N) = \frac{e_{k+1}(x_1, \dots, x_N)}{e_k(x_1, \dots, x_N)}$$

is concave, non-decreasing in all of its arguments, and invariant under permutations of its arguments. Therefore, averaging together some of the arguments cannot decrease the value of f. For every $(x_1, \dots, x_N)$, it follows that

$$f(x_1, \dots, x_N) \le f\Bigg(\underbrace{\sum_{i=1}^{r}\frac{x_i}{r}, \dots, \sum_{i=1}^{r}\frac{x_i}{r}}_{r\text{ times}}, \underbrace{\sum_{i=r+1}^{N}\frac{x_i}{N-r}, \dots, \sum_{i=r+1}^{N}\frac{x_i}{N-r}}_{N-r\text{ times}}\Bigg)$$

Additionally, for every (a, r, b, N),

$$\begin{aligned} f\Big(\underbrace{\frac{a}{r}, \dots, \frac{a}{r}}_{r\text{ times}}, \underbrace{\frac{b}{N-r}, \dots, \frac{b}{N-r}}_{N-r\text{ times}}\Big) &= f\Big(\underbrace{\frac{a}{r}, \dots, \frac{a}{r}}_{r\text{ times}}, \underbrace{\frac{b}{N-r}, \dots, \frac{b}{N-r}}_{N-r\text{ times}}, 0\Big) \\ &\le f\Big(\underbrace{\frac{a}{r}, \dots, \frac{a}{r}}_{r\text{ times}}, \underbrace{\frac{b}{N-r+1}, \dots, \frac{b}{N-r+1}}_{N-r+1\text{ times}}\Big). \end{aligned}$$

Consequently, k-DPP sampling achieves the worst-case error for the diagonal matrix

$$\boldsymbol{A} = \operatorname{diag}\Big(\underbrace{\frac{a}{r}, \dots, \frac{a}{r}}_{r \text{ times}}, \underbrace{\frac{b}{N-r}, \dots, \frac{b}{N-r}}_{N-r \text{ times}}\Big) \tag{C.4}$$

in the limit as $a \to \infty$ and $N \to \infty$. Assuming diagonal $\boldsymbol{A}$ and $k \geq r$, the dominant error arises when k-DPP sampling selects just $r-1$ of the r large diagonal entries. The error is explicitly

$$\lim_{N,a\to\infty} f\Big(\underbrace{\frac{a}{r}, \dots, \frac{a}{r}}_{r \text{ times}}, \underbrace{\frac{b}{N-r}, \dots, \frac{b}{N-r}}_{N-r \text{ times}}\Big) = \lim_{N,a\to\infty} \left[b + \frac{a}{r} \cdot \frac{\binom{r}{r-1}\binom{N-r}{k-r+1}\left(\frac{a}{r}\right)^{r-1}\left(\frac{b}{N-r}\right)^{k-r+1}}{\binom{r}{r}\binom{N-r}{k-r}\left(\frac{a}{r}\right)^{r}\left(\frac{b}{N-r}\right)^{k-r}} \right]$$

$$= \left(1 + \frac{r}{k-r+1}\right) b,$$

whence

$$\mathbb{E}\,\operatorname{tr}(\boldsymbol{A} - \widehat{\boldsymbol{A}}^{(k)}) \leq \left(1 + \frac{r}{k-r+1}\right) b.$$

Because this is an explicit expression for the worst-case error, setting $k \geq \frac{r}{\varepsilon} + r - 1$ always guarantees an (r, ε)-approximation. Conversely, when $k < \frac{r}{\varepsilon} + r - 1$, k-DPP sampling fails to produce an (r, ε)-approximation for a diagonal matrix of the form (C.4) with N and a chosen sufficiently high. □

C.5 | Ridge leverage score sampling

Ridge leverage score (RLS) sampling [2, 13, 45, 56] is a Nyström approximation with a sampling distribution that is potentially more tractable than in DPP sampling. To perform RLS sampling with parameter $\lambda > 0$, we first introduce the vector of ridge leverage scores:

$$\boldsymbol{\ell}^{\lambda} = \operatorname{diag}(\boldsymbol{A}(\boldsymbol{A} + \lambda \mathbf{I})^{-1}). \tag{C.5}$$

Then, we calculate the vector of sampling probabilities [45]:

$$\boldsymbol{p} = \min\{1, f \cdot \boldsymbol{\ell}^{\lambda}\},$$

where $f > 1$ is the oversampling factor. Last, we generate the coordinate set $\mathsf{S} \subseteq \{1, \dots, N\}$ by independently including each index i with probability $\boldsymbol{p}(i)$. The available implementations of RLS sampling [13, 45, 56] are all fairly complicated, as they require selecting parameters λ and f and approximating the resulting RLS sampling distribution. Because of these preprocessing steps, RLS sampling requires a significantly higher number of entry evaluations than RPCHOLESKY for a fixed approximation rank k. In practice, we have found that RLS is also less reliable.

In 2017, Musco & Musco analyzed RLS sampling and proved it produces good low-rank approximations at moderate cost. Here is a slightly simplified version of one of their results [45, Theorem 18]:

Theorem C.5 (Ridge leverage score sampling: probability bound [45]). *For any psd matrix $\boldsymbol{A}$, there exist parameters $\lambda, f > 0$ such that RLS sampling produces a column Nyström approximation $\widehat{\boldsymbol{A}}$ such*

that

$$\operatorname{tr}(\boldsymbol{A} - \widehat{\boldsymbol{A}}) \leq (1+\varepsilon)\operatorname{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r) \quad \textit{with probability at least } 1-\delta. \tag{C.6}$$

The matrix $\widehat{\boldsymbol{A}}$ *has rank*

$$k = \mathcal{O}\Big(\frac{r}{\varepsilon}\log\Big(\frac{r}{\delta\varepsilon}\Big)\Big). \tag{C.7}$$

This result is not directly comparable to our (r,ε)-approximation guarantees for RPCHOLESKY because the error bound (C.6) controls the trace-norm error up to a failure probability rather than in expectation. In addition, the Musco–Musco result does not have explicit constants for the approximation rank k.

To obtain results for RLS sampling that are directly comparable to our own, we reanalyzed RLS sampling, resulting in Theorem C.6 below. Our proof, following [45, Theorem 3], uses the matrix Bernstein inequality to show that $\mathbb{P}\{\|\boldsymbol{A} - \widehat{\boldsymbol{A}}\| > \lambda\}$ decreases exponentially fast as we increase the oversampling f. By appropriately choosing λ and f, we are able to guarantee an (r,ε)-approximation. Our resulting error bounds for RLS sampling in Theorem C.6 depend primarily on $r + r/\varepsilon$, similar to the bounds for DPP sampling. In order to pass from the probability bound (C.6) to an expectation bound (2.8), the approximation rank k acquires a logarithmic dependence on the inverse relative error $1/\eta$.

Theorem C.6 (Ridge leverage score sampling: expectation bound, explicit constants). *Fix* $r \geq 1$ *and* $\varepsilon > 0$. *For any psd input matrix* $\boldsymbol{A} \in \mathbb{C}^{N\times N}$, *the approximation* $\widehat{\boldsymbol{A}}$ *produced by RLS sampling with parameters*

$$\lambda = \frac{\varepsilon}{2r}\operatorname{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r) \quad \textit{and} \quad f = 27\log\Big(\frac{4}{\eta}\Big(r + \frac{r}{\varepsilon}\Big)\Big)$$

has the following error properties:

(a) With probability at least $1 - \varepsilon\eta/2$, *the Nyström approximation* $\widehat{\boldsymbol{A}}$ *satisfies*

$$\frac{\operatorname{tr}\big(\boldsymbol{A} - \widehat{\boldsymbol{A}}\big)}{\operatorname{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r)} \leq 1 + \frac{\varepsilon}{2}. \qquad \operatorname{rank}\widehat{\boldsymbol{A}} \leq 65\Big(r + \frac{r}{\varepsilon}\Big)\log\Big(\frac{4}{\eta}\Big(r + \frac{r}{\varepsilon}\Big)\Big).$$

(b) Define the truncation rank

$$k = 65\Big(r + \frac{r}{\varepsilon}\Big)\log\Big(\frac{4}{\eta}\Big(r + \frac{r}{\varepsilon}\Big)\Big),$$

and set $\widehat{\boldsymbol{A}}^{(k)} = \widehat{\boldsymbol{A}}$ *if* $\operatorname{rank}\widehat{\boldsymbol{A}} \leq k$ *and* $\widehat{\boldsymbol{A}}^{(k)} = \boldsymbol{0}$ *otherwise. Then,* $\widehat{\boldsymbol{A}}^{(k)}$ *is a Nyström approximation with rank at most k, which satisfies* $\mathbb{E}\operatorname{tr}\big(\boldsymbol{A} - \boldsymbol{A}^{(k)}\big) \leq (1+\varepsilon)\cdot\operatorname{tr}\big(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r\big)$.

As usual, we have defined the relative error $\eta := \operatorname{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r)/\operatorname{tr}(\boldsymbol{A})$.

Proof. To prove part (a), we start by bounding the rank of the Nyström approximation. We observe $\operatorname{rank}\widehat{\boldsymbol{A}} \leq |\mathsf{S}|$, where S denotes the set of indices sampled using RLS sampling. From the description of RLS sampling, $|\mathsf{S}|$ is the sum of independent Bernoulli random variables, with expected

value

$$\mathbb{E}|\mathsf{S}| = \sum_{i=1}^{N} \boldsymbol{p}(i) \le f \sum_{i=1}^{N} \boldsymbol{\ell}^{\lambda}(i).$$

The sum of the leverage scores $\sum_{i=1}^{N} \ell_i^{\lambda}$ is bounded by

$$\sum_{i=1}^{N} \boldsymbol{\ell}^{\lambda}(i) = \operatorname{tr}\big(\boldsymbol{A}\big(\boldsymbol{A}+\lambda\mathbf{I}\big)^{-1}\big) = \sum_{i=1}^{N} \frac{\lambda_i(\boldsymbol{A})}{\lambda+\lambda_i(\boldsymbol{A})} \le r + \frac{1}{\lambda}\sum_{i>r}\lambda_i(\boldsymbol{A}) \le 2\Big(r+\frac{r}{\varepsilon}\Big), \tag{C.8}$$

where we have substituted $\lambda = \varepsilon \sum_{i>r} \lambda_i(\boldsymbol{A})/(2r)$. It follows that $\mathbb{E}|\mathsf{S}| \le t$, where

$$t = 54\Big(r+\frac{r}{\varepsilon}\Big)\log\Big(\frac{4}{\eta}\Big(r+\frac{r}{\varepsilon}\Big)\Big).$$

We apply Chernoff's inequality [65, Theorem 2.3.1] with $\delta \approx 0.199$ to yield

$$\mathbb{P}\Big\{\,|\mathsf{S}| > (1+\delta)t\,\Big\} < \mathrm{e}^{-\mathbb{E}|\mathsf{S}|}\left(\frac{\mathrm{e}\,\mathbb{E}|\mathsf{S}|}{(1+\delta)t}\right)^{(1+\delta)t} \le \mathrm{e}^{-t}\left(\frac{\mathrm{e}\,t}{(1+\delta)t}\right)^{(1+\delta)t} = \mathrm{e}^{-t/54} \le \frac{\varepsilon\eta}{4}.$$

With probability at least $1-\varepsilon\eta/4$, we have shown

$$\operatorname{rank}\widehat{\boldsymbol{A}} \le (1+\delta)t \le 65\Big(r+\frac{r}{\varepsilon}\Big)\log\Big(\frac{4}{\eta}\Big(r+\frac{r}{\varepsilon}\Big)\Big).$$

Next, we bound the spectral norm approximation error $\|\boldsymbol{A}-\widehat{\boldsymbol{A}}\|$. To that end, consider the random rank-one matrices

$$\boldsymbol{X}_i = \begin{cases} \big(\frac{1}{\boldsymbol{p}(i)}-1\big)\boldsymbol{B}(:,i)\boldsymbol{B}(i,:) & i \in \mathsf{S}, \\ -\boldsymbol{B}(:,i)\boldsymbol{B}(i,:), & i \notin \mathsf{S}, \end{cases} \quad \text{where} \quad \boldsymbol{B} = \boldsymbol{A}^{1/2}(\boldsymbol{A}+\lambda\mathbf{I})^{-1/2}.$$

Each matrix $\boldsymbol{X}_i$ is mean-zero for $1 \le i \le N$. The matrix $\boldsymbol{X}_i$ is exactly zero if the ith leverage score ℓ_i^{λ} is as large or larger than $1/f$. Otherwise, the matrix $\boldsymbol{X}_i$ is bounded from above by

$$\lambda_{\max}(\boldsymbol{X}_i) \le \frac{1}{\boldsymbol{p}(i)}\|\boldsymbol{B}(:,i)\boldsymbol{B}(i,:)\| = \frac{1}{\boldsymbol{p}(i)}\boldsymbol{\ell}^{\lambda}(i) = \frac{1}{f}.$$

and bounded from below by

$$\lambda_{\min}(\boldsymbol{X}_i) \ge -\|\boldsymbol{B}(:,i)\boldsymbol{B}(i,:)\| = -\boldsymbol{\ell}^{\lambda}(i) \ge -\frac{1}{f}.$$

Hence, $\|\boldsymbol{X}_i\| \le 1/f$. We upper bound the variance of $\sum_{i=1}^{N}\boldsymbol{X}_i$ as

$$\begin{aligned}\sum_{i=1}^{N}\mathbb{E}\boldsymbol{X}_i^2 &= \sum_{\boldsymbol{\ell}^{\lambda}(i)<1/f}\left(\frac{1}{\boldsymbol{p}(i)}-1\right)\boldsymbol{B}(:,i)\boldsymbol{B}(i,:)\boldsymbol{B}(:,i)\boldsymbol{B}(i,:) \\ &\preceq \sum_{\boldsymbol{\ell}^{\lambda}(i)<1/f}\frac{1}{\boldsymbol{p}(i)}\boldsymbol{\ell}^{\lambda}(i)\boldsymbol{B}(:,i)\boldsymbol{B}(i,:) \preceq \frac{1}{f}\boldsymbol{A}(\boldsymbol{A}+\lambda\mathbf{I})^{-1}\end{aligned}$$

By increasing the largest eigenvalue of the right-hand side to $1/f$, we obtain a right-hand side matrix with spectral norm $1/f$ and trace at most $\frac{1}{f}\left(\sum_{i=1}^{N} \boldsymbol{\ell}^{\lambda}(i) + 1\right)$. Therefore, the matrix Bernstein inequality [62, Theorem 7.7.1] gives

$$\mathbb{P}\left\{ \lambda_{\min}\left(\sum_{i=1}^{N} \boldsymbol{X}_i \right) < -\frac{1}{2} \right\} \leq 4\left(\sum_{i=1}^{N} \boldsymbol{\ell}^{\lambda}(i) + 1 \right) \exp\left(-\frac{3}{28} f \right) < \frac{\varepsilon\eta}{4}.$$

Here, we have used the fact that

$$\frac{28}{3} \log\left(\frac{16}{\varepsilon\eta}\left(\sum_{i=1}^{N} \boldsymbol{\ell}^{\lambda}(i) + 1 \right) \right) \leq f = 27 \log\left(\frac{4}{\eta}\left(r + \frac{r}{\varepsilon} \right) \right),$$

which can be shown by a direct calculation using (C.8). With probability at least $1 - \varepsilon\eta/4$, we have shown that $-\frac{1}{2}\mathbf{I} \preceq \sum_{i=1}^{N} \boldsymbol{X}_i$. By multiplying both sides with $(\boldsymbol{A} + \lambda\mathbf{I})^{1/2}$, we obtain

$$-\frac{1}{2}(\boldsymbol{A} + \lambda\mathbf{I}) \preceq (\boldsymbol{A} + \lambda\mathbf{I})^{1/2}\left(\sum_{i=1}^{N} \boldsymbol{X}_i \right)(\boldsymbol{A} + \lambda\mathbf{I})^{1/2}.$$

Using the definition of $\boldsymbol{X}_i$, the right-hand side is exactly $\sum_{i\in \mathsf{S}} \frac{1}{\boldsymbol{p}(i)} \boldsymbol{A}^{1/2}(:, i)\boldsymbol{A}^{1/2}(i, :) - \boldsymbol{A}$. Hence, we can simplify the expression to yield

$$\boldsymbol{A} \preceq 2 \sum_{i\in \mathsf{S}} \frac{1}{\boldsymbol{p}(i)} \boldsymbol{A}^{1/2}(:, i)\boldsymbol{A}^{1/2}(i, :) + \lambda\mathbf{I}.$$

By multiplying both sides with the orthogonal projection $\mathbf{I} - \boldsymbol{\Pi}_{\boldsymbol{A}^{1/2}(:,\mathsf{S})}$, we obtain

$$(\mathbf{I} - \boldsymbol{\Pi}_{\boldsymbol{A}^{1/2}(:,\mathsf{S})})\boldsymbol{A}(\mathbf{I} - \boldsymbol{\Pi}_{\boldsymbol{A}^{1/2}(:,\mathsf{S})}) \preceq \lambda(\mathbf{I} - \boldsymbol{\Pi}_{\boldsymbol{A}^{1/2}(:,\mathsf{S})}).$$

The right-hand side is bounded from above by $\lambda\boldsymbol{I}$, so we have shown $\|(\mathbf{I} - \boldsymbol{\Pi}_{\boldsymbol{A}^{1/2}(:,\mathsf{S})})\boldsymbol{A}(\mathbf{I} - \boldsymbol{\Pi}_{\boldsymbol{A}^{1/2}(:,\mathsf{S})})\| \leq \lambda$. By considering the singular value decomposition for $(\mathbf{I} - \boldsymbol{\Pi}_{\boldsymbol{A}^{1/2}(:,\mathsf{S})})\boldsymbol{A}^{1/2}$, we arrive at the spectral norm error bound

$$\|\boldsymbol{A} - \widehat{\boldsymbol{A}}\| = \|\boldsymbol{A}^{1/2}(\mathbf{I} - \boldsymbol{\Pi}_{\boldsymbol{A}^{1/2}(:,\mathsf{S})})\boldsymbol{A}^{1/2}\| = \|(\mathbf{I} - \boldsymbol{\Pi}_{\boldsymbol{A}^{1/2}(:,\mathsf{S})})\boldsymbol{A}(\mathbf{I} - \boldsymbol{\Pi}_{\boldsymbol{A}^{1/2}(:,\mathsf{S})})\| \leq \lambda.$$

We can convert the spectral-norm error bound into a trace-norm error bound by calculating

$$\operatorname{tr}(\boldsymbol{A} - \widehat{\boldsymbol{A}}) = \sum_{i=1}^{N} \lambda_i(\boldsymbol{A} - \widehat{\boldsymbol{A}}) \leq r\lambda + \sum_{i=r+1}^{N} \lambda_i(\boldsymbol{A}) = \left(1 + \frac{\varepsilon}{2}\right) \cdot \operatorname{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r).$$

Here, we have used the Weyl monotonicity principle [71, Theorem 8.11], which guarantees that $\lambda_i(\boldsymbol{A} - \widehat{\boldsymbol{A}}) \leq \lambda_i(\boldsymbol{A})$ for $1 \leq i \leq N$ since $\widehat{\boldsymbol{A}} \succeq \mathbf{0}$. This completes the proof of part (a).

To prove part (b), we consider the failure event which occurs when the rank or spectral norm approximation error exceeds the bounds in part (b). Even on the failure event, the trace-norm error is bounded by

$$\operatorname{tr}(\boldsymbol{A} - \widehat{\boldsymbol{A}}^{(k)}) \leq \operatorname{tr} \boldsymbol{A}.$$

Since the failure event can only occur with probability $(\varepsilon\eta)/2$, we arrive at the conclusion

$$\begin{aligned}\mathbb{E}\,\mathrm{tr}\big(\boldsymbol{A}-\boldsymbol{A}^{(k)}\big) &= \mathbb{E}\big[\mathrm{tr}\big(\boldsymbol{A}-\boldsymbol{A}^{(k)}\big)\mathbb{1}\{\text{success}\}\big] + \mathbb{E}\big[\mathrm{tr}\big(\mathrm{A}-\mathrm{A}^{(\mathrm{k})}\big)\mathbb{1}\{\mathrm{failure}\}\big] \\ &\leq \left(1+\frac{\varepsilon}{2}\right)\cdot \mathrm{tr}(\boldsymbol{A}-[\![\boldsymbol{A}]\!]_r) + \frac{\varepsilon}{2}\cdot \mathrm{tr}(\boldsymbol{A}-[\![\boldsymbol{A}]\!]_r)\end{aligned}$$

This completes the proof of part (b). □