SUPERCHARGING TRACE ESTIMATION FOR CONCAVE MATRIX FUNCTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose the *accelerated funNyström++* method for computing spectral sums $\mathrm{tr}(f(\mathbf{A}))$ for positive increasing concave functions f, like the log determinant. Our methods requires asymptotically fewer matrix–vector products (matvecs) with $\mathbf{A}$ than all prior methods. In particular, despite Hutch++ and Lanczos-FA each having optimal matvec complexity for their respective tasks, their composition does not provide the optimal complexity matvec for computing spectral sums. Our method is especially efficient when $\mathbf{A}$ has large condition number. We pair our algorithm with a high-quality implementation, provided as part of the open-source RandLAPACK library. We demonstrate the effectiveness of our algorithm and our implementation on a variety of real-world ML tasks, including evaluating Vendi scores and log-determinants. Compared to existing methods, we achieve an order of magnitude improvement in matvec complexity and wall-clock runtime.

1 INTRODUCTION

Many computationally intensive tasks in machine learning (ML) can be formulated as a *spectrum sum* (aka *linear spectral statistic*) estimation problem, where the goal is to efficiently estimate $\mathrm{tr}(f(\boldsymbol{A})) = \sum_i f(\lambda_i(\boldsymbol{A}))$, where f is a given function and $\boldsymbol{A}$ is a large matrix with eigenvalues $\lambda_i(\boldsymbol{A})$. This paradigm arises in many contemporary ML contexts, including:

- **Vendi Score.** The Vendi Score $\exp(-\mathrm{tr}(\boldsymbol{A}\log\boldsymbol{A}))$ measures the diversity of data produced by a generative model. Here, $\boldsymbol{A}$ may be the kernel matrix for the latent embeddings of images (Friedman & Dieng, 2022).
- **Interpolating Information Criterion (IIC).** The IIC is uses a PAC-Bayesian framework to approximate the generalization error of an overparameterized neural network. Computing the IIC depends on the (regularized) log determinant $\mathrm{tr}(\log(\boldsymbol{A}+\sigma\mathbf{I}))$. Here, $\boldsymbol{A} = \boldsymbol{J}\boldsymbol{J}^\top$ where $\boldsymbol{J}$ is the Jacobian of the output of the network with respect to the model parameters (Hodgkinson et al., 2023a;b).
- **A-Optimal Design.** The expected squared reconstruction error in Bayesian inverse problems, used, e.g., for determining sensor placement, is often computed as $\mathrm{tr}(\boldsymbol{A}^{-1})$, where $\boldsymbol{A}$ is an information matrix associated with a candidate set of sensor locations (Alexanderian et al., 2014; Saibaba & Kitanidis, 2015).

In many of these ML applications, the matrix $\boldsymbol{A}$ may be extremely large and only implicitly accessible, e.g., via Jacobian-vector products or by small sub-matrices of the original matrix (Ameli et al., 2025; 2026). In these cases, one cannot apply direct methods for computing spectral sums. Instead, we may resort to *matrix–vector algorithms* that approximate $\mathrm{tr}(f(\boldsymbol{A}))$ using a small number of matrix–vector products (matvecs) with $\boldsymbol{A}$. Understanding the number of matvecs needed to solve problems in linear algebra has major focus of the numerical linear algebra, theoretical computer science, and learning theory communities over the past 10+ years (Braverman et al., 2020; Sun et al., 2021; Tropp & Webber, 2023; Meyer, 2024).

Many prior works on computing spectral sums (e.g., Ubaru et al., 2017; Meyer et al., 2021; Chen et al., 2025; Persson et al., 2022; Epperly et al., 2024) can be understood as using a two-phase approach that composes (1) a *matrix function approximator* to estimate the action $\boldsymbol{x} \mapsto f(\boldsymbol{A})\boldsymbol{x}$ and (2) a *trace estimator* that estimates the trace of a matrix $\boldsymbol{B} = f(\boldsymbol{A})$ using the outputs of (1) as a

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

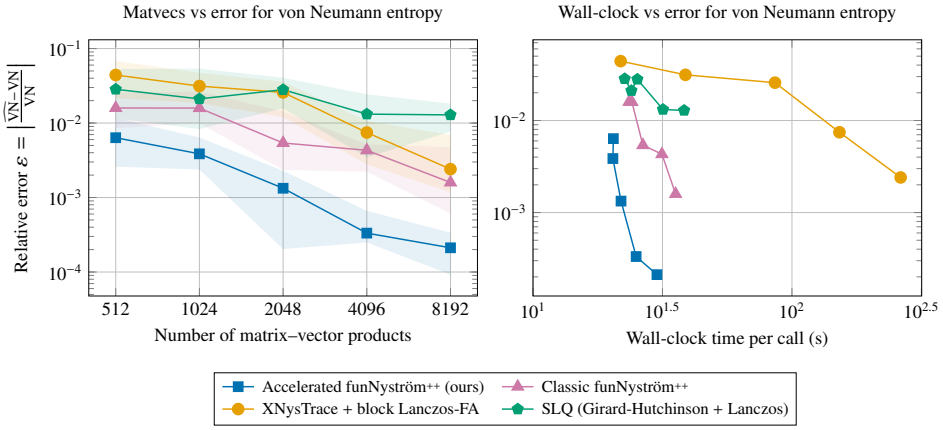


Figure 1: Accelerated funNyström^{++} for estimating spectral sums. Computational resources needed for various methods to estimate the von Neumann entropy $\mathrm{VN}(\boldsymbol{A}) := -\operatorname{tr}(\boldsymbol{A}\log(\boldsymbol{A}))$, the central quantity used to compute Vendi Scores, a metric of diversity for generative models (Friedman & Dieng, 2022). Here, $\boldsymbol{A}$ is a normalized RBF kernel matrix on $n = 50{,}000$ SigLIP ImageNet Embeddings (Zhai et al., 2023) We use a variety of algorithms to estimate $\operatorname{tr}(f(\boldsymbol{A}))$ for $f(x) = -x\log(x)$, which is increasing and concave on $[0, e^{-1}]$, meaning that accelerated funNyström^{++} can be used for any matrix with $\mathbf{0} \preceq \boldsymbol{A} \preceq e^{-1}\mathbf{I}$. Left shows the number of matrix–vector products (matvecvs) computed versus relative error of the computed entropy. Right shows the wall–clock time versus relative error. Solid markers and shaded areas show median error and the $25^{th} - 75^{th}$ quartiles across 20 runs. Compared to competing methods, accelerated funNyström^{++} can achieve **10× lower error** at a fixed matvec budget, and run **almost 2× faster**. See Section 5.1 for details.

black box. We know algorithms with optimal matvec complexities for both tasks (1) and (2) in isolation. For matrix-function approximation (1), the Lanczos-FA algorithm is known to achieve the optimal matvec complexity (Dereziński et al., 2026), with the precise number of matvecs $t_{(\varepsilon,f,\boldsymbol{A})}$ depending on the error tolerance ε, the function f, and the spectrum $[\lambda_{\min}(\boldsymbol{A}), \lambda_{\max}(\boldsymbol{A})]$ of the matrix $\boldsymbol{A}$. For trace estimation (2), the Hutch^{++} algorithm estimates the trace of a positive-semidefinite (psd) matrix $\boldsymbol{A}$ to relative error $(1 \pm \varepsilon)$ using at most $O(1/\varepsilon)$ matvecs, which is also known to be optimal (Meyer et al., 2021; Meyer, 2024). Given that we know optimal algorithms for the matrix-function approximation and trace estimation tasks separately, it is natural to surmise that the composition of these algorithms, using $\mathcal{O}(t_{(\varepsilon,f,\boldsymbol{A})}/\varepsilon)$ matvecs, should achieve the optimal matvec complexity for the spectral sum problem. In this article, we refute this intuition.

1.1 Contributions

Our main result is that the spectral sum $\operatorname{tr}(f(\boldsymbol{A}))$ associated with a psd matrix $\boldsymbol{A}$ and a positive, concave, increasing function f can be estimated at a ***fundamentally lower computational complexity*** than the composition of an optimal matrix-function approximator (like Lanczos-FA) with an optimal trace estimator (like Hutch^{++}). To accomplish this task, we propose the *accelerated funNyström^{++} algorithm* (Algorithm 1), a significant retooling of the original funNyström^{++} algorithm of Persson & Kressner (2023).

Our algorithm starts by using k matvecs with $\boldsymbol{A}$ to form a rank-k Nyström approximation $\widehat{\boldsymbol{A}} \approx \boldsymbol{A}$ (Williams & Seeger, 2000; Gittens & Mahoney, 2013). Our first central observation is that

$$\textit{If } \widehat{\boldsymbol{A}} \approx \boldsymbol{A} \textit{ then } f(\widehat{\boldsymbol{A}}) \approx f(\boldsymbol{A}) \textit{ for any positive increasing concave } f.$$

Thus, for a wide range of functions f, we can form a low-rank approximation to $f(\boldsymbol{A})$ *without* using a matrix-function approximator like Lanczos. With this *funNyström approximation* $f(\widehat{\boldsymbol{A}})$ in hand, we can implement a variance-reduced trace estimator (Meyer et al., 2021; Persson et al., 2022). Expand

$$\operatorname{tr}(f(\boldsymbol{A})) = \operatorname{tr}(f(\widehat{\boldsymbol{A}})) + \operatorname{tr}(f(\boldsymbol{A}) - f(\widehat{\boldsymbol{A}})).$$

We compute the first term directly from $\widehat{\boldsymbol{A}}$, and we use ℓ samples of the classic Girard-Hutchinson estimator to approximate the second term (Girard, 1989; Hutchinson, 1989). This requires us to com-

pute ℓ matvecs with $f(\boldsymbol{A})$, which we approximate using a Lanczos method as our matrix-function approximator. This necessitates $\mathcal{O}(\ell\, t_{(\varepsilon,f,\boldsymbol{A})})$ matvecs with $\boldsymbol{A}$.

Prior work that used the funNyström approximation for trace estimation took $k = \ell = \mathcal{O}(1/\varepsilon)$ (Persson & Kressner, 2023). Our second central observation is that this is suboptimal:

$$\textit{The procedure uses } \mathcal{O}\Big(k + \ell\, t_{(\varepsilon,f,\boldsymbol{A})}\Big) \textit{ matvecs.} \tag{1}$$

So, increasing the rank k is much cheaper than increasing the number of samples ℓ.

The asymmetry arises because $f(\widehat{\boldsymbol{A}})$ is built without a matrix-function approximator. The analysis of variance-reduced trace estimators proves that the output is a $(1 \pm \varepsilon)$ approximation to $\operatorname{tr}(f(\boldsymbol{A}))$ when

$$k\ell \gtrsim \frac{1}{\varepsilon^2} \qquad \text{and} \qquad \ell \geq 1. \tag{2}$$

Our accelerated funNyström^{++} method minimizes the matvec complexity in Equation (1) subject to the constraints in Equation (2), using a total of

$$\mathcal{O}\left(\max\left\{t_{(\varepsilon,f,\boldsymbol{A})},\ \frac{\sqrt{t_{(\varepsilon,f,\boldsymbol{A})}}}{\varepsilon}\right\}\right) \text{ matvecs.} \tag{3}$$

1.1.1 Summary of results

We split our contributions into theoretical improvements and practical improvements. We begin with the theoretical improvements:

1. **More low-rank approximation, lower complexity.** Prior work ran funNyström^{++} with $k = \ell = \mathcal{O}(1/\varepsilon)$, using a total of $\mathcal{O}(t_{(\varepsilon,f,\boldsymbol{A})}/\varepsilon)$ matvecs. In this work, by taking $k \gg \ell$, we reduce the matvec complexity for estimating $\operatorname{tr}(f(\boldsymbol{A}))$ to the improved rate in Equation (3).
2. **Increasing concave functions.** Underlying our analysis is a rigorous claim that $f(\widehat{\boldsymbol{A}}) \approx f(\boldsymbol{A})$ when $\widehat{\boldsymbol{A}} \approx \boldsymbol{A}$ if f is increasing and concave; see Section B.2. In prior work, this claim was only supported for *operator monotone functions* f, a restrictive subclass of concave functions. Our guarantee enables us to estimate a far wider variety of quantities, like the von Neumann entropy, at the accelerated rate.

In addition to the theoretical benefits above, we also have provided practical improvements:

3. **Faster low-rank approximation.** The existing work (Persson & Kressner, 2023) implements funNyström^{++} using a comparatively slow two-pass Nyström approximation relying on a dense Gaussian matrix. We instead use a single-pass Nyström approximation with a sparse random matrix $\boldsymbol{\Omega}$, which is exceptionally fast on real hardware. The theoretical analysis of this variant, developed in Section B.1, requires a significant strengthening of the results from Persson & Kressner (2023).
4. **User-friendly, optimized implementation.** To make accelerated funNyström^{++} useful and accessible for practitioners, we develop a high-performance implementation, which we contributed to the publicly-available RandLAPACK software library. The implementation uses a highly efficient block Lanczos algorithm, leading to substantial performance improvements over single-vector implementations from prior work. We also develop a parameter-free implementation that automatically determines the rank k, the Lanczos depth t, and the number of probes ℓ. This optimized software should enable accelerated funNyström^{++} to be used a drop-in solution for spectral sums in any ML workflow.

Section D.1 describes our implementation and links to its anonymized source code.

In Section 5, we demonstrate the utility of our method for a three different ML tasks: computing the generalization error of ML models via the log-determinant, computing the diversity of outputs of a generative model via the Vendi score, and computing . For brevity, in the next section we highlight results for just one of these tasks: computing the log-determinant.

Algorithm 1. Accelerated funNyström^{++} *(conceptual version; see Algorithm 7)*

Input: Matvec access to $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, increasing positive concave $f : \mathbb{R} \to \mathbb{R}$, parameters k, ℓ, t.
Output: Approximation $\widehat{\mathrm{tr}}_{\mathrm{fN}++}$ to $\mathrm{tr}(f(\boldsymbol{A}))$

1 Construct rank-k Nyström approximation $\hat{\boldsymbol{A}} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top$ ▷ k matvecs, Algorithm 5
2 Compute HEAD $= \mathrm{tr}(f(\Lambda))$
3 Sample $\boldsymbol{S} \in \mathbb{R}^{n\times\ell}$ with iid $\{\pm 1\}$ entries
4 Compute MID $= \frac{1}{\ell}\widehat{\mathrm{tr}}_{\mathrm{BL}}(\boldsymbol{A}, \boldsymbol{S}, f, t)$ ▷ ℓt matvecs, Block Lanczos (Algorithm 3)
5 Compute CORRECTION $= \frac{1}{\ell}\,\mathrm{tr}(\boldsymbol{S}^\top \boldsymbol{U} f(\boldsymbol{\Lambda})\boldsymbol{U}^\top \boldsymbol{S})$
6 **return** $\widehat{\mathrm{tr}}_{\mathrm{fN}++}$ = HEAD + MID − CORRECTION.

1.2 Related Work

There is a large literature on related topics; we will discuss only some of these.

Trace estimation. The most classic method is the Girard-Hutchinson estimator (Girard, 1989; Hutchinson, 1989), which requires many samples to be accurate. To pursue higher precision, we can use *variance-reduced* trace estimators like Hutch^{++}, Nyström^{++}, and XTrace (Meyer et al., 2021; Persson et al., 2022; Epperly et al., 2024). These methods are not aware of the function f. So, to estimate a spectral sum, we must compose the trace estimator with a matrix-function approximator, requiring $\mathcal{O}(t_{(\varepsilon,f,\boldsymbol{A})}/\varepsilon)$ matvecs.

Matrix-function application. The most popular methods for approximately computing the action $\boldsymbol{x} \mapsto f(\boldsymbol{A})\boldsymbol{x}$ from matvecs with $\boldsymbol{A}$ are essentially all Krylov methods. These methods generally all have the same worst-case asymptotic matvec complexity Chen (2024). These methods are unaware of the trace estimation problem, so we must compose them with a trace estimator to estimate a spectral sum, requiring $\mathcal{O}(t_{(\varepsilon,f,\boldsymbol{A})}/\varepsilon)$ matvecs.

Spectral sum estimation. As we have discussed, a conceptually simple way to estimate a spectral sum is by composing a Krylov methods with a trace estimator. However, several algorithms do not have this shape. The *Krylov-aware trace estimator* of Chen & Hallman (2023) merges some computation between Hutch^{++} and Lanczos-FA, which can be very fast in practice, despite apparent having the same worst-case matvec complexity as the naive Hutch^{++} plus Lanczos-FA approach. There are also many methods fine-tuned for specific matrix functions, especially the log-determinant (Boutsidis et al., 2017; Han et al., 2015; Madhavan et al., 2026; Cortinovis & Toni, 2026). As far as we are aware, none of these methods outperform the basic $\mathcal{O}(t_{(\varepsilon,f,\boldsymbol{A})}/\varepsilon)$ worst-case matvec complexity even in these specialized settings.

2 Preliminaries

Scalars are written in italic, vectors are in bold lowercase, and matrices are in bold uppercase. For a diagonalizable matrix $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{-1}$ with eigenvalues λ_i and function $f : \mathbb{R} \to \mathbb{R}$, the matrix function is $f(\boldsymbol{A}) = \boldsymbol{U}f(\boldsymbol{\Lambda})\boldsymbol{U}^{-1}$, where $f(\boldsymbol{\Lambda})$ is the diagonal matrix with entries $f(\lambda_i)$. A function f is operator monotone if we have $f(\boldsymbol{A}) \preceq f(\boldsymbol{B}) \preceq \boldsymbol{0}$ for all matrices $\boldsymbol{A} \preceq \boldsymbol{B} \preceq \boldsymbol{0}$. The trace of a matrix $\boldsymbol{A}$ is denoted by $\mathrm{tr}(\boldsymbol{A})$, the Frobenius norm by $\|\boldsymbol{A}\|_{\mathrm{F}}$, the spectral norm by $\|\boldsymbol{A}\|_2$, and the Schatten-4 norm by $\|\boldsymbol{A}\|_{(4)}$.

3 Accelerated funNyström++

To phrase our core guarantee for the accelerated funNyström^{++} algorithm, we need to first understand the complexity parameter t for matrix-function approximators like Lanczos-FA. The Lanczos-FA algorithm is a Krylov method that approximates $f(\boldsymbol{A})\boldsymbol{b}$ using repeated matvecs with $\boldsymbol{A}$, and the output after running t steps of the algorithm always takes the form $p(\boldsymbol{A})\boldsymbol{b}$ for a degree-$(t-1)$ polynomial p. In fact, the complexity of Lanczos-FA is governed by the the degree of the *best* polynomial approximation to f. We make the following definition.

Definition 3.1 (Polynomial approximation degree)**.** Fix a function f, symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, and tolerance $\varepsilon > 0$. The degree of polynomial approximation $t_{(\varepsilon,f,\boldsymbol{A})}$ is the minimum degree t of

any polynomial p that approximates f to ε/n relative error on the interval $\mathcal{S} = [\lambda_{\min}(\boldsymbol{A}), \lambda_{\max}(\boldsymbol{A})]$:

$$t_{(\varepsilon,f,\boldsymbol{A})} := \min\left\{\deg(p) \;:\; \max_{x\in\mathcal{S}} |f(x) - p(x)| \le (\varepsilon/n) \max_{x\in\mathcal{S}} |f(x)|\right\}.$$

It is well established that $\mathcal{O}(t_{(\varepsilon,f,\boldsymbol{A})})$ matvecs suffice for Lanczos-FA and related Krylov methods to compute the action $\boldsymbol{x} \mapsto f(\boldsymbol{A})\boldsymbol{x}$ to error ε (Chen, 2024). Further, in the worst case over all matrices $\boldsymbol{A}$ with eigenvalues any fixed interval $\mathcal{S}$, there is a matching lower bound showing that $\Omega(t_{(\varepsilon,f,\boldsymbol{A})})$ matvecs are necessary to approximate $f(\boldsymbol{A})\boldsymbol{x}$ (Dereziński et al., 2026).

We may now state our main result.

Theorem 3.2 (Accelerated funNyström^{++})**.** *Fix any psd matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and any $\varepsilon > 0$. Let f be non-negative, concave, and increasing on $[0, \lambda_{\max}(\boldsymbol{A})]$. Then* accelerated funNyström^{++} *(Algorithm 1) returns an estimate $\widehat{\operatorname{tr}} \in (1 \pm \varepsilon)\operatorname{tr}(f(\boldsymbol{A}))$ with probability at least $^{99}/_{100}$ using*

$$\mathcal{O}\left(\max\left\{t_{(\varepsilon,f,\boldsymbol{A})}, \frac{\sqrt{t_{(\varepsilon,f,\boldsymbol{A})}}}{\varepsilon}\right\}\right) \text{ matvecs with } \boldsymbol{A}.$$

To derive the accelerated funNyström^{++} algorithm and to explain where the accelerated matvec complexity budget arises from, we need to introduce the a variety of algorithmic building blocks.

3.1 Ingredient 1: Nyström

There are multiple way to form a low-rank approximation $\widehat{\boldsymbol{A}} \approx \boldsymbol{A}$. We will use the Nyström approximation (Gittens & Mahoney, 2013; Tropp et al., 2017; Camaño et al., 2025), which is useful for psd matrices $\boldsymbol{A}$. To define this approximation, let $\boldsymbol{\Omega} \in \mathbb{R}^{n\times k}$ be a *random test matrix*, often being either a random sparse matrix or having iid random sign bit entries. Then, the Nyström approximation to $\boldsymbol{A}$ with respect to $\boldsymbol{\Omega}$ is

$$\widehat{\boldsymbol{A}} = \boldsymbol{A}\langle\boldsymbol{\Omega}\rangle := \boldsymbol{A}^\top\boldsymbol{\Omega}(\boldsymbol{\Omega}^\top\boldsymbol{A}\boldsymbol{\Omega})^{-1}\boldsymbol{\Omega}^\top\boldsymbol{A}.$$

This perhaps nonintuitive definition has several excellent properties. First, we have the psd ordering $\boldsymbol{0} \preceq \widehat{\boldsymbol{A}} \preceq \boldsymbol{A}$. Second, $\widehat{\boldsymbol{A}}$ can be computed using k matvecs with $\boldsymbol{A}$. Third, there exists an algorithm that is numerically reliable and returns the eigendecomposition of $\widehat{\boldsymbol{A}}$ directly Algorithm 5[1]. Fourth, if we let $\boldsymbol{\Omega}$ be a sparse random matrix, we can bound the residual $\|\boldsymbol{A} - \widehat{\boldsymbol{A}}\|_{\mathrm{F}}$:

Theorem 3.3 (Sparse sketching Nyström (informal), Theorem B.3)**.** *Fix psd matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and target rank $r \le n$. Let $\boldsymbol{\Omega} \in \mathbb{R}^{n\times k}$ be a sparse sketching matrix with embedding dimension $k \asymp r$. Then the single pass Nyström approximation $\widehat{\boldsymbol{A}} = \boldsymbol{A}\langle\boldsymbol{\Omega}\rangle$, with probability at least $^{99}/_{100}$ has*

$$\|\boldsymbol{A} - \widehat{\boldsymbol{A}}\|_{\mathrm{F}} \lesssim \|\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r\|_{\mathrm{F}} + \frac{1}{\sqrt{r}}\operatorname{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r), \tag{4}$$

where $[\![\boldsymbol{A}]\!]_r$ denotes the best rank-r approximation to $\boldsymbol{A}$.

Bounds with the form of Equation (5) are known when $\boldsymbol{\Omega}$ has iid Gaussian entries (Tropp & Webber, 2023, Cor. 8.8), but have not been shown when $\boldsymbol{\Omega}$ is a sparse test matrix. The proof of Theorem 3.3 uses several recent tools from random matrix theory and numerical linear algebra; see Section B.1 for details. This guarantee is important: as we empirically demonstrate in , the choice to make $\boldsymbol{\Omega}$ sparse makes accelerated funNyström^{++} much faster in practice.

3.2 Ingredient 2: Nyström++

The most basic trace estimator is a Monte Carlo method called the Girard-Hutchinson trace estimator (Girard, 1989; Hutchinson, 1989). Letting $\boldsymbol{S} \in \mathbb{R}^{n\times\ell}$ have iid ± 1 entries, the estimator is

[1] We use a Gram-matrix based algorithm to compute the eigendecomposition of $\widehat{\boldsymbol{A}}$, which is less numerically stable than the more standard implementation (e.g. as found in Tropp et al. (2017)), but is significantly faster when the rank k is large. If the user needs over 8 digits of accuracy from their spectral sum estimate, it may be necessary to use the more stable method. See for discussion.

$H_\ell(\boldsymbol{A}) := \frac{1}{\ell}\operatorname{tr}(\boldsymbol{S}^\top \boldsymbol{A}\boldsymbol{S})$. The method is unbiased, uses ℓ matvecs with $\boldsymbol{A}$ and produces a $(1 \pm \varepsilon)$ relative error estimate to $\operatorname{tr}(\boldsymbol{A})$ for psd $\boldsymbol{A}$ with high probability when $\ell \gtrsim 1/\varepsilon^2$. Fortunately, this rate can be accelerated.

The most efficient modern trace estimators rely on *variance reduction*. Here, we follow the blueprint of the Nyström^{++} method of Persson et al. (2022). The method starts by expending k matvecs to form a Nyström approximation $\widehat{\boldsymbol{A}} \approx \boldsymbol{A}$. Then, we return the Nyström^{++} estimate $\widehat{\operatorname{tr}}_{\mathrm{N}++} = \operatorname{tr}(\widehat{\boldsymbol{A}}) + H_\ell(\boldsymbol{A} - \widehat{\boldsymbol{A}})$. Accuracy guarantees for this method were proven by Persson et al. (2022):

Imported Theorem 3.4 (Nyström^{++}, Persson et al. (2022))**.** *Let $\boldsymbol{A}$ be a psd matrix, and fix parameters k, ℓ and ε. Suppose that $k\ell \gtrsim 1/\varepsilon^2$. Let $\widehat{\boldsymbol{A}}$ be an approximation of $\boldsymbol{A}$ that satisfies Equation (5) for some $r \asymp k$. Then $\widehat{\operatorname{tr}}_{\mathrm{N}++} \in (1 \pm \varepsilon)\operatorname{tr}(\boldsymbol{A})$ with probability at least $99/100$.*

If our goal was only to estimate $\operatorname{tr}(\boldsymbol{A})$, then the story would end here. Subject to the constraint $k\ell \gtrsim 1/\varepsilon^2$, the total matvec cost $k + \ell$ is minimized when $k = \ell \asymp 1/\varepsilon$. It is known that this matvec complexity cannot be improved in the worst case (Meyer et al., 2021; Meyer, 2024). As we will show, this accounting of the total matvec cost will change when we look at estimating $\operatorname{tr}(f(\boldsymbol{A}))$.

3.3 Ingredient 3: funNyström and funNyström++

In order to use a variance-reduced trace estimator to approximate $\operatorname{tr}(f(\boldsymbol{A}))$, we need a low-rank approximation to $f(\boldsymbol{A})$. In general, we can use a matrix-function approximator like Lanczos-FA to approximate $f(\boldsymbol{A})\boldsymbol{\Omega}$ and thereby construct a rank-k Nyström approximation to $f(\boldsymbol{A})$. However, this would use $\mathcal{O}(kt_{(\varepsilon,f,\boldsymbol{A})})$ matvecs, which can be very expensive. For positive increasing concave functions f, we show that a simpler strategy suffices: approximate $f(\boldsymbol{A})$ by $f(\widehat{\boldsymbol{A}})$ where $\widehat{\boldsymbol{A}}$ is a Nyström approximation to $\boldsymbol{A}$. The matrix $f(\widehat{\boldsymbol{A}})$ is called the *funNyström approximation* (Persson & Kressner, 2023).

Theorem 3.5 (Mixed-norm funNyström approximation, Special case of)**.** *Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ be psd, and let $\widehat{\boldsymbol{A}}$ be a Nyström approximation to $\boldsymbol{A}$ that satisfies Equation (5). Then, for any function f that is positive increasing and concave on $[0, \lambda_{\max}(\boldsymbol{A})]$, we have*

$$\|f(\boldsymbol{A}) - f(\widehat{\boldsymbol{A}})\|_{\mathrm{F}} \lesssim \|f(\boldsymbol{A}) - [\![f(\boldsymbol{A})]\!]_r\|_{\mathrm{F}} + \frac{1}{\sqrt{r}}\operatorname{tr}(f(\boldsymbol{A}) - [\![f(\boldsymbol{A})]\!]_r), \tag{5}$$

So, by Imported Theorem 3.4, the funNyström approximation is accurate enough that we can use it in a variance-reduced approxmiation of $\operatorname{tr}(f(\boldsymbol{A}))$. The resulting method is known as funNyström^{++}:

$$\widehat{\operatorname{tr}}_{\mathrm{fN}++} := \operatorname{tr}(f(\widehat{\boldsymbol{A}})) + H_\ell(f(\boldsymbol{A}) - f(\widehat{\boldsymbol{A}})).$$

The guarantees in Imported Theorem 3.4 and Theorem 3.5 show that, if $k\ell \gtrsim 1/\varepsilon^2$, the funNyström^{++} estimator returns a $(1 \pm \varepsilon)$ relative error approximation to $\operatorname{tr}(f(\boldsymbol{A}))$ with high probability.

3.4 Ingredient 4: Block Lanczos-QF

The only remaining ingredient is the matrix-function approximator. The funNyström^{++} algorithm requires us to compute the term

$$H_\ell(f(\boldsymbol{A}) - f(\widehat{\boldsymbol{A}})) = \frac{1}{\ell}\operatorname{tr}(\boldsymbol{S}^\top f(\boldsymbol{A})\boldsymbol{S}) - \frac{1}{\ell}\operatorname{tr}(\boldsymbol{S}^\top f(\widehat{\boldsymbol{A}})\boldsymbol{S}).$$

The second term is easy to compute, since we already have the eigendecomposition $\widehat{\boldsymbol{A}} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top$ available to us. We use the block Lanczos-QF method to approximate the first term:

Lemma 3.6 (Block Lanczos-QF, Corollary C.2)**.** *Fix a function f, a symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, an error tolerance $\varepsilon > 0$, and a matrix $\boldsymbol{S} \in \mathbb{R}^{n\times\ell}$ with $\|\boldsymbol{S}\|_{\mathrm{F}} \le n$. The the block Lanczos-QF algorithm (Algorithm 3) uses $\mathcal{O}(\ell t_{(\varepsilon,f,\boldsymbol{A})})$ matvecs with $\boldsymbol{A}$ to return an estimate $\widehat{\operatorname{tr}}_{\mathrm{BL}} \in (1 \pm \varepsilon)\operatorname{tr}(\boldsymbol{S}^\top f(\boldsymbol{A})\boldsymbol{S})$.*

The accelerated funNyström^{++} method deterministically has $\|\boldsymbol{S}\|_{\mathrm{F}} = \sqrt{n\ell}$, so the condition on $\|\boldsymbol{S}\|_{\mathrm{F}}$ always holds. The block Lanczos-QF algorithm is typically stated in a more general fashion, as a way to estimate the entire matrix $\boldsymbol{S}^\top f(\boldsymbol{A})\boldsymbol{S}$ (Chen, 2024). Our characterization here is chosen for

ease of presentation; for a more general discussion see Section C. The more standard Lanczos-FA method would give the worst-case matvec complexity. However, block Lanczos converges with a much smaller number of matvecs on many real-world matrices. In , we show an ablation over different Lanczos methods, demonstrating this impact in practice.

3.5 Result: Accelerated funNyström++

We have established that we can run the funNyström++ method using a single-pass sparse test matrix $\boldsymbol{\Omega}$, for any increasing concave function f, and utilizing the block Lanczos-QF method to implement the Girard-Hutchinson evaluations on $f(\boldsymbol{A})$. The resulting estimator can be written as

$$\widehat{\mathrm{tr}}_{\mathrm{fN++}} = \mathrm{tr}(f(\boldsymbol{\Lambda})) + \frac{1}{\ell}\widehat{\mathrm{tr}}_{\mathrm{BL}}(\boldsymbol{A}, \boldsymbol{S}, f, t_{(\varepsilon,f,\boldsymbol{A})}) - \frac{1}{\ell}\,\mathrm{tr}(\boldsymbol{S}^\top \boldsymbol{U} f(\boldsymbol{\Lambda})\boldsymbol{U}^\top \boldsymbol{S}),$$

where $\widehat{\boldsymbol{A}} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top$ is the eigendecomposition of our Nyström approximation. The exact procedure is reflected in Algorithm 1. To optimize the matvec complexity of this method, we minimize the overall matvec complexity $k + \ell t_{(\varepsilon,f,\boldsymbol{A})}$ subject to the Nyström++ constraint $k\ell \gtrsim 1/\varepsilon^2$ and the requirement that we make at least $\ell \geq 1$ Girard-Hutchinson samples.

The resulting optimized matvec complexity is a casewise result. If $t_{(\varepsilon,f,\boldsymbol{A})} \gtrsim 1/\varepsilon^2$, then we take

$$\ell = 1 \quad \text{and} \quad k \asymp \frac{1}{\varepsilon^2}, \quad \text{for a total of } k + \ell t_{(\varepsilon,f,\boldsymbol{A})} = O(t_{(\varepsilon,f,\boldsymbol{A})}) \text{ matvecs.}$$

Alternatively, if $t_{(\varepsilon,f,\boldsymbol{A})} \lesssim 1/\varepsilon^2$, then we take

$$\ell \asymp \frac{1}{\varepsilon\sqrt{t_{(\varepsilon,f,\boldsymbol{A})}}} \quad \text{and} \quad k \asymp \frac{\sqrt{t_{(\varepsilon,f,\boldsymbol{A})}}}{\varepsilon}, \quad \text{for a total of } k + \ell t_{(\varepsilon,f,\boldsymbol{A})} = O\left(\frac{\sqrt{t_{(\varepsilon,f,\boldsymbol{A})}}}{\varepsilon}\right) \text{ matvecs.}$$

the maximum of these these two cases yield the result in Theorem 3.2.

4 Case study: Operator monotone functions

The prior analysis holds for any function $f(x)$ that is increasing, positive, and concave on the spectrum of psd matrix $\boldsymbol{A}$. The prior work (Persson & Kressner, 2023; Persson et al., 2025) on funNyström further required that f was *operator monotone*, meaning that $\mathbf{0} \preceq f(\boldsymbol{A}) \preceq f(\boldsymbol{A})$ for any matrices $\mathbf{0} \preceq \boldsymbol{A} \preceq \boldsymbol{B}$. All operator monotone functions are increasing, positive, and concave on $[0, \infty)$. Examples include $f(x) = \log(x+1)$, $f(x) = x(x+\lambda)^{-1}$ for any positive λ, and $f(x) = x^\alpha$ for any $\alpha \in [0, 1]$ (Bhatia, 2013, Chapter V).

While we achieve the matvec bound $\tilde{\mathcal{O}}(t_{(\varepsilon,f,\boldsymbol{A})} + \sqrt{t_{(\varepsilon,f,\boldsymbol{A})}}/\varepsilon)$ for any increasing concave f, we have better control on the degree $t_{(\varepsilon,f,\boldsymbol{A})}$ for operator monotone functions. First, we give theoretical bounds on the degree $t_{(\varepsilon,f,\boldsymbol{A})}$ for any operator monotone function:

Theorem 4.1 (Polynomial approximation of operator monotone functions)**.** *Let f be operator monotone, and let $r \geq 0$ be the radius of convergence for the Taylor expansion of f at $x = 0$. Let $0 \leq a \leq b$ and $\kappa = b/a$. Then any symmetric matrix $\boldsymbol{A}$ whose eigenvalues lie in the interval $[a, b]$ has*

$$t_{(\varepsilon,f,\boldsymbol{A})} \leq \frac{1}{2}\sqrt{\kappa}\log\left(\frac{1}{\varepsilon}\right) \qquad \textit{and} \qquad t_{(\varepsilon,f,\boldsymbol{A})} \lesssim \sqrt{b/r}\log\left(\frac{1}{\varepsilon}\right). \tag{6}$$

Theorem 4.1 is proven in Section A. The first bound in Equation (6) shows that all operator monotone functions and matrices $\boldsymbol{A}$ of condition number κ have $t_{(\varepsilon,f,\boldsymbol{A})} \lesssim \sqrt{\kappa}\log(1/\varepsilon)$. The second bound applies only to operator monotone functions without a singularity at $x = 0$, like the function $f(x) = \log(x+1)$, which has $r = 1$. In such cases, we get that $t_{(\varepsilon,f,\boldsymbol{A})} \lesssim \sqrt{\lambda_{\max}(\boldsymbol{A})}\log(1/\varepsilon)$, *independently of the condition number of $\boldsymbol{A}$*. So, for example, combining Theorem 4.1 with Theorem 3.2 gives

Corollary 4.2 (Regularized log determinant estimation)**.** *For the function $f(x) = \log(x+1)$, the accelerated funNyström++ algorithm (Algorithm 1) returns an estimate $\widehat{\mathrm{tr}}_{\mathrm{fN++}} \in (1 \pm \varepsilon)\,\mathrm{tr}(\log(\boldsymbol{A}+\mathbf{I}))$ with probability at least $9/10$ using $\mathcal{O}(\max\{\lambda_{\max}(\boldsymbol{A})^{1/2},\ \lambda_{\max}(\boldsymbol{A})^{1/4}/\varepsilon\})$ matvecs with $\boldsymbol{A}$.*

While Theorem 4.1 gives worst-case a-priori upper bounds on the degree $t_{(\varepsilon,f,\boldsymbol{A})}$, this upper bound can be maybe very pessimistic and requires knowing the spectrum of $\boldsymbol{A}$ in advance. So, to make accelerated funNyström++ practical, we need a method to estimate $t_{(\varepsilon,f,\boldsymbol{A})}$ on the fly.

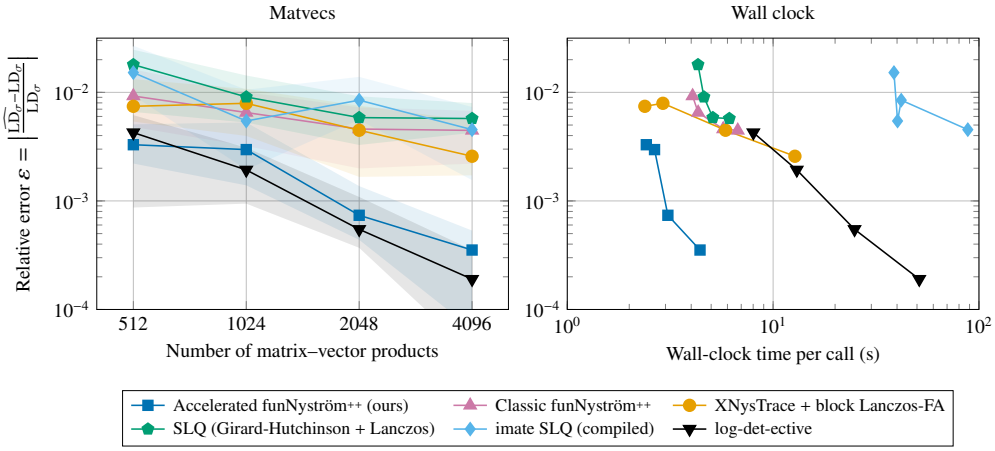


Figure 2: Estimating the regularized log-determinant. Computational resources required to estimate the regularized log-determinant $\mathrm{LD}_\sigma(\boldsymbol{A}) := \mathrm{tr}(\log(\boldsymbol{A} + \sigma\mathbf{I}))$ by various algorithms. Here, $\boldsymbol{A}$ is an empirical NTK matrix of size $n = 25{,}000$ from Ameli et al. (2025), for a MobileNet (Howard et al., 2017) trained on a 2,500-example subset of MNIST (LeCun et al., 1998), and $\sigma = 10$. Left: We see that our method, and the related log-det-ective method of (Cortinovis & Toni, 2026) both achieve about 10× lower error than competing methods. Right: Measured in wall-clock time, our implementation is much more efficient, producing **10× lower error while being 10× faster** than competitors. See Section 5.2 for details.

In Section C, we adapt the recent *block Gauss-Radau Lanzos-QF* method of Zimmerling et al. (2025) to automatically determine a degree $\tilde{t} \geq t_{(\varepsilon,f,\boldsymbol{A})}$, so that block Lanczos-QF run to depth t reliably converges to error ε. The resulting in the parameter-free accelerated funNyström++ is shown in . If f is operator monotone, then our convergence criteria is provably reliable. We show in that the criteria is also effective in practice for other increasing concave functions like $f(x) = x\log(x)$ and $f(x) = 1 - 1/x$. In Figure 2 and , we demonstrate the effectiveness of accelerated funNyström++ for the regularized log determinant.

5 Experiments

There are a great variety of computational problems in ML that can be reduced to the task of estimating $\mathrm{tr}(f(\boldsymbol{A}))$ for a positive increasing concave function f. In this section, we develop several such examples and show that accelerated funNyström++ reliably performs extremely well on real-world data. This is true in terms of both the number of matvecs computed and the wall clock time.

Remark 5.1 (Wall-clock time). We compare our implementation of accelerated funNyström++ against several codes that were designed to use as few matvecs as possible, but were written in MATLAB and were not designed to optimize wall-clock time. It is likely that some of these reference implementations can be made run faster. Nonetheless, the proof is in the pudding, and our optimized code is available now.

5.1 Experiment 1: Vendi Scores

An important topic in generative AI studies the *diversity* of data generated by an ML model. The *Vendi Score* is a recent and increasingly popular metric for the diversity of generated data.

We invoke the Vendi Score setup from Section F of (De Santi et al., 2025). Let $g : \mathcal{D} \to \mathbb{R}^m$ be an encoder which maps data from a domain $\mathcal{D}$ to embeddings in $\mathbb{R}^m$. If we let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathcal{D}$ be a collection of generated data, then we can construct the kernel matrix

$$[\boldsymbol{K}]_{ij} = k(g(\boldsymbol{x}_i), g(\boldsymbol{x}_j)),$$

for some kernel with $k(\boldsymbol{x}, \boldsymbol{x}) = 1$, often the cosine or RBF kernel. The Vendi Score is then the exponential of the von Neumann entropy of the kernel matrix:

$$\mathrm{VS}(\boldsymbol{K}) := \exp(\mathrm{VN}(K)) \quad \text{where} \quad \mathrm{VN}(\boldsymbol{K}) := -\mathrm{tr}(n^{-1}\boldsymbol{K}\log(n^{-1}\boldsymbol{K})).$$

The matrix function $f(x) = -x\log(x)$ is positive increasing and concave on the interval $[0, e^{-1}]$. So, we may apply accelerated funNyström^{++} to the matrix $n^{-1}\boldsymbol{K}$ so long as $\|\boldsymbol{K}\|_2 \leq e^{-1}n^2$. The size n of this matrix is often huge, making direct methods slow. Indeed, several prior works have pursued the goal of efficiently approximating Vendi Scores (Ospanov et al., 2024; Bilmes et al., 2026; Krebs et al., 2025; Ospanov & Farnia, 2025). In Figure 1, we use $n = 50{,}000$ embeddings from the SigLIP ImageNet (Zhai et al., 2023), constructed using the RBF kernel with different choices of lengthscale parameter equal to the median distance between.

Across the board, we see that our method has **4× to 16× lower error** than competing methods. The closest competitor is the original funNyström^{++} method of Persson & Kressner (2023), which does use the funNyström approximation, however it takes rank k equal to the number of Girard-Hutchinson samples ℓ.

5.2 Experiment 2: Regularized Log-determinant

The log-determinant is likely the most popular spectral sum in the ML literature, and there have been many efforts to efficiently compute the log-determinant of massive matrices, see e.g. (Han et al., 2015; Boutsidis et al., 2017; Cortinovis & Toni, 2026). Since $\log(0) = -\infty$, we often try to compute the *regularized log determiant* $\operatorname{tr}(\log(\boldsymbol{A} + \sigma\mathbf{I}))$ for some parameter $\sigma > 0$. Observe that

$$\operatorname{tr}(\log(\boldsymbol{A} + \sigma\mathbf{I})) = \operatorname{tr}(\log(\sigma^{-1}\boldsymbol{A} + \mathbf{I})) + n\log(\sigma).$$

So, to approximate the regularized log-determinant, we can take apply funNyström^{++} to to estimate $\operatorname{tr}(f(\sigma^{-1}\boldsymbol{A}))$ where $f(x) = \log(x + 1)$, and simply shift the result by $n + \log(1/\sigma)$. Motivated by recent work on the *Interpolating Information Criterion (IIC)*, we compute the log-determinant of the empirical neural tangent kernel (NTK) matrices of Ameli et al. (2025), for image classifiers trained on subsets of MNIST (LeCun et al., 1998) and CIFAR-10 (Krizhevsky, 2009). The IIC uses the log determinant, among other quantities, to provide usable a posteriori generalization error bounds for neural networks (Hodgkinson et al., 2023a;b).

The result of these experiments are shown in Figure 2. Our methods outperform all competitors except one: the log-det-ective method of Cortinovis & Toni (2026). Log-det-ective is conceptually similar to accelerated funNyström^{++}, in that both methods build a large low-rank approximation uses a small number of Girard-Hutchinson samples. However, it also uses properties of the log-determinant that do not generalize to other functions f. Hence, it is not surprising that we are merely competitive with log-det-ective for $f(x) = \log(x + 1)$, while both our method and log-det-ective outperform all others.

5.3 Further experiments and ablations

The appendix contains a variety of further experimental results validating both the efficacy of our code and validating the choices we made to design our code. In , we run experiments on various synthetic matrices that demonstrate the impact of the spectrum on the matvec complexity of spectral sum estimators, including a demonstration that the difficulty of approximation $\operatorname{tr}(\log(\boldsymbol{A} + \mathbf{I}))$ is independent of the condition number of $\boldsymbol{A}$. In , we shift our focus to wall-clock comparisons, showing that our method is very efficient across a variety of inputs. In , we breakdown what subroutine dominate the runtime of acclerated funNyström^{++}. In particular, we show that sparse sketching significantly speeds up our code, as does the exact numerical implementation of the eigendecomposition in the Nyström approximation. In , we demonstrate that the reliability of the Gauss and Gauss-Radau based method for determining the depth that block Lanczos should run to.

6 Conclusion

This work demonstrates that it is possible and practical to estimate spectral sums for positive increasing concave functions f with matvec complexity asymptotically lower than the composition of a matrix-function approximator and a trace estimator. A natural open question is if any broader categories of function f admit spectral sum estimation at a similarly accelerated rate. Another question

[2] Since $n^{-1}\boldsymbol{K}$ is psd with unit trace, $n^{-1}\boldsymbol{K}$ can only have 2 eigenvalues greater than e^{-1}. Though not explicitly covered by our theory, our low-rank approximation stage would recover any such isolated eigenvalues to very high accuracy, preventing the fact that $f(x)$ is decreasing on $[e^{-1}, 1]$ from being an issue in practice.

is if any other linear algebra problems about efficient approximating properties of $f(\boldsymbol{A})$ can also be accelerated when f is positive increasing and concave.

AI USE STATEMENT

Every word in this article is human-written, except for Section D where LLMs helped with copy-editing text. Otherwise, we used generative AI tools for three tasks in this project. First, LLMs were used to generate proofs of Theorems B.8, C.3 and 4.1. Those proofs were then digested by a human and then written in an approachable manner. Second, LLMs proposed using the Zimmerling et al. (2025) block Gauss and Gauss-Radau methods to automatically determine the depth t of our block Lanczos subspaces. Third, LLMs were integral in developing and quickly iterating code for this project. For the code, Claude Fable 5.1 (Anthropic) was used for planning and Claude Opus 5.5 (Anthropic) for execution. The authors reviewed all resulting code and text and take full responsibility for it.

REFERENCES

Alen Alexanderian, Noemi Petra, Georg Stadler, and Omar Ghattas. A-optimal design of experiments for infinite-dimensional Bayesian linear inverse problems with regularized ℓ_0-sparsification. *SIAM Journal on Scientific Computing*, 36(5):A2122–A2148, 2014.

Siavash Ameli. IMATE, a high-performance Python package for implicit matrix trace estimation. https://pypi.org/project/imate/, 2022.

Siavash Ameli and Shawn C. Shadden. Interpolating log-determinant and trace of the powers of matrix $\mathbf{A} + t\mathbf{B}$. *Statistics and Computing*, 32(6):108, 2022. doi: 10.1007/s11222-022-10173-4.

Siavash Ameli, Chris Van Der Heide, Liam Hodgkinson, Fred Roosta, and Michael W. Mahoney. Determinant estimation under memory constraints and neural scaling laws. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 1367–1400. PMLR, 2025. URL https://proceedings.mlr.press/v267/ameli25a.html.

Siavash Ameli, Chris van der Heide, Liam Hodgkinson, and Michael Mahoney. Spectral estimation with free decompression. *Advances in Neural Information Processing Systems*, 38:1888–1929, 2026.

Edward Anderson, Zhaojun Bai, Christian Bischof, L. Susan Blackford, James Demmel, Jack Dongarra, Jeremy Du Croz, Anne Greenbaum, Sven Hammarling, Alan McKenney, and Danny Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999. doi: 10.1137/1.9780898719604.

BallisticLA. RandBLAS: A header-only C++ library for sketching in randomized linear algebra. https://github.com/BallisticLA/RandBLAS, 2026. Accessed 2026.

Rajendra Bhatia. *Matrix analysis*. Springer Science & Business Media, 2013. doi: 10.1017/CBO9780511810817.

Jeff A Bilmes, Gantavya Bhatt, and Arnav M Das. How much is a dataset worth? scaling laws, the Vendi score, and matrix spectral functions. *arXiv preprint arXiv:2605.29448v2*, 2026.

Christos Boutsidis, Petros Drineas, Prabhanjan Kambadur, Eugenia-Maria Kontopoulou, and Anastasios Zouzias. A randomized algorithm for approximating the log determinant of a symmetric positive definite matrix. *Linear Algebra and its Applications*, 533:95–117, 2017.

Mark Braverman, Elad Hazan, Max Simchowitz, and Blake Woodworth. The gradient complexity of linear regression. In *Conference on Learning Theory*, pp. 627–647. PMLR, 2020. URL http://proceedings.mlr.press/v125/braverman20a.html.

Chris Camaño, Ethan N Epperly, Raphael A Meyer, and Joel A Tropp. Faster linear algebra algorithms with structured random matrices. *arXiv preprint arXiv:2508.21189v1*, 2025.

Chris Camaño, Ethan N Epperly, Raphael A Meyer, and Joel A Tropp. Linear algebra at exponential scale via tensor network dimension reduction. *arXiv preprint arXiv:2606.15350v1*, 2026.

Tyler Chen. The Lanczos algorithm for matrix functions: A handbook for scientists. *arXiv preprint arXiv:2410.11090v1*, 2024.

Tyler Chen and Eric Hallman. Krylov-aware stochastic trace estimation. *SIAM Journal on Matrix Analysis and Applications*, 44(3):1218–1244, August 2023. doi: 10.1137/22m1494257.

Tyler Chen, Thomas Trogdon, and Shashanka Ubaru. Randomized matrix-free quadrature: unified and uniform bounds for stochastic Lanczos quadrature and the kernel polynomial method. *SIAM Journal on Scientific Computing*, 47(3):A1733–A1757, 2025. doi: 10.1137/23M1600414.

Alice Cortinovis and Daniele Toni. Detecting when one probe vector is enough for preconditioned log-determinant approximation. *Numerical Linear Algebra with Applications*, 33(4), 2026. doi: 10.1002/nla.70106.

J. J. M. Cuppen. A divide and conquer method for the symmetric tridiagonal eigenproblem. *Numerische Mathematik*, 36(2):177–195, 1980. doi: 10.1007/BF01396757.

Riccardo De Santi, Marin Vlastelica, Ya-Ping Hsieh, Zebang Shen, Niao He, and Andreas Krause. Provable maximum entropy manifold exploration via diffusion models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 12739–12761. PMLR, 2025. URL https://dl.acm.org/doi/10.5555/3780338.3780828.

Michał Dereziński, Ethan N. Epperly, and Raphael A. Meyer. The matrix–vector complexity of $Ax = b$. In *Proceedings of the Thirty-Ninth Annual Conference on Learning Theory (COLT)*, volume 336 of *Proceedings of Machine Learning Research*, pp. 1737–1770. PMLR, 2026. URL https://proceedings.mlr.press/v336/derezinski26a.html.

Jack J. Dongarra, Jeremy Du Croz, Sven Hammarling, and Iain S. Duff. A set of level 3 basic linear algebra subprograms. *ACM Transactions on Mathematical Software*, 16(1):1–17, 1990. doi: 10.1145/77626.79170.

Ethan N. Epperly. Don't use Gaussians in stochastic trace estimation, November 2024. URL https://www.ethanepperly.com/index.php/2024/01/28/dont-use-gaussians-in-stochastic-trace-estimation/.

Ethan N. Epperly. Ethan-Epperly-Thesis. https://github.com/eepperly/Ethan-Epperly-Thesis, 2025. MATLAB code for E. N. Epperly's PhD thesis, *Make the Most of What You Have: Resource-Efficient Randomized Algorithms for Matrix Computations*; accessed 2026.

Ethan N. Epperly, Joel A. Tropp, and Robert J. Webber. XTrace: Making the most of every sample in stochastic trace estimation. *SIAM Journal on Matrix Analysis and Applications*, 45(1):1–23, 2024. doi: 10.1137/23M1548323.

Dan Friedman and Adji Bousso Dieng. The vendi score: A diversity evaluation metric for machine learning. *arXiv preprint arXiv:2210.02410va*, 2022.

Mark Gates, Piotr Luszczek, Ahmad Abdelfattah, Jakub Kurzak, Jack Dongarra, Konstantin Arturov, Cris Cecka, and Chip Freitag. C++ API for BLAS and LAPACK. SLATE Working Note 2, Innovative Computing Laboratory, University of Tennessee, 2017. ICL-UT-17-03.

Antoine Girard. A fast 'Monte-Carlo cross-validation'procedure for large least squares problems with noisy data. *Numerische Mathematik*, 56(1):1–23, 1989.

Alex Gittens and Michael Mahoney. Revisiting the Nyström method for improved large-scale machine learning. In *International Conference on Machine Learning*, pp. 567–575. PMLR, 2013. URL https://dl.acm.org/doi/abs/10.5555/2946645.3007070.

Ming Gu and Stanley C. Eisenstat. A divide-and-conquer algorithm for the bidiagonal SVD. *SIAM Journal on Matrix Analysis and Applications*, 16(1):79–92, 1995. doi: 10.1137/S0895479892242232.

Insu Han, Dmitry Malioutov, and Jinwoo Shin. Large-scale log-determinant computation through stochastic Chebyshev expansions. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 908–917, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/hana15.html.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Liam Hodgkinson, Chris van der Heide, Robert Salomone, Fred Roosta, and Michael W Mahoney. The interpolating information criterion for overparameterized models. *arXiv preprint arXiv:2307.07785va*, 2023a.

Liam Hodgkinson, Chris van der Heide, Robert Salomone, Fred Roosta, and Michael W Mahoney. A PAC-bayesian perspective on the interpolating information criterion. *arXiv preprint arXiv:2311.07013va*, 2023b.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861v1*, 2017.

Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989. doi: 10.1080/03610918908812806.

Intel Corporation. Intel oneAPI Math Kernel Library (oneMKL). https://www.intel.com/content/www/us/en/developer/tools/oneapi/onemkl.html, 2025.

Daniel M Kane and Jelani Nelson. Sparser Johnson-Lindenstrauss transforms. *Journal of the ACM (JACM)*, 61(1):1–23, 2014. URL https://dl.acm.org/doi/abs/10.5555/2095116.2095210.

Sebastian Krebs, Alexander Bernhardt, Bastian Goldlücke, and Christopher Knievel. Optimizing dataset quality and diversity in neural networks: A study of the Vendi score. In *2025 10th International Conference on Frontiers of Signal Processing (ICFSP)*, pp. 134–138. IEEE, 2025.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Department of Computer Science, University of Toronto, 2009.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Madhusudan Madhavan, Alen Alexanderian, and Arvind K. Saibaba. FlexTrace: Exchangeable randomized trace estimation for matrix functions. *arXiv preprint arXiv:2603.05721v1*, 2026.

M. W. Mahoney and P. Drineas. Structural properties underlying high-quality randomized numerical linear algebra algorithms. In P. Bühlmann, P. Drineas, M. Kane, and M. van de Laan (eds.), *Handbook of Big Data*, pp. 137–154. CRC Press, 2016. doi: 10.1201/b19567-16.

JC Mason and David C Handscomb. *Chebyshev Polynomials*. CRC Press, 2002. doi: 10.1201/9781420036114.

Raphael A Meyer, Cameron Musco, Christopher Musco, and David P Woodruff. Hutch++: Optimal stochastic trace estimation. In *Symposium on Simplicity in Algorithms (SOSA)*, pp. 142–155. SIAM, 2021. doi: 10.1137/1.9781611976496.

Raphael Arkady Meyer. *Towards Optimal Matrix-Vector Complexity in Numerical Linear Algebra*. PhD thesis, New York University Tandon School of Engineering, 2024.

Riley Murray, James Demmel, Michael W. Mahoney, N. Benjamin Erichson, Maksim Melnichenko, Osman Asif Malik, Laura Grigori, Piotr Luszczek, Michał Dereziński, Miles E. Lopes, Tianyu Liang, Hengrui Luo, and Jack Dongarra. Randomized numerical linear algebra: A perspective on the field with an eye to software. arXiv:2302.11474v2, 2023.

Azim Ospanov and Farzan Farnia. Do Vendi scores converge with finite samples? truncated Vendi score for finite-sample convergence guarantees. In *Proceedings of the Forty-first Conference on Uncertainty in Artificial Intelligence*, volume 286 of *Proceedings of Machine Learning Research*, pp. 3272–3299, 2025. URL https://proceedings.mlr.press/v286/ospanov25a.html.

Azim Ospanov, Jingwei Zhang, Mohammad Jalali, Xuenan Cao, Andrej Bogdanov, and Farzan Farnia. Towards a scalable reference-free evaluation of generative models. *Advances in Neural Information Processing Systems*, 37:120892–120927, 2024.

David Persson. funNystrom. https://github.com/davpersson/funNystrom, 2025. MATLAB code, version 1.0.0, doi: 10.5281/zenodo.15149263; accessed 2026.

David Persson and Daniel Kressner. Randomized low-rank approximation of monotone matrix functions. *SIAM Journal on Matrix Analysis and Applications*, 44(2):894–918, June 2023. doi: 10.1137/22M1523923.

David Persson, Alice Cortinovis, and Daniel Kressner. Improved variants of the Hutch++ algorithm for trace estimation. *SIAM Journal on Matrix Analysis and Applications*, 43(3):1162–1185, 2022. doi: 10.1137/21M1447623.

David Persson, Raphael A Meyer, and Christopher Musco. Algorithm-agnostic low-rank approximation of operator monotone matrix functions. *SIAM Journal on Matrix Analysis and Applications*, 46(1):1–21, 2025. doi: 10.1137/23M1619435.

Arvind K Saibaba and Peter K Kitanidis. Fast computation of uncertainty quantification measures in the geostatistical approach to solve inverse problems. *Advances in water resources*, 82:124–138, 2015.

John K. Salmon, Mark A. Moraes, Ron O. Dror, and David E. Shaw. Parallel random numbers: as easy as 1, 2, 3. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC '11)*, pp. 1–12. ACM, 2011. doi: 10.1145/2063384.2063405.

Xiaoming Sun, David P Woodruff, Guang Yang, and Jialin Zhang. Querying a matrix through matrix-vector products. *ACM Transactions on Algorithms (TALG)*, 17(4):1–19, 2021.

Daniele Toni. log-det. https://github.com/dantoni2000/log-det, 2026. MATLAB code, accessed 2026.

Lloyd N Trefethen. *Approximation Theory and Approximation Practice, Extended Edition*. SIAM, 2019. doi: 10.1137/1.9781611975949.

Joel A. Tropp. Comparison theorems for the minimum eigenvalue of a random positive-semidefinite matrix. *Comm. Amer. Math. Soc.*, 2026. doi: 10.1090/cams/72. To appear.

Joel A. Tropp and Robert J. Webber. Randomized algorithms for low-rank matrix approximation: Design, analysis, and applications. *arXiv preprint arXiv:2306.12418v3*, 2023.

Joel A Tropp, Alp Yurtsever, Madeleine Udell, and Volkan Cevher. Fixed-rank approximation of a positive-semidefinite matrix from streaming data. *Advances in Neural Information Processing Systems*, 30, 2017. URL https://dl.acm.org/doi/10.5555/3294771.3294888.

Joel A. Tropp, Alp Yurtsever, Madeleine Udell, and Volkan Cevher. Streaming low-rank matrix approximation with an application to scientific simulation. *SIAM Journal on Scientific Computing*, 41(4):A2430–A2463, 2019. doi: 10.1137/18M1201068.

Shashanka Ubaru, Jie Chen, and Yousef Saad. Fast estimation of tr(f(A)) via stochastic Lanczos quadrature. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1075–1099, 2017. doi: 10.1137/16M1104974.

Christopher K. I. Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'00, pp. 661–667, Cambridge, MA, USA, January 2000. MIT Press. URL https://dl.acm.org/doi/10.5555/3008751.3008847.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11941–11952. IEEE, 2023.

Jörn Zimmerling, Vladimir Druskin, and Valeria Simoncini. Monotonicity, bounds and acceleration of block Gauss and Gauss–Radau quadrature for computing $B^T\phi(A)B$. *Journal of Scientific Computing*, 103(1):5, 2025. doi: 10.1007/s10915-025-02799-z.

A Proof of Theorem 4.1

Central to the proof of Theorem 4.1 is the following classical result on operator monotone functions, which says that any operator monotone function can be represented as a linear combination of functions of the form $x \mapsto x/(1+\lambda x)$ over various $\lambda \geq 0$.

Imported Theorem A.1 (Integral form of operator monotone functions, (Bhatia, 2013, Eq. (V.53)))**.** *Let $f : \mathbb{R}_+ \to \mathbb{R}_+$ be operator monotone on $\mathbb{R}_+$. Then, there exists $\alpha, \beta \in \mathbb{R}_+$ and positive measure μ on $[0, \infty)$ such that*

$$f(x) = \alpha + \beta x + \int_0^\infty \frac{\lambda x}{\lambda + x} d\mu(\lambda).$$

From here, the proof Theorem 4.1 follows in three steps. First, we construct a polynomial approximation to the function $x \mapsto \lambda x/(\lambda + x)$ for any $\lambda > 0$:

Lemma A.2 (Polynomial approximation to $x \mapsto \lambda x/(\lambda + x)$)**.** *Let $0 < a < b$ and $\varepsilon > 0$, and define $\kappa = b/a$. Then, for any $\lambda > 0$, there exists a polynomial p_λ of degree $t \leq \frac{1}{2}\sqrt{\frac{b+\lambda}{a+\lambda}}\log(1/\varepsilon)$ such that*

$$\sup_{x \in [a,b]} \left| \frac{\lambda x}{\lambda + x} - p_\lambda(x) \right| \leq \varepsilon \frac{\lambda b}{\lambda + b}.$$

Once we have this result, we can immediately prove Theorem 4.1.

Proof of Theorem 4.1. Invoke Imported Theorem A.1 to write f in the integral form

$$f(x) = \alpha + \beta x + \int_0^\infty \frac{\lambda x}{\lambda + x} d\mu(\lambda)$$

for some $\alpha, \beta \in \mathbb{R}_+$ and positive measure μ on $[0, \infty)$. Now, introduce the polynomial

$$p(x) = \alpha + \beta x + \int_0^\infty p_\lambda(x) d\mu(\lambda).$$

Then, by the triangle inequality and Lemma A.2, for any $x \in [a, b]$ we have

$$\begin{aligned}
|f(x) - p(x)| &= \left| \int_0^\infty \left(\frac{\lambda x}{\lambda + x} - p_\lambda(x) \right) d\mu(\lambda) \right| \\
&\leq \int_0^\infty \left| \frac{\lambda x}{\lambda + x} - p_\lambda(x) \right| d\mu(\lambda) \\
&\leq \int_0^\infty \varepsilon \frac{\lambda b}{\lambda + b} d\mu(\lambda) \\
&\leq \varepsilon \left(\alpha + \beta x + \int_0^\infty \frac{\lambda b}{\lambda + b} d\mu(\lambda) \right) \\
&= \varepsilon f(b)
\end{aligned}$$

Finally, note that the degree of p is at most the maximum degree of any p_λ we integrated over. The degree of p is therefore at most $\max_{\lambda \geq 0}\{\frac{1}{2}\sqrt{\frac{b+\lambda}{a+\lambda}}\log(1/\varepsilon)\} = \frac{1}{2}\sqrt{\kappa}\log(1/\varepsilon)\}$. This completes the first part of the theorem.

If we further know that a power series for f centered at 0 converges in a radius $r > 0$, then Imported Theorem A.1 guarantees that μ has no mass on the interval $[0, r)$. In this case, our polynomial p has degree at most $\frac{1}{2}\sqrt{\frac{b+r}{a+r}}\log(1/\varepsilon) \leq \frac{1}{2}\left(\frac{b}{r} + 1\right)^{1/2}\log(1/\varepsilon)$. This completes the second part of the theorem. □

A.1 Proof of Lemma A.2

We now turn our attention to proving Lemma A.2. For this result, we will utilize the Chebyshev projection, whose most relevant idea is encapsulated below:

Imported Theorem A.3 (Chebyshev projection, Trefethen (2019, Eq. (4.8)))**.** *Let $f : [-1, 1] \to \mathbb{R}$ be a continous function. Let T_i denote the degree i Chebyshev polynomial of the first kind, and define the Chebyshev coefficients*

$$c_i = \frac{2}{\pi} \int_{-1}^{1} \frac{f(x)T_i(x)}{\sqrt{1-x^2}} dx.$$

Then, for all $t \geq 1$, there exists a polynomial p of degree t such that

$$\sup_{x\in[-1,1]} |f(x) - p(x)| \leq \sum_{i=t+1}^{\infty} |c_i| \tag{7}$$

Before tackling our function of interest, we first prove the following about a simpler function defined on $[-1, 1]$.

Lemma A.4 (Approximation of $x \mapsto 1/(x + \mu)$)**.** *Fix $\mu > 1$ and $g_\mu(x) = \frac{1}{x+\mu}$. For all integer $t \geq 1$, there exists a polynomial p of degree t such that*

$$\sup_{x\in[-1,1]} |g_\mu(x) - p(x)| \leq \frac{4\rho}{(\rho-1)^2(\rho+1)} \rho^{-t},$$

where $\rho = \mu + \sqrt{\mu^2 - 1}$.

Proof. Equation (5.14) of the textbook by Mason & Handscomb (2002) shows that the function g_μ has Chebyshev coefficients $c_i = (-1)^i 2\rho^{-i}/\sqrt{\mu^2 - 1}$. Activating Imported Theorem A.3, we find a polynomial p of degree t such that

$$\sup_{x\in[-1,1]} |g_\mu(x) - p(x)| \leq \sum_{i=t+1}^{\infty} |c_i| \leq \sum_{i=t+1}^{\infty} \frac{2\rho^{-i}}{\sqrt{\mu^2-1}} = \frac{2}{\sqrt{\mu^2-1}} \cdot \frac{\rho^{-t}}{\rho - 1}.$$

Since $\rho^{-1} = \mu - \sqrt{\mu^2 - 1}$, we have $\rho - \rho^{-1} = 2\sqrt{\mu^2 - 1}$. Substituting this expression in completes the proof. □

We have now setup enough pieces to prove the polynomial approximation of the function $f_\lambda(x) = \frac{\lambda x}{x+\lambda}$ shown in Lemma A.2.

Proof of Lemma A.2. Let $h(x) := -\frac{a+b-2x}{b-a}$ be the linear map from $[a, b]$ to $[-1, 1]$. Further, let $g_\mu(y) = \frac{1}{y+\mu}$ for $\mu = \frac{b+a+2\lambda}{b-a}$. With some algebra, we can verify that

$$\lambda - \frac{2\lambda^2}{b-a} g_\mu(h(x)) = \frac{\lambda x}{x+\lambda} = f(x).$$

Let q_μ be the polynomial guaranteed to exist by Lemma A.4 for the function g_μ. Then the degree t polynomial $p(x) = \lambda - \frac{2\lambda^2}{b-a} g_\mu(h(x))$ has

$$\sup_{x\in[a,b]} |p(x) - f_\lambda(x)| = \frac{2\lambda^2}{b-a} \sup_{y\in[-1,1]} |q_\mu(y) - g_\mu(y)| \leq \frac{8\lambda^2\rho}{(b-a)(\rho-1)^2(\rho+1)} \rho^{-t}.$$

Defining $c = \sqrt{(a+\lambda)(b+\lambda)} > 0$, we can verify that

$$\frac{8\lambda^2\rho}{(b-a)(\rho-1)^2(\rho+1)} = \lambda^2 \frac{b-a}{(a+\lambda)(b+\lambda+c)} \leq \lambda^2 \frac{b}{\lambda(b+\lambda)} = f_\lambda(b).$$

Further algebra reveals that $\rho = 1 + \frac{2}{\sqrt{\kappa_\lambda}-1}$ where $\kappa_\lambda := \frac{b+\lambda}{a+\lambda}$. Whence,

$$\sup_{x\in[a,b]} |p(x) - f_\lambda(x)| \leq \left(1 + \frac{2}{\sqrt{\kappa_\lambda} - 1}\right)^{-t} f_\lambda(b) \leq \varepsilon f_\lambda(b),$$

where the last inequality holds for $t \geq \frac{1}{2}\sqrt{\kappa_\lambda} \log(1/\varepsilon)$. □

B Single-pass FunNyström approximation

In this section, we prove that our procedure for constructing a funNyström low-rank approximation is theoretically sound. In order to maximize the efficiency of funNyström^{++} in practice, we ideally want to form a *single-pass* low-rank approximation using a *sparse test matrix*. That is, we only need to compute sparse matrix–vector products in order to produce our low-rank approximation. Since sparse arithmetic is extremely efficient, this further accelerate the low-rank approximation stage of our algorithm. Our theory focuses on the following specific construction of a sparse test matrix, called a *SparseStack*.

Definition B.1 (SparseStack, Kane & Nelson (2014); Camaño et al. (2025))**.** Fix ambient dimension n, block size b, and sparsity parameter ζ. Then a *SparseStack* of embedding dimension $k = b\zeta$ is the matrix $\boldsymbol{\Omega} \in \mathbb{R}^{n\times k}$ with

$$\boldsymbol{\Omega} = \frac{1}{\sqrt{\zeta}} \begin{bmatrix} \rho_{11}\mathbf{e}_{s_{11}}^\top & \cdots & \rho_{1\zeta}\mathbf{e}_{1\zeta}^\top \\ \rho_{21}\mathbf{e}_{s_{21}}^\top & \cdots & \rho_{2\zeta}\mathbf{e}_{2\zeta}^\top \\ \vdots & \ddots & \vdots \\ \rho_{n1}\mathbf{e}_{s_{n1}}^\top & \cdots & \rho_{n\zeta}\mathbf{e}_{n\zeta}^\top \end{bmatrix} \quad \text{where} \quad \begin{array}{l} \rho_{ij} \sim \textsc{uniform}(\{\pm 1\}), \\ s_{ij} \sim \textsc{uniform}(\{1, \dots, b\}). \end{array}$$

Here, $\mathbf{e}_i$ is the i^{th} standard basis vector in $\mathbb{R}^b$.

This sparse test matrix is known to be extremely reliable in practice while having as few as $\zeta = 4$ nonzeros per row (Camaño et al., 2025), and is amendable to strong theoretical guarantees. We prove the following result:

Theorem B.2 (Single-pass funNyström guarantee)**.** *Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ be a psd matrix, let $f : \mathbb{R}_+ \to \mathbb{R}_+$ be operator monotone on $\mathbb{R}_+$, and fix $r \in [n]$. Let $\boldsymbol{\Omega}$ be a SparseStack test matrix with parameter $\zeta \gtrsim \log(r)$ and embedding dimension $k \gtrsim r$. Let $\widehat{\boldsymbol{A}} = \boldsymbol{A}\boldsymbol{\Omega}(\boldsymbol{\Omega}^\top\boldsymbol{A}\boldsymbol{\Omega})^+\boldsymbol{\Omega}^\top\boldsymbol{A}$ be the Nyström approximation to $\boldsymbol{A}$ with respect to $\boldsymbol{\Omega}$. Then, with probability at least $19/20$, the funNyström approximation $f(\widehat{\boldsymbol{A}})$ is a near-optimal low-rank approximation to $f(\boldsymbol{A})$:*

$$\|f(\boldsymbol{A}) - f(\widehat{\boldsymbol{A}})\|_{\mathrm{F}}^2 \lesssim \|f(\boldsymbol{A}) - [\![f(\boldsymbol{A})]\!]_r\|_{\mathrm{F}}^2 + \frac{1}{r}\operatorname{tr}(f(\boldsymbol{A})). \tag{8}$$

If we cared only about low-rank approximation, this guarantee might not look very favorable as the term $\frac{1}{r}\operatorname{tr}(f(\boldsymbol{A}))$ decays slowly as a function of r even when $f(\boldsymbol{A})$ exhibits fast spectral decay. However, for the purposes of variance-reduced trace estimation, this result is sufficient to achieve the Hutch^{++} style $\mathcal{O}(1/\varepsilon)$ convergence rate. The proof of this result relies on two intermediary facts. The first fact says that $\widehat{\boldsymbol{A}}$ is a near-optimal low-rank approximation to $\boldsymbol{A}$ in a suitable sense. Similar guarantees are known if $\boldsymbol{\Omega}$ has iid Gaussian entries (Tropp & Webber, 2023) or has iid columns that obey certain variance bounds (Camaño et al., 2026). We are not aware of any work showing that sparse test matrices $\boldsymbol{\Omega}$ are able to achieve this.

Theorem B.3 (Single-pass Nyström approximation)**.** *Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ be psd, and fix rank parameter r. Let $\boldsymbol{\Omega} \in \mathbb{R}^{n\times k}$ be a SparseStack matrix with parameter $\zeta \gtrsim \log(r)$ and embedding dimension $k \gtrsim r$. Then, with probability at least $19/20$, the Nyström approximation $\widehat{\boldsymbol{A}} = \boldsymbol{A}\boldsymbol{\Omega}(\boldsymbol{\Omega}^\top\boldsymbol{A}\boldsymbol{\Omega})^+\boldsymbol{\Omega}^\top\boldsymbol{A}$ satisfies the bound*

$$\|\boldsymbol{A} - \widehat{\boldsymbol{A}}\|_{\mathrm{F}}^2 \lesssim \|\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r\|_{\mathrm{F}}^2 + \frac{1}{r}\operatorname{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r)^2. \tag{9}$$

We prove Theorem B.3 in Section B.1; our proof is an extension of a corresponding proof in (Camaño et al., 2026, Sec. 5.4). The second fact says that any low-rank approximation $\widehat{\boldsymbol{A}}$ that satisfies the error bound Equation (9) and has $\mathbf{0} \preceq \widehat{\boldsymbol{A}} \preceq \boldsymbol{A}$ achieves the funNyström guarantee Equation (8). This claim is fully deterministic, using only these two facts.

Theorem B.4 (Good Nyström approximation implies good funNyström approximation)**.** *Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ be psd, and fix rank parameter r. Let $\widehat{\boldsymbol{A}}$ be a matrix which satisfies Equation (9) and has $\mathbf{0} \preceq \widehat{\boldsymbol{A}} \preceq \boldsymbol{A}$. Let $f : \mathbb{R}_+ \to \mathbb{R}_+$ be increasing, positive, and have $x \mapsto \frac{f(x)}{x}$ decreasing on the interval $[0, \lambda_{\max}(\boldsymbol{A})]$. Then,*

$$\|f(\boldsymbol{A}) - f(\widehat{\boldsymbol{A}})\|_{\mathrm{F}}^2 \lesssim \|f(\boldsymbol{A}) - [\![f(\boldsymbol{A})]\!]_r\|_{\mathrm{F}}^2 + \frac{1}{r}\Big(\operatorname{tr}(f(\boldsymbol{A}))\Big)^2.$$

We prove Theorem B.4 in Section B.2. The main funNyström result we provide, Theorem B.2, is a direct corollary of Theorems B.3 and B.4.

B.1 Single-pass Nyström approximation

In this section, we prove Theorem B.3 by generalizing a technique in Section 5.4 of Camaño et al. (2026). The proof from the prior work requires our test matrix $\boldsymbol{\Omega}$ to have independent random columns, which is not the case for the SparseStack. However, since the SparseStack is constructed with independent *blocks* of columns, a fairly direct generalization of the existing argument can be applied to the SparseStack. We present this generalized analysis here, in its totality.

Lemma B.5. *Let $\boldsymbol{\Omega}$ be a SparseStack matrix with embedding dimensions k. For Hermitian $B \in \mathbb{C}^{n\times n}$, we have*

$$\mathbb{E}\left[\|\boldsymbol{\Omega}^* \boldsymbol{B}\boldsymbol{\Omega}\|_F^2\right] \leq \left(1 + \frac{1}{k}\right)\|\boldsymbol{B}\|_F^2 + \frac{1}{k}\Big(\mathrm{tr}(\boldsymbol{B})\Big)^2.$$

Proof. We break the Frobenius norm into the sum of submatrices

$$\mathbb{E}\left[\|\boldsymbol{\Omega}^\top \boldsymbol{B}\boldsymbol{\Omega}\|_F^2\right] = \frac{1}{\zeta^2}\sum_{i,j}^{\zeta}\mathbb{E}\left[\|\boldsymbol{\Phi}_i^\top \boldsymbol{B}\boldsymbol{\Phi}_j\|_{\mathrm{F}}^2\right] \quad \text{where} \quad \boldsymbol{\Phi}_i := \begin{bmatrix}\rho_{1i}\boldsymbol{e}_{s_{1i}} \\ \rho_{2i}\boldsymbol{e}_{s_{2i}} \\ \vdots \\ \rho_{ni}\boldsymbol{e}_{s_{1i}}\end{bmatrix}.$$

First we consider the case $i \neq j$. Using that $\mathbb{E}[\boldsymbol{\Phi}_i\boldsymbol{\Phi}_i^\top] = \mathbf{I}$, we have

$$\mathbb{E}\left[\|\boldsymbol{\Phi}_i^\top \boldsymbol{B}\boldsymbol{\Phi}_j\|_{\mathrm{F}}^2\right] = \mathrm{tr}\Big(\mathbb{E}[\boldsymbol{\Phi}_i\boldsymbol{\Phi}_i^\top]\boldsymbol{B}\,\mathbb{E}[\boldsymbol{\Phi}_j\boldsymbol{\Phi}_j^\top]\boldsymbol{B}\Big) = \mathrm{tr}(\boldsymbol{B}^2) = \|\boldsymbol{B}\|_{\mathrm{F}}^2.$$

In the case $i = j$, we pursue a more detailed analysis. Since $[\boldsymbol{\Phi}\boldsymbol{\Phi}^\top]_{k\ell} = \rho_{ki}\rho_{\ell i}\mathbb{1}_{[s_{ki}=s_{\ell i}]}$, we have

$$\mathbb{E}\left[\|\boldsymbol{\Phi}_i^\top \boldsymbol{B}\boldsymbol{\Phi}_i\|_{\mathrm{F}}^2\right] = \sum_{k,\ell,p,q=1}^{n}\boldsymbol{B}_{k\ell}\boldsymbol{B}_{pq}\,\mathbb{E}[\rho_{ki}\rho_{\ell i}\rho_{pi}\rho_{qi}]\,\mathbb{E}[\mathbb{1}_{[s_{ki}=s_{qi}]}\mathbb{1}_{[s_{\ell i}=s_{pi}]}].$$

With some algebra, can verify that

$$\mathbb{E}\left[\|\boldsymbol{\Phi}_i^\top \boldsymbol{B}\boldsymbol{\Phi}_i\|_{\mathrm{F}}^2\right] = \sum_{k,\ell=1}^{n}\boldsymbol{B}_{k\ell}^2 + \frac{1}{b}\sum_{\substack{k,\ell=1\\k\neq\ell}}^{n}\boldsymbol{B}_{k\ell}^2 + \frac{1}{b}\sum_{\substack{k,\ell=1\\k\neq\ell}}\boldsymbol{B}_{kk}\boldsymbol{B}_{\ell\ell} \leq \left(1+\frac{1}{b}\right)\|\boldsymbol{B}\|_{\mathrm{F}}^2 + \frac{1}{b}\Big(\mathrm{tr}(\boldsymbol{B})\Big)^2.$$

Thereby, recalling that $k = \zeta b$, we conclude that

$$\begin{aligned}\mathbb{E}\left[\|\boldsymbol{\Omega}^\top \boldsymbol{B}\boldsymbol{\Omega}\|_{\mathrm{F}}^2\right] &\leq \frac{\zeta(\zeta-1)}{\zeta^2}\|\boldsymbol{B}\|_{\mathrm{F}}^2 + \frac{1}{\zeta}\left(\left(1+\frac{1}{b}\right)\|\boldsymbol{B}\|_{\mathrm{F}}^2 + \frac{1}{b}\Big(\mathrm{tr}(\boldsymbol{B})\Big)^2\right) \\ &= \left(1+\frac{1}{k}\right)\|\boldsymbol{B}\|_{\mathrm{F}}^2 + \frac{1}{k}\Big(\mathrm{tr}(\boldsymbol{B})\Big)^2.\end{aligned}$$

□

The proof of Theorem B.3 pulls together several other prior results, which we state here. First is a structural bound on the error of a Nyström approximation, resulting from combining the results (Tropp & Webber, 2023, Prop. 8.5) and (Gittens & Mahoney, 2013, Lem. 1); see also (Mahoney & Drineas, 2016).

Imported Theorem B.6 (Nyström approximation error)**.** *Fix psd $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{\Omega} \in \mathbb{R}^{n\times k}$, and let $\hat{\boldsymbol{A}} = \boldsymbol{A}\boldsymbol{\Omega}(\boldsymbol{\Omega}^\top\boldsymbol{A}\boldsymbol{\Omega})^+\boldsymbol{\Omega}^\top\boldsymbol{A}$ be a Nyström approximation to $\boldsymbol{A}$. Partition $\boldsymbol{A}$ into its eigenvectors, letting $\boldsymbol{V}_1 \in \mathbb{R}^{n\times r}$ contain its top r eigenvectors and $\boldsymbol{V}_2 \in \mathbb{R}^{n\times(n-r)}$ contain the rest:*

$$\boldsymbol{A} = \boldsymbol{V}\begin{bmatrix}\boldsymbol{\Sigma}_1 & \\ & \boldsymbol{\Sigma}_2\end{bmatrix}\begin{bmatrix}\boldsymbol{V}_1^\top \\ \boldsymbol{V}_2^\top\end{bmatrix}.$$

Then, if $\boldsymbol{V}_1^\top\boldsymbol{\Omega}$ has rank r,

$$\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_F^2 \leq \|\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r\|_{\mathrm{F}}^2 + \|\sqrt{\boldsymbol{\Sigma}_2}(\boldsymbol{V}_2^\top\boldsymbol{\Omega})(\boldsymbol{V}_1^\top\boldsymbol{\Omega})^+\|_{(4)}^4$$

Imported Theorem B.7 (SparseStack injectivity, Tropp (2026, Rem. 6.5))**.** *Let $\boldsymbol{\Omega} \in \mathbb{R}^{n\times k}$ be a SparseStack matrix with embedding dimension $k \gtrsim r$ and parameter $\zeta \gtrsim \log r$. Let $\boldsymbol{Q} \in \mathbb{R}^{n\times r}$ be any matrix with orthonormal columns. Then $\mathbb{E}[\boldsymbol{\Omega}\boldsymbol{\Omega}^\top] = \mathbf{I}$ and, with probability at least $19/20$, we have $\sigma_{\min}(\boldsymbol{Q}^\top\boldsymbol{\Omega})^2 \geq 1/2$.*

Proof of Theorem B.3. Condition on the event that $\sigma^2_{\min}(\boldsymbol{V}_1^\top\boldsymbol{\Omega}) \geq 1/2$, which happens with probability at least $19/20$ by Imported Theorem B.7. Thereby, $\boldsymbol{V}_1^\top\boldsymbol{\Omega}$ has full column rank and we can apply Imported Theorem B.6 to bound

$$\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_{\mathrm{F}}^2 \leq \|\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r\|_{\mathrm{F}}^2 + \|\sqrt{\boldsymbol{\Sigma}_2}(\boldsymbol{V}_2^\top\boldsymbol{\Omega})(\boldsymbol{V}_1^\top\boldsymbol{\Omega})^+\|_{(4)}^4.$$

To complete the proof, it suffices to bound the second term above. Since $\|(\boldsymbol{V}_1^\top\boldsymbol{\Omega})^+\|_2^2 = 1/\sigma^2_{\min}(\boldsymbol{V}_1^\top\boldsymbol{\Omega}) \leq 2$ we can use the submultiplicative relation $\|\boldsymbol{E}\boldsymbol{F}\|_{(4)} \leq \|\boldsymbol{E}\|_{(4)}\|\boldsymbol{F}\|_2$ to bound

$$\|\sqrt{\boldsymbol{\Sigma}_2}(\boldsymbol{V}_2^\top\boldsymbol{\Omega})(\boldsymbol{V}_1^\top\boldsymbol{\Omega})^+\|_{(4)}^4 \leq 4\|\sqrt{\boldsymbol{\Sigma}_2}(\boldsymbol{V}_2^\top\boldsymbol{\Omega})\|_{(4)}^4.$$

From the expectation bound in Lemma B.5,

$$\begin{aligned}\mathbb{E}\left[4\|\sqrt{\boldsymbol{\Sigma}_2}(\boldsymbol{V}_2^\top\boldsymbol{\Omega})\|_{(4)}^4\right] &= \mathbb{E}\left[4\|\boldsymbol{\Omega}^\top(\boldsymbol{V}_2\boldsymbol{\Sigma}_2\boldsymbol{V}_2^\top)\boldsymbol{\Omega}\|_{\mathrm{F}}^2\right] \\ &\leq \left(1 + \frac{1}{k}\right)\|\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r\|_{\mathrm{F}}^2 + \frac{1}{k}\Big(\mathrm{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r)\Big)^2.\end{aligned}$$

Finally, applying Markov's inequality gives us the desired result with constant probability. □

B.2 From Nyström to funNyström

In this section, we prove Theorem B.4. We state the result in a slightly more precise way here.

Theorem B.8 (Good Nyström implies good funNyström approximation, general form)**.** *Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ be pdf, and fix rank parameter r. Let $\widehat{\boldsymbol{A}}$ be a matrix which satisfies $\mathbf{0} \preceq \widehat{\boldsymbol{A}} \preceq \boldsymbol{A}$ and*

$$\|\widehat{\boldsymbol{A}} - \boldsymbol{A}\|_{\mathrm{F}}^2 \leq \mathrm{C}_2\|\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r\|_{\mathrm{F}}^2 + \mathrm{C}_1(\mathrm{tr}(\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r))^2, \tag{10}$$

for some parameters $\mathrm{C}_2 \geq 1$ and $\mathrm{C}_1 \geq 0$. Let f be a continuous non-decreasing positive function on $[0, \lambda_{\max}(\boldsymbol{A})]$, and that $x \mapsto f(x)/x$ is non-increasing on $[0, \lambda_{\max}(\boldsymbol{A})]$. Then,

$$\|f(\widehat{\boldsymbol{A}}) - f(\boldsymbol{A})\|_{\mathrm{F}}^2 \leq \mathrm{C}_2\|f(\boldsymbol{A}) - [\![f(\boldsymbol{A})]\!]_r\|_{\mathrm{F}}^2 + \mathrm{C}_1(\mathrm{tr}(f(\boldsymbol{A}) - [\![f(\boldsymbol{A})]\!]_r))^2.$$

The proof of this claim moves in two parts. First, we prove a technical lemma about masking non-negative entries of a matrix using a majorization-like relationship. Second, we relate the norm $\|\widehat{\boldsymbol{A}} - \boldsymbol{A}\|_{\mathrm{F}}^2$

B.2.1 Technical lemmas

In this section, we develop a variety of short technical lemmas that are helpful throughout this argument.

Lemma B.9 (Non-negative mask)**.** *Let $\boldsymbol{W} \in \mathbb{R}^{n\times n}$, and let $\mathbf{e} \in \mathbb{R}^n$ denote the all-ones vector. Suppose that the row sums $\boldsymbol{\rho} = \boldsymbol{W}\mathbf{e}$ and the column sums $\boldsymbol{\sigma} = \boldsymbol{W}^\top\mathbf{e}$ are all bounded between 0 and 1. Let $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$ be vectors such that*

$$\sum_{i:a_i\geq t} \rho_i \geq \sum_{j:b_j\geq t} \sigma_j \quad \text{for all thresholds } t \in \mathbb{R}. \tag{11}$$

Then, there exists a matrix $\boldsymbol{M} \in \mathbb{R}^{n\times n}$ such that

1. ***(Nonnegative)*** *$\boldsymbol{M}_{i,j} \geq 0$,*
2. ***(Row and Column Sums)*** *$\boldsymbol{M}\mathbf{e} = \boldsymbol{\rho}$ and $\boldsymbol{M}^\top\mathbf{e} = \boldsymbol{\sigma}$,*
3. ***(Masking)*** *If $a_i \leq b_j$ then $\boldsymbol{M}_{i,j} = 0$.*

Proof. Without loss of generality, we permute the input so that $a_1 \geq \cdots \geq a_n$ and $b_1 \geq \cdots \geq b_n$. We construct the matrix $\boldsymbol{M}$ with a simple recursively dynamic program, via the formula

$$\boldsymbol{M}_{i,j} = \min\left\{\rho_i - \sum\nolimits_{\ell<j} \boldsymbol{M}_{i,\ell},\ \sigma_j - \sum\nolimits_{k<i} \boldsymbol{M}_{k,j}\right\}.$$

The rest of the proof verifies that this formula correctly satisfies these three requirements. We start by defining the running sum of row sums and columns sums, as well as the of entries above and left of the (i, j) entry.

$$R_i := \sum\nolimits_{k=1}^{i} \rho_k, \quad C_j := \sum\nolimits_{\ell=1}^{j} \sigma_\ell, \quad P_{i,j} := \sum\nolimits_{k=1}^{i} \sum\nolimits_{\ell=1}^{j} \boldsymbol{M}_{k,\ell}.$$

First, we demonstrate via induction that $P_{i,j} = \min\{R_i, C_i\}$. In the base case, we have $P_{1,1} = \min\{R_1, C_1\}$. By induction we assume that $P_{j,k} = \min\{R_j, C_k\}$ for all $j + k < i + j$. Then, we expand

$$\begin{aligned}
P_{i,j} &= \min\{P_{i-1,j} + \rho_i,\ P_{i,j-1} + \sigma_j\} \\
&= \min\{R_{i-1} + \rho_i,\ C_j + \rho_i,\ R_i + \sigma_j,\ C_{j-1} + \sigma_j\} \\
&= \min\{R_i,\ C_j + \rho_i,\ R_i + \sigma_j,\ C_j\}. && \text{(Definition of } R_i \text{ and } C_j\text{)} \\
&= \min\{R_i,\ C_j\}. && (\rho_i, \sigma_i \geq 0)
\end{aligned}$$

Now, we show that $\boldsymbol{M}_{i,j}$ are all non-negative. We do this by showing that both terms in the minimum that define $\boldsymbol{M}_{i,j}$ are non-negative. For any i, j, expand

$$\sum\nolimits_{\ell=1}^{j-1} \boldsymbol{M}_{i,\ell} = P_{i,j-1} - P_{i-1,j-1} = \min\{R_i, C_{j-1}\} - \min\{R_{i-1}, C_{j-1}\} \leq R_i - R_{i-1} = \rho_i$$

and therefore that $\rho_i - \sum_{\ell<j} \boldsymbol{M}_{i,\ell} \geq 0$. An analogous argument shows that $\sigma_j - \sum_{k<i} \boldsymbol{M}_{k,j} \geq 0$.

To observe the control over row sums, note that $\sum_{j=1}^{n} \sigma_j = \mathbf{e}^\top \boldsymbol{W} \mathbf{e} = \sum_{i=1}^{n} \rho_1$, and therefore that $C_n = \sum_{j=1}^{n} \sigma_j = \sum_{k=1}^{n} \rho_k \geq R_i$ for all i. So,

$$\sum\nolimits_{j=1}^{n} \boldsymbol{M}_{i,j} = P_{i,n} - P_{i-1,n} = R_i - R_{i-1} = \rho_i,$$

with an analogous argument recovering the column sums.

For the third point, fix any $a_i < b_j$. Because we sorted $\boldsymbol{a}$ and $\boldsymbol{b}$, we know that if $a_k \geq b_j$ then $k \leq i-1$, so

$$\sum_{k:a_k \geq b_j} \rho_k \leq \sum\nolimits_{k=1}^{i-1} \rho_k = R_{i-1}.$$

Further, we have $b_\ell \geq b_j$ for all $\ell \in \{1, \cdots, j\}$, so

$$\sum_{\ell:b_\ell \geq b_j} \sigma_\ell = \sum\nolimits_{\ell=1}^{j} \sigma_k = C_j.$$

Therefore, using the threshold $t = b_j$, we have that

$$R_{i-1} \geq \sum_{k:a_k \geq b_j} \rho_k \geq \sum_{\ell:b_\ell \geq b_j} \sigma_\ell = C_j,$$

and hence that $R_i \geq R_{i-1} \geq C_j \geq C_{j-1}$. Finally, this lets us recover

$$\boldsymbol{M}_{i,j} = P_{i,j} - P_{i-1,j} - P_{i,j-1} + P_{i-1,j-1} = C_j - C_j - C_{j-1} + C_{j-1} = 0.$$

□

Fact B.10 (Properties of f)**.** *Let f be positive and non-decreasing on an interval $\mathcal{S}$, and suppose that $x \mapsto f(x)/x$ is non-increasing on $\mathcal{S}$. Then, for all $a, b \in \mathcal{S}$ with $a \leq b$, we have*

$$\Big(f(a) - f(b)\Big)^2 \leq \frac{a-b}{a+b}\Big((f(a))^2 - (f(b))^2\Big)$$

and

$$(f(a))^2 - (f(b))^2 \leq \left(\tfrac{f(a)}{a}\right)^2 (a^2 - b^2).$$

B.2.2 Reduction to a linear program

Let $\mathbf{0} \prec \boldsymbol{B} \preceq \boldsymbol{A}$ be matrices with eigendecompositions $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top$ and $\boldsymbol{B} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{V}^\top$. Let a_i be the i^{th} eigenvalue of $\boldsymbol{A}$, and b_i the i^{th} eigenvalue of $\boldsymbol{B}$. Define the matrix $\boldsymbol{C} = \boldsymbol{U}^\top\boldsymbol{V}$, so that $\boldsymbol{C}_{i,j} = \langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle$ is the inner product between eigenvectors of $\boldsymbol{A}$ and $\boldsymbol{B}$. Then, since $\boldsymbol{U}^\top(\boldsymbol{A} - \boldsymbol{B})\boldsymbol{V} = \boldsymbol{\Lambda}\boldsymbol{C} - \boldsymbol{C}\boldsymbol{\Sigma}$, we have

$$\left[\boldsymbol{U}^\top(\boldsymbol{A} - \boldsymbol{B})\boldsymbol{V}\right]_{i,j}^2 = \boldsymbol{C}_{i,j}^2(a_i - b_j)^2 = \boldsymbol{C}_{i,j}^2\left(\tfrac{a_i - b_j}{a_i + b_j}\right)(a_i^2 - b_j^2) = \boldsymbol{W}_{i,j}(a_i^2 - b_j^2)$$

where we have defined $\boldsymbol{W} \in \mathbb{R}^{n \times n}$ by $\boldsymbol{W}_{i,j} := \langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle^2 \cdot \frac{a_i - b_j}{a_i + b_j}$. Let $\boldsymbol{\rho} = \boldsymbol{W}\mathbf{e}$ and $\boldsymbol{\sigma} = \boldsymbol{W}^\top\mathbf{e}$ be the vectors of row sums and columns sums of $\boldsymbol{W}$. We can then expand the squared norm

$$\|\boldsymbol{A} - \boldsymbol{B}\|_{\mathrm{F}}^2 = \sum_{i,j=1}^n \boldsymbol{W}_{i,j}(a_i^2 - b_j^2) = \sum_{i=1}^n a_i^2\rho_i - \sum_{j=1}^n b_j^2\sigma_j \tag{12}$$

That is, we have managed to write down the squared norm between $\boldsymbol{A}$ and $\boldsymbol{B}$ using two disjoint sums, where the first sum contains all of the eigenvalues of $\boldsymbol{A}$ and the second sum contains all of the eigenvalues of $\boldsymbol{B}$. However, there are complicated correlations between the row sums ρ_i and columns sums σ_j that depend on the overlap of eigenvalues and eigenvectors between $\boldsymbol{A}$ and $\boldsymbol{B}$. Momentarily, we will appeal to Lemma B.9 in order to replace $\boldsymbol{W}$ with a simpler $\boldsymbol{M}$ matrix.

Before doing so, however, we will first demonstrate the broader idea behind the proof. Since $f(\boldsymbol{A})$ and $f(\boldsymbol{B})$ have the same eigenvectors as $\boldsymbol{A}$ and $\boldsymbol{B}$, respectively, we can expand

$$\begin{aligned}
\|f(\boldsymbol{A}) - f(\boldsymbol{B})\|_{\mathrm{F}}^2 &= \textstyle\sum_{i,j=1}^n \boldsymbol{C}_{i,j}^2(f(a_i) - f(b_j))^2 \\
&\leq \textstyle\sum_{i,j=1}^n \boldsymbol{W}_{i,j}\left(\left(f(a_i)\right)^2 - \left(f(b_j)\right)^2\right) && \text{(Fact B.10)} \\
&= \textstyle\sum_{i=1}^n \left(f(a_i)\right)^2\rho_i - \sum_{j=1}^n \left(f(b_i)\right)^2\sigma_j. && (13)
\end{aligned}$$

Notice that this bound on $\|f(\boldsymbol{A}) - f(\boldsymbol{B})\|_{\mathrm{F}}^2$ exactly recover Equation (12) but where we have replaced the eigenvalues of $\boldsymbol{A}$ and $\boldsymbol{B}$ with the eigenvalues of $f(\boldsymbol{A})$ and $f(\boldsymbol{B})$. The final stage of this proof will use this relationship to bound the error $\|f(\boldsymbol{A}) - f(\boldsymbol{B})\|_{\mathrm{F}}^2$ in terms of how small $\|\boldsymbol{A} - \boldsymbol{B}\|_{\mathrm{F}}^2$ is. Further, the fact that both Equation (12) and Equation (13) are linear in the (squared) eigenvalues of the matrices involved is crucial to the analysis.

Before we proceed however, we first verify that $\boldsymbol{W}$ satisfies the requirements of Lemma B.9 First, we show that the row and column sums ρ_i and σ_j of $\boldsymbol{W}$ are bounded above by 1. Since $\frac{a_i - b_j}{a_i + b_j}$, we trivially have

$$\rho_i = \textstyle\sum_{j=1}^n \boldsymbol{W}_{i,j} \leq \sum_{j=1}^n \langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle^2 = 1,$$

with an analogous proof holding for column sums. To show that the row sums are non-negative, we first write $\frac{a_i - b_j}{a_i + b_j} = \frac{2a_i}{a_i + b_j} - 1$, so that

$$\rho_i = 2a_i \textstyle\sum_{j=1}^n \frac{\langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle^2}{a_i + b_j} - 1 = 2a_i\boldsymbol{u}_i^\top(\boldsymbol{B} + a_i\mathbf{I})^{-1}\boldsymbol{u}_i - 1.$$

Since $\boldsymbol{B} \preceq \boldsymbol{A}$, and recalling that $\boldsymbol{u}_i$ is the i^{th} eigenvector of $\boldsymbol{A}$, we have

$$\rho_i \geq 2a_i\boldsymbol{u}_i^\top(\boldsymbol{A} + a_i\mathbf{I})^{-1}\boldsymbol{u}_i - 1 = 2a_i \cdot \frac{1}{2a_i} - 1 = 0,$$

with an analogous proof holding for column sums. Lastly, we show that the vectors $\boldsymbol{a} = [a_1 \ \cdots \ a_n]^\top$ and $\boldsymbol{b} = [b_1 \ \cdots \ b_n]^\top$ satisfy the relationship Equation (11). First, note that $\boldsymbol{W}_{i,j} \geq 0$ when $a_i \geq b_i$ and $\boldsymbol{W}_{i,j} \leq 0$ only if $a_i \leq b_i$. Then, for any $t \in \mathbb{R}$, we may expand

$$\sum_{i:a_i \geq t} \rho_i - \sum_{j:b_j \geq t} \sigma_j = \sum_{\substack{i:a_i \geq t \\ j:b_j < t}} \boldsymbol{W}_{i,j} - \sum_{\substack{j:b_j \geq t \\ i:a_i < t}} \boldsymbol{W}_{i,j} \geq 0,$$

where the inequality holds because all terms in the first summation are non-negative while all terms in the second summation are non-positive. Therefore, we may invoke Lemma B.9 to construct a mask $\boldsymbol{M} \in \mathbb{R}^{n\times n}$ with the enilsted properties. Since the row sums of $\boldsymbol{M}$ match those of $\boldsymbol{W}$, we have

$$\|\boldsymbol{A} - \boldsymbol{B}\|_{\mathrm{F}}^2 = \sum_{i=1}^{n} a_i^2\rho_i - \sum_{j=1}^{n} b_j^2\sigma_j = \sum_{i,j=1}^{n} \boldsymbol{M}_{i,j}(a_i^2 - b_i^2)$$

and

$$\|f(\boldsymbol{A}) - f(\boldsymbol{B})\|_{\mathrm{F}}^2 \le \sum_{i=1}^{n} (f(a_i))^2\rho_i - \sum_{j=1}^{n} (f(b_i))^2\sigma_j = \sum_{i,j=1}^{n} \boldsymbol{M}_{i,j}\Big((f(a_i))^2 - (f(b_i))^2\Big),$$

where $\boldsymbol{M}_{i,j} \ge 0$, where the row sums of $\boldsymbol{M}$ are at most 1, and where $\boldsymbol{M}_{i,j}$ is nonzero only if $a_i \ge b_j$. The sums above are therefore all only nonnegative on entries where $a_i \ge b_j$. For the rest of the proof, we therefore assume this relation holds.

Next we use the second point from Fact B.10 to say that

$$(f(a_i))^2 - (f(b_i))^2 \le \theta_i(a_i^2 - b_i^2) \quad \text{where } \theta_i := \left(\frac{f(a_i)}{a_i}\right)^2.$$

B.2.3 A fractional knapsack problem

Let $\gamma_i := \sum_j \boldsymbol{M}_{i,j}(a_i^2 - b_i^2) = a_i^2 \sum_j \boldsymbol{M}_{i,j} - \sum_j \boldsymbol{M}_{i,j} b_j^2$. By the properties of $\boldsymbol{M}$,

$$0 \le \gamma_i \le a_i^2 \quad \text{and} \quad \sum\nolimits_i \gamma_i = \|\boldsymbol{A} - \boldsymbol{B}\|_{\mathrm{F}}^2.$$

So, we have that $\|f(\boldsymbol{A}) - f(\boldsymbol{B})\|_{\mathrm{F}}^2 \le \sum_i \theta_i\gamma_i$. We have now reduced the problem to a fractional knapsack problem; a very simple linear program.

Indeed, let us examine the maximum possible value that $\|f(\boldsymbol{A}) - f(\boldsymbol{B})\|_{\mathrm{F}}^2$ may take subject to these constraints:

$$\max_{\gamma_i} \sum\nolimits_i \theta_i\gamma_i \quad \text{s.t. } 0 \le \gamma_i \le a_i^2,\ \sum \gamma_i \le U$$

where U is our given upper bound on $\|\boldsymbol{A} - \boldsymbol{B}\|_{\mathrm{F}}^2$:

$$U := \alpha \sum\nolimits_{i=r+1}^{n} a_i^2 + \beta\left(\sum\nolimits_{i=r+1}^{n} a_i\right)^2.$$

A greedy construction solves this linear program. Since all $\theta_i \ge 0$, we construct γ_i by first setting $\gamma_i = a_i^2$ for all $i \ge r+1$. This is always possible since $\alpha \ge 1$. The remaining budget of $\sum_i \gamma_i$ is

$$(\alpha - 1) \sum\nolimits_{i=r+1}^{n} a_i^2 + \beta\left(\sum\nolimits_{i=r+1}^{n} a_i\right)^2.$$

and can only be used by putting positive mass on $\gamma_1, \ldots, \gamma_r$. Let $\bar{\theta} = \theta_{r+1}$, which since $f(x)/x$ is decreasing, we have $\theta_i \le \bar{\theta}$ for $i \le r$ and otherwise have $\theta_i \ge \bar{\theta}$. Therefore,

$$\begin{aligned}
\sum\nolimits_i \theta_i\gamma_i &= \sum\nolimits_{i=r+1}^{n} (f(a_i))^2 + \sum\nolimits_{i=1}^{r} \theta_i\gamma_i \\
&\le \sum\nolimits_{i=r+1}^{n} (f(a_i))^2 + \bar{\theta} \sum\nolimits_{i=1}^{r} \gamma_i \\
&\le \sum\nolimits_{i=r+1}^{n} (f(a_i))^2 + \bar{\theta}\left(U - \sum\nolimits_{i=r+1}^{n} \theta_i\right) \\
&= \sum\nolimits_{i=r+1}^{n} (f(a_i))^2 + \bar{\theta}(\alpha - 1) \sum\nolimits_{i=r+1}^{n} a_i^2 + \beta\left(\sum\nolimits_{i=r+1}^{n} a_i\right)^2 \\
&\le \alpha \sum\nolimits_{i=r+1}^{n} (f(a_i))^2 + \beta\left(\sum\nolimits_{i=r+1}^{n} f(a_i)\right)^2
\end{aligned}$$

Which completes the proof of Theorem B.8.

Algorithm 2. Block Lanczos Tridiagonalization, (Chen, 2024)

Input: Matrix–vector access to matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, matrix $\boldsymbol{B} \in \mathbb{R}^{n\times \ell}$, degree t.
Output: Block-Tridiagonal $\boldsymbol{T} = \textsc{BlockTridiagonal}(\{\boldsymbol{\alpha}_i\}_{i=0}^{t-1}, \{\boldsymbol{\beta}_i\}_{i=0}^{t-2})$, matrix $\boldsymbol{\beta}_{-1} \in \mathbb{R}^{\ell\times\ell}$

Initialize $\hat{\boldsymbol{Q}}_{-1} = \boldsymbol{0}$
$\hat{\boldsymbol{Q}}_0, \boldsymbol{\beta}_{-1} = \textsc{qr}(\boldsymbol{B})$
for $i = 0, \ldots, t-1$ **do**
  $\boldsymbol{Y}_{i+1} = \boldsymbol{A}\hat{\boldsymbol{Q}}_i - \boldsymbol{Q}_{i-1}\boldsymbol{\beta}_{i-1}^\top$
  $\boldsymbol{\alpha}_i = \hat{\boldsymbol{Q}}_i^\top \boldsymbol{Y}_{i+1}$
  $\boldsymbol{Z}_{i+1} = \boldsymbol{Y}_{i+1} - \hat{\boldsymbol{Q}}_i\boldsymbol{\alpha}_i$
  Reorthogonalize $\boldsymbol{Z}_{i+1}$ against $\hat{\boldsymbol{Q}}_0, \ldots, \hat{\boldsymbol{Q}}_i$ ▷ Optional
  $\hat{\boldsymbol{Q}}_{i+1}, \boldsymbol{\beta}_i = \textsc{qr}(\boldsymbol{Z}_{i+1})$
end for
return $\boldsymbol{T} = \textsc{BlockTridiagonal}(\{\boldsymbol{\alpha}_i\}_{i=0}^{t-1}, \{\boldsymbol{\beta}_i\}_{i=0}^{t-2}), \boldsymbol{\beta}_{-1}$

Algorithm 3. Block Lanczos-QF, (Chen, 2024)

Input: Lanczos decomposition $\boldsymbol{T}_t = \textsc{BlockTridiagonal}(\{\boldsymbol{\alpha}_i\}_{i=0}^{t-1}, \{\boldsymbol{\beta}_i\}_{i=0}^{t-2}), \boldsymbol{\beta}_{-1}$
Output: Approximation $\boldsymbol{L}$ to $\boldsymbol{B}^\top f(\boldsymbol{A})\boldsymbol{B}$

Eigen-decompose $\boldsymbol{T}_t = \boldsymbol{U}^\top \boldsymbol{\Lambda} \boldsymbol{U}$
Let $\boldsymbol{W} = \boldsymbol{U}[:, 1:\ell]^\top f(\boldsymbol{\Lambda}[1:\ell])\boldsymbol{U}[:, 1:\ell] \in \mathbb{R}^{\ell\times\ell}$
return $\boldsymbol{L} = \boldsymbol{\beta}_{-1}^\top \boldsymbol{W} \boldsymbol{\beta}_{-1}$

C Block Lanczos-QF

The block Lanczos-QF method is a deterministic method for approximating the matrix $\boldsymbol{B}^\top f(\boldsymbol{A})\boldsymbol{B}$ using a small number of matvecs with $\boldsymbol{A}$. The following theorem provides entry error bounds on the output of block Lanczos-QF.

Imported Theorem C.1 (Block Lanczos-QF error bound, Chen (2024, Cor. 9.11))**.** *Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ be a matrix with eigenvalues in the interval $[a, b]$. Fix any function $f : [a, b] \to \mathbb{R}$, and let $\boldsymbol{B} = [\boldsymbol{b}_1\ \boldsymbol{b}_2\ \cdots\ \boldsymbol{b}_\ell] \in \mathbb{R}^{n\times\ell}$ Then the degree t Lanczos-QF approximation $\boldsymbol{L} = \text{BLanczos-QF}(\boldsymbol{A}, \boldsymbol{B}, f, t)$ satisfies the bound*

$$|\boldsymbol{L}_{i,j} - \boldsymbol{b}_i^\top f(\boldsymbol{A})\boldsymbol{b}_j| \le 2\|\boldsymbol{b}_i\|_2\|\boldsymbol{b}_j\|_2 \min_{\deg(p)\le 2k} \max_{x\in[a,b]} |f(x) - p(x)|.$$

Our interest is in estimating $\operatorname{tr}(\boldsymbol{B}^\top f(\boldsymbol{A})\boldsymbol{B})$. For this purpose, an immediate corollary of the above result is helpful:

Corollary C.2 (Trace error of Block Lanczos-QF)**.** *Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ be a matrix with eigenvalues in the interval $[a, b]$. Fix any function $f : [a, b] \to \mathbb{R}$, and let $\boldsymbol{B} \in \mathbb{R}^{n\times\ell}$ Then the degree t Lanczos-QF approximation $\boldsymbol{L} = \text{BLanczos-QF}(\boldsymbol{A}, \boldsymbol{B}, f, t)$ satisfies the bound*

$$\left|\operatorname{tr}(\boldsymbol{L}) - \operatorname{tr}(\boldsymbol{B}\top f(\boldsymbol{A})\boldsymbol{B})\right| \le 2\|\boldsymbol{B}\|_F^2 \min_{\deg(p)\le 2k} \max_{x\in[a,b]} |f(x) - p(x)|.$$

We can use Corollary C.2 to bound the a priori error of block Lanczos-QF. However, if we do not know $t_{(\varepsilon,f,\boldsymbol{A})}$ in advance or want to avoid wasting computation when Corollary C.2 is pessimistic, we should use an adaptive stopping criteria based on a posteriori error estimation. The following fact contrasts usual block Lanczos-QF, related to Gauss quadrature, to a slightly different block Lanczos-QF method, related to Gauss-Radau quadrature. In standard block Lanczos-QF, we construct the block Lanczos recurrence with the block Tridiagonal matrix $\boldsymbol{T}_t = \textsc{BlockTridiagonal}(\{\boldsymbol{\alpha}_i\}_{i=0}^{t-1}, \{\boldsymbol{\beta}_i\}_{i=0}^{t-2})$ and initialization matrix $\boldsymbol{\beta}_{-1}$, and return the Gauss quadrature approximation

$$\boldsymbol{L}_t := \boldsymbol{\beta}_{-1}^\top \boldsymbol{E}_1^\top f(\boldsymbol{T}_t)\boldsymbol{E}_1\boldsymbol{\beta}_{-1},$$

where $\boldsymbol{E}_1 \in \mathbb{R}^{n\times\ell}$ contains the first ℓ columns of the identity on $\mathbb{R}^n$. In the block Gauss-Radau case, as constructed in Zimmerling et al. (2025), we slightly alter the tridiagonal matrix to be

$$\tilde{\boldsymbol{T}}_t = \textsc{BlockTridiagonal}\Big(\{\boldsymbol{\alpha}_0, \ldots, \boldsymbol{\alpha}_{t-2}, \tilde{\boldsymbol{\alpha}}_{t-1}\}, \{\boldsymbol{\beta}_i\}_{i=0}^{t-2}\Big) \text{ where } \tilde{\boldsymbol{\alpha}}_{t-1} := \boldsymbol{\beta}_{t-2}(\boldsymbol{T}_{t-1})^{-1}\boldsymbol{\beta}_{t-2}^\top.$$

Algorithm 4. Block Lanczos-QF, Gauss-Radau variant

Input: Lanczos decomposition $\boldsymbol{T}_t = \textsc{BlockTridiagonal}(\{\boldsymbol{\alpha}_i\}_{i=0}^{t-1}, \{\boldsymbol{\beta}_i\}_{i=0}^{t-2})$, $\boldsymbol{\beta}_{-1}$
Output: Approximation $\tilde{\boldsymbol{L}}$ to $\boldsymbol{B}^\top f(\boldsymbol{A})\boldsymbol{B}$
1 Compute $\tilde{\boldsymbol{\alpha}}_{t-1} = \boldsymbol{\beta}_{t-2}(\boldsymbol{T}_{t-1})^{-1}\boldsymbol{\beta}_{t-2}^\top$.
2 Form $\tilde{\boldsymbol{T}}_t = \textsc{BlockTridiagonal}(\{\boldsymbol{\alpha}_0, \ldots, \boldsymbol{\alpha}_{t-2}, \tilde{\boldsymbol{\alpha}}_{t-1}\}, \{\boldsymbol{\beta}_i\}_{i=0}^{t-2})$.
3 Eigen-decompose $\tilde{\boldsymbol{T}}_t = \boldsymbol{U}^\top \boldsymbol{\Lambda} \boldsymbol{U}$
4 Let $\boldsymbol{W} = \boldsymbol{U}[:, 1:\ell]^\top f(\boldsymbol{\Lambda}[1:\ell])\boldsymbol{U}[:, 1:\ell] \in \mathbb{R}^{\ell \times \ell}$
5 **return** $\tilde{\boldsymbol{L}} = \boldsymbol{\beta}_{-1}^\top \boldsymbol{W} \boldsymbol{\beta}_{-1}$

We then return the Gauss-Radau quadrature approximation

$$\tilde{\boldsymbol{L}}_t := \boldsymbol{\beta}_{-1}^\top \boldsymbol{E}_1^\top f(\tilde{\boldsymbol{T}}_t)\boldsymbol{E}_1 \boldsymbol{\beta}_{-1}.$$

In the following, we show that the matrices $\boldsymbol{L}_t$ and $\tilde{\boldsymbol{L}}_t$ can be used to determine a simple and provably accurate stopping criteria for the block Lanczos method on operator monotone functions.

Theorem C.3 (Adaptive stopping criteria)**.** *Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ be psd, let $\boldsymbol{B} \in \mathbb{R}^{n\times \ell}$, and let $f : \mathbb{R}_+ \to \mathbb{R}_+$ be operator monotone. Let $\boldsymbol{L}_t$ and $\tilde{\boldsymbol{L}}_t$ be the degree t Gauss and Gauss-Radau approximations of $\boldsymbol{B}^\top f(\boldsymbol{A})\boldsymbol{B}$. We have that*

$$|\operatorname{tr}(\boldsymbol{L}_t) - \operatorname{tr}(\boldsymbol{B}^\top f(\boldsymbol{A})\boldsymbol{B})| \leq \varepsilon \operatorname{tr}(\boldsymbol{B}^\top f(\boldsymbol{A})\boldsymbol{B}) \quad \text{if} \quad |\operatorname{tr}(\boldsymbol{L}_t) - \operatorname{tr}(\tilde{\boldsymbol{L}}_t)| \leq \varepsilon \operatorname{tr}(\boldsymbol{L}_t). \tag{14}$$

Further, the gap $|\operatorname{tr}(\boldsymbol{L}_t) - \operatorname{tr}(\tilde{\boldsymbol{L}}_t)|$ is monotonically decreasing in t.

Equation (14) gives us a simple and formal stopping criteria: we check if $|\operatorname{tr}(\boldsymbol{L}_t) - \operatorname{tr}(\tilde{\boldsymbol{L}}_t)| \leq \varepsilon \operatorname{tr}(\boldsymbol{L}_t)$. The fact that the error metric decreases monotonically ensures that this stopping criteria is reliable.

The key theoretical idea underpinning Theorem C.3 arises from that same work of Zimmerling et al. (2025). Namely, for the function $g_\lambda(x) = 1/(x+\lambda)$, Gauss quadrature systematically underestimates $\boldsymbol{B}^\top g_\lambda(\boldsymbol{A})\boldsymbol{B}$ while Gauss-Radau systematically overestimates it.

Imported Theorem C.4 (Gauss and Gauss-Radau for ridge functions, Zimmerling et al. (2025, Prop. 3))**.** *Let $g_\lambda(x) := (x+\lambda)^{-1}$ for any $\lambda > 0$. Fix any matrices $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\boldsymbol{B} \in \mathbb{R}^{n\times \ell}$, and integer $t \geq 2$. Let $\boldsymbol{T}_t = \textsc{BlockTridiagonal}(\{\boldsymbol{\alpha}_i\}_{i=0}^{t-1}, \{\boldsymbol{\beta}_i\}_{i=0}^{t-2})$ be the depth t block tridiagonal matrix resulting from the Block Lanczos proces on $\boldsymbol{A}$ generated by $\boldsymbol{B}$, so that $\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i \in \mathbb{R}^{\ell\times\ell}$. Then let $\boldsymbol{L}_{t,\lambda}$ and $\tilde{\boldsymbol{L}}_{t,\lambda}$ be the degree t Gauss and Gauss-Radau approximations constructed from the function g_λ. Then,*

$$\boldsymbol{L}_{t-1,\lambda} \preceq \boldsymbol{L}_\lambda \preceq \boldsymbol{B}^\top g_\lambda(\boldsymbol{A})\boldsymbol{B} \preceq \tilde{\boldsymbol{L}}_{t,\lambda} \preceq \tilde{\boldsymbol{L}}_{t-1,\lambda}.$$

To recover Theorem C.3 from Imported Theorem C.4, we first recall from Imported Theorem A.1 that any operator monotone f can be written as

$$f(x) = c_0 + c_1 x + \int_0^\infty \left(\lambda - \lambda^2 g_\lambda(x)\right) d\mu(\lambda)$$

for some measure μ on $[0, \infty)$. Therefore, we can write the Gauss and Gauss-Radau approximations as

$$\boldsymbol{L}_t = c_0 \boldsymbol{\beta}_{-1}^\top \boldsymbol{\beta}_{-1} + c_1 \boldsymbol{\beta}_{-1}^\top \boldsymbol{E}_1^\top \boldsymbol{T}_t \boldsymbol{E}_1 \boldsymbol{\beta}_{-1} + \int_0^\infty \left(\lambda \boldsymbol{\beta}_{-1}^\top \boldsymbol{\beta}_{-1} - \lambda^2 \boldsymbol{L}_{t,\lambda}\right) d\mu(\lambda),$$
$$\tilde{\boldsymbol{L}}_t = c_0 \boldsymbol{\beta}_{-1}^\top \boldsymbol{\beta}_{-1} + c_1 \boldsymbol{\beta}_{-1}^\top \boldsymbol{E}_1^\top \tilde{\boldsymbol{T}}_t \boldsymbol{E}_1 \boldsymbol{\beta}_{-1} + \int_0^\infty \left(\lambda \boldsymbol{\beta}_{-1}^\top \boldsymbol{\beta}_{-1} - \lambda^2 \tilde{\boldsymbol{L}}_{t,\lambda}\right) d\mu(\lambda).$$

Since these are merely linear in $\boldsymbol{L}_{t,\lambda}$, we can directly apply Imported Theorem C.4 pointwise for each $\lambda \geq 0$, proving that

$$\tilde{\boldsymbol{L}}_{t-1,\lambda} \preceq \tilde{\boldsymbol{L}}_\lambda \preceq \boldsymbol{B}^\top f(\boldsymbol{A})\boldsymbol{B} \preceq \boldsymbol{L}_{t,\lambda} \preceq \boldsymbol{L}_{t-1,\lambda}. \tag{15}$$

Notice that the prefactor of -1 in from of $\boldsymbol{L}_{t,\lambda}$ and $\tilde{\boldsymbol{L}}_{t,\lambda}$ above has cause the psd ordering to invert. Now, the Gauss-Radau is the underestimate and the Gauss is the overestimate. Theorem C.3 is now a direct consequence of applying the trace to every term in Equation (15):

$$\operatorname{tr}(\tilde{\boldsymbol{L}}_{t-1,\lambda}) \leq \operatorname{tr}(\tilde{\boldsymbol{L}}_\lambda) \leq \operatorname{tr}(\boldsymbol{B}^\top f(\boldsymbol{A})\boldsymbol{B}) \leq \operatorname{tr}(\boldsymbol{L}_{t,\lambda}) \leq \operatorname{tr}(\boldsymbol{L}_{t-1,\lambda}).$$

D Computational details

This appendix describes implementation details of accelerated funNyström^{++}, the setup of our experiments, and further experimental results. We do not attempt to describe every implementation detail, as such could fill a paper of its own. Instead, the implementation notes are a selection of details that we found instructive and useful for practitioners implementing similar methods.

Throughout this appendix, the *sample vectors* are the ℓ random vectors of the Girard-Hutchinson estimator in Section 3, and the *Lanczos phase* is the part of accelerated funNyström^{++} that applies $f(\boldsymbol{A})$ to them with Lanczos; our implementation draws the sample vectors uniformly from the sphere of radius $\sqrt{n}$ rather than with random signs, which keeps the estimator unbiased and is well behaved (Epperly, 2024). *Depth selection* is a short preliminary Lanczos run that chooses the depth t (Section D.5).

D.1 Our software

Accelerated funNyström^{++} and its components (the Nyström approximation, including its self-tuning variant, the scalar and block Lanczos quadrature routines with their Gauss and Gauss-Radau stopping rules, and the symmetric linear operator interface they act on) are implemented as part of RandLAPACK, an open-source C++ library of high-performance RandNLA algorithms. Its companion library RandBLAS (BallisticLA, 2026) generates and applies sketching operators, using the counter-based random number generators of the Random123 library (Salmon et al., 2011). RandLAPACK obtains dense linear algebra kernels through BLAS++ and LAPACK++ (Gates et al., 2017); these are thin C++ wrappers over vendor-optimized BLAS (Dongarra et al., 1990) and LAPACK (Anderson et al., 1999). All experiments in this paper use Intel oneMKL (Intel Corporation, 2025) (MKL below) as BLAS and LAPACK backend. RandBLAS also provides some sparse BLAS functionality, both through its own kernels and by wrapping MKL's sparse routines. The code used in this paper, including the MATLAB binding and the benchmark scripts, is available at `https://anonymous.4open.science/r/accelerated-funnystrompp-1E1B`.

We compare RandLAPACK's implementation of accelerated funNyström^{++} against the following methods:

- *Classic funNyström^{++}*, the original funNyström^{++} of Persson & Kressner (2023), which uses the Nyström routine of the authors' MATLAB code (Persson, 2025);
- *XNysTrace* (Epperly et al., 2024), from the MATLAB code of Epperly (2025), applied to $f(\boldsymbol{A})$ with the action of $f(\boldsymbol{A})$ computed by block Lanczos-FA at twice the selected block Lanczos depth;
- *Log-det-ective* (Cortinovis & Toni, 2026), using the authors' MATLAB routines (Toni, 2026);
- *Stochastic Lanczos quadrature* (Girard-Hutchinson with Lanczos), both in our own MATLAB implementation and in the compiled implementation of the imate package (Ameli & Shadden, 2022; Ameli, 2022).

Because every method except imate, which we drive from Python, is implemented in MATLAB, we run every benchmark from MATLAB and call RandLAPACK through a MEX function (MATLAB's interface to compiled C and C++ code), part of a binding layer that is still under development. All methods run from MATLAB see the same matrices, trial seeds, matvec budgets and thread counts, and each derives its own random stream from the trial seed. Where a competitor needs a Lanczos-based matrix function action or quadrature (Classic funNyström^{++}, XNysTrace and our stochastic Lanczos quadrature), we supply a blocked MATLAB implementation that applies $\boldsymbol{A}$ to all vectors at once (see *Fair baselines* in Section D.3).

D.2 Experiments configuration

Every experiment that compares the running times of different methods, namely the wall-clock panels of Figures 1 and 2 and Figure 7, ran on the Cascade Lake nodes of Table 1, and the algorithm performance breakdown of Figure 3 ran on its Ice Lake node. Experiments that compare only errors and matvec counts, which do not depend on the hardware beyond rounding, ran wherever capacity

was free: the Gauss-Radau bracket of Figure 4 and about 30% of the synthetic runs of Figures 5 and 6 ran on a 16-core laptop (Intel Core Ultra 9 386H), and all other runs on the nodes. The laptop also produced two timings quoted in the text, and neither compares methods: both compare implementations of a single routine, the sparse sketching kernel (Section D.4) and the blocked Lanczos routines of the competitors (Section D.3); the other timings of single routines in these sections come from the cluster nodes. Each timing run had a node to itself and ran 48 threads, set through `OMP_NUM_THREADS`, `MKL_NUM_THREADS` and MATLAB's `maxNumCompThreads`, with no thread pinning or explicit memory-placement policy such as `numactl`. For the wall-clock comparisons, each matrix is loaded once and then copied by all threads, which spreads its memory pages over both sockets (Section D.3); after one untimed warm-up call, every method runs three timed trials at each matvec budget, and we plot the median time. The timed trials use the first three of the 20 trial seeds of the error runs and reproduce those runs' estimates to within 10^{-10} relative difference, consistent with the thread- and processor-dependent order of floating-point reductions (*Experiment reproducibility* in Section D.3); the error axis in all plots shows the median over all 20 trials. Wall-clock time is measured in MATLAB around the entire call, so for our method it includes copying the matrix into RandLAPACK. Each method that runs Lanczos at a fixed depth takes that depth from one depth-selection run (Section D.5) per matrix and Lanczos variant (scalar or block), which is shared by all budgets and excluded from both matvec counts and wall-clock time; XNysTrace, which we pair with block Lanczos-FA, runs at twice the block depth. Log-det-ective needs no depth selection and runs with the parameters recommended by its authors. The self-tuned variant of funNyström++ (Algorithm 7) is generally not shown in plots, since it would be the only method charged for its depth selection. The algorithm performance breakdown in Figure 3 departs from this protocol: it reports the median of 10 trials and charges depth selection to the budget, showing it as a separate layer. All experiments use IEEE 754 binary64 (double-precision) arithmetic.

Table 1: Hardware and software used for the timed experiments.

Hardware	**Cascade Lake node** *all wall-clock figures*	**Ice Lake node** *Figure 3*
Processor	2 × Intel Xeon Gold 6248R	2 × Intel Xeon Gold 6348
Cores per socket	24	28
Threads used	48	48
Base / boost clock	3.0 / 4.0 GHz	2.6 / 3.5 GHz
L1d / L2 cache per core	32 KB / 1 MB	48 KB / 1.25 MB
L3 cache per socket	35.75 MB	42 MB
Memory	192 GB DDR4	256 GB DDR4
Nominal FP64 peak	4.6 TFLOP/s	4.7 TFLOP/s (4.0 on 48 cores)
Software	*same on both nodes*	
Operating system	Red Hat Enterprise Linux 8.10	
MATLAB	R2025b	
Linear algebra	Intel oneMKL 2024.1, bundled with MATLAB (dense routines) Intel oneMKL 2025.3 (sparse routines in our code)	
C++ compiler	GCC 13.4.0 (`-O3`, C++20, OpenMP)	
CMake	3.30.5	
Python	3.11.11 with imate 0.25.2	

D.3 Benchmark infrastructure

Linear algebra backend. In our experiments, RandLAPACK is built against MKL 2025.3, while MATLAB R2025b bundles MKL 2024.1. Inside MATLAB, however, the dynamic linker resolves RandLAPACK's dense BLAS and LAPACK calls to MATLAB's already loaded MKL, so our method and the MATLAB baselines run on the same dense kernels. Only MKL's sparse routines, which are not part of the standard BLAS and LAPACK interfaces, come from MKL 2025.3. Users who want a single MKL version throughout can arrange one: MATLAB loads another BLAS and LAPACK when pointed to them through its documented `BLAS_VERSION` and

`LAPACK_VERSION` environment variables, and older oneMKL releases remain available from PyPI (for example `mkl==2024.1.0`) and conda.

Passing input problems from MATLAB to C++ efficiently. The MEX function copies the input matrix $\boldsymbol{A}$ into a buffer owned by RandLAPACK; MATLAB passes $\boldsymbol{A}$ read-only, since a writable reference would make its copy-on-write duplicate the whole matrix first. A sparse $\boldsymbol{A}$ is passed as compressed-column arrays built on the MATLAB side with `find`, because MATLAB's C++ Data API exposes a sparse matrix only element by element (55 s instead of 35 ms for a matrix with $n = 80{,}000$). On a two-socket node, each socket has its own memory, and Linux places each page in the memory of the socket whose thread first writes it (non-uniform memory access, or NUMA), so a single-threaded copy such as `memcpy` or `std::copy` puts the whole matrix (20 GB at $n = 50{,}000$) on one socket and limits every later multithreaded product with $\boldsymbol{A}$ to one memory controller. The slowdown is easy to miss: it does not occur on a single-socket machine and leaves every result unchanged. We therefore allocate the buffer without zero-initialization and fill it with a copy made by all threads; at $n = 50{,}000$ this made one product take 0.25 to 0.30 s instead of 0.60 to 0.79 s and cut the copy itself from 6.5 s to 0.4 s per call, with bitwise identical estimates. MATLAB's own products are affected in the same way (one product with a four-column block went from 0.517 s to 0.199 s after re-copying the matrix with all threads), so our timing runs re-copy each matrix with all threads before timing any method.

Fair experimental baselines. Equal matvec counts do not imply equal running times when memory traffic differs: applying $\boldsymbol{A}$ to one vector at a time reads the whole matrix once per vector, and on the Cascade Lake nodes, for the kernel matrices with $n = 10{,}000$ to $25{,}000$, it took 17 to 37 times as long per vector as applying $\boldsymbol{A}$ to a block of 32 vectors; on the MNIST NTK, one product with a 32-column block was even faster than one product with a single vector. The Lanczos routines of the MATLAB competitors (Classic funNyström^{++}, XNysTrace and our stochastic Lanczos quadrature) originally applied $\boldsymbol{A}$ to one vector at a time, so we rewrote them to advance the recurrences of all sample vectors together, applying $\boldsymbol{A}$ to one block of vectors per step while leaving the recurrence of each sample vector unchanged; this made them 11.8 to 14.1 times faster at $n = 4{,}000$ on the laptop and changed their results by at most 5×10^{-14} relative. We did this to be fair to these methods: without it, their wall-clock times would reflect how the matrix is applied rather than the algorithms themselves. Log-det-ective, the remaining MATLAB competitor, is the exception: its reference code applies $\boldsymbol{A}$ to one sample vector at a time inside its preconditioned Lanczos iteration, and we run it unchanged.

Experiment reproducibility. Inside RandLAPACK, all random matrices, namely the sketching matrix $\boldsymbol{\Omega}$ and the sample vectors, are generated by RandBLAS with the counter-based Philox generator of Random123, which assigns counters by position in the matrix rather than by thread, so they are identical for any number of threads. Floating-point reductions in MKL and in our OpenMP loops do depend on the thread count, so results are bitwise reproducible only for a fixed thread count, processor model and MKL version. After every rebuild of the library, a fixed set of test cases must reproduce the recorded estimates, matvec counts and Lanczos depths to the last bit, and every result file records the versions of the library, the MATLAB binding and the benchmark scripts that produced it.

D.4 Tuning the subroutines of accelerated funNyström^{++}

Applying the input operator efficiently. Our method never needs the entries of $\boldsymbol{A}$, only its products with blocks of vectors, so RandLAPACK takes the input as an abstract symmetric linear operator: an object that knows its dimension and computes products of $\boldsymbol{A}$ with blocks of vectors. RandLAPACK provides such operators for dense, sparse and diagonal symmetric matrices, and a user can supply their own. For a dense matrix the natural choice is the BLAS routine `symm` (Dongarra et al., 1990), which multiplies a symmetric matrix stored as one triangle by a block of vectors; the general matrix-matrix product `gemm` instead reads the whole matrix, which MATLAB stores anyway. At $n = 50{,}000$, MKL's `gemm` was never slower than `symm` at the block sizes we measured between 2 and 256 vectors, and up to 1.7 times faster; it was slightly slower for a single vector. Just past 256 columns its time jumps (0.72 s at 256 columns, 1.70 s at 264 and 2.02 s at 269, against

about 1.25 s for `symm`), so our option to use `gemm`, which the algorithm performance breakdown in Figure 3 uses, is restricted to blocks of 2 to 256 vectors.

Sparse sketching. RandBLAS does not yet implement the SparseStack test matrix of Section B, for which our theory is stated, so our implementation uses its short-axis-sparse sketching operator (SASO) instead (Murray et al., 2023, Sec. 2.4.1). A SASO $\boldsymbol{\Omega} \in \mathbb{R}^{n\times k}, n > k$, has the same number of nonzeros in every row, each equal to ± 1 up to scaling, at positions drawn uniformly without replacement from the k columns; a SparseStack is the variant that instead places one nonzero in each of ζ fixed blocks of columns. We use 8 nonzeros per row, following the recommendation of Tropp et al. (2019, Sec. 3.3), although 4 is widely reported to work well (Tropp et al., 2019; Camaño et al., 2025). In our setting the choice barely affects the running time: in 53 runs on the leading 50,000 × 50,000 block of the empirical NTK of a ResNet-9 trained on CIFAR-10 (Ameli et al., 2025) (nine values of the share p of the budget spent on the Nyström approximation, as in Figure 3, by six budgets), the algorithm time summed over the runs was 1,815 s with 8 nonzeros per row and 1,803 s with 4. Since the input is an abstract operator, the sketch $\boldsymbol{A\Omega}$ is computed by the operator itself, and each kind of operator needs its own way of applying a sparse $\boldsymbol{\Omega}$. For a dense symmetric operator, which, like `symm`, reads only one triangle of $\boldsymbol{A}$, this is harder than it looks. MKL's general sparse routines read both upper and lower triangles of the input, so the triangle must first be mirrored, and our first single-triangle kernel avoided the mirror but was 12 to 21 times slower than MKL, because it traversed the matrix row by row with strided reads. The RandBLAS kernel we use instead packs 64-column panels of the symmetric matrix into thread-private buffers and streams the nonzeros of the sketch over them as contiguous vectorized updates, preserving their order so that the result does not depend on the number of threads; at $n = 4{,}096$ and $k = 64$, on four threads of the laptop, it takes 13.8 ms, against 20.0 ms for MKL on an already full matrix and 30.9 ms when the triangle must first be mirrored. At $n = 50{,}000$ the per-thread panels (26 MB) no longer fit in cache, and for sketch sizes k of about 1,000 and above a general sparse product on the full matrix (RandBLAS's `right_spmm`, which dispatches to MKL's sparse routines) is faster; our code offers it as an option, which the algorithm performance breakdown in Figure 3 uses.

Replacing the SVD with an eigendecomposition in the Nyström approximation. Given the sketch $\boldsymbol{Y} = \boldsymbol{A\Omega}$ and a small shift $\nu > 0$, the single-pass Nyström approximation of Tropp et al. (2017) and Camaño et al. (2025, Alg. 2) ends with a thin SVD of the $n\times k$ matrix $\boldsymbol{W} = (\boldsymbol{Y}+\nu\boldsymbol{\Omega})\boldsymbol{R}^{-\top}$, where $\boldsymbol{R}\boldsymbol{R}^\top = \boldsymbol{\Omega}^\top(\boldsymbol{Y} + \nu\boldsymbol{\Omega})$ is a Cholesky factorization, and needs only its singular values and left singular vectors. We instead implement Algorithm 5, following the Gram-matrix approach of Camaño et al. (2026, Alg. 2). Our implementation instead forms the $k \times k$ matrix $\boldsymbol{W}^\top\boldsymbol{W}$ (the matrix $\boldsymbol{E}$ of Algorithm 5) with the BLAS routine `syrk`, computes its eigendecomposition $\boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^\top$ with `syevd`, LAPACK's divide-and-conquer symmetric eigensolver (Cuppen, 1980), and recovers $\boldsymbol{U} = \boldsymbol{W}\boldsymbol{V}\boldsymbol{\Sigma}^{-1}$ with `gemm`. At $n = 50{,}000$ this is 2.5 to 3.5 times faster than `gesdd`, LAPACK's divide-and-conquer SVD (Gu & Eisenstat, 1995) and usually its fastest SVD driver when singular vectors are needed; at $k = 7{,}276$ it takes 7.5 s instead of 26.5 s, where `gesdd` had sometimes consumed 90% of the compute time spent after the sketch. Forming $\boldsymbol{W}^\top\boldsymbol{W}$ squares the singular values, so eigenvalues below about $\varepsilon_{\text{mach}}\|\boldsymbol{W}\|_2^2$ are lost and those slightly above it lose relative accuracy, where $\varepsilon_{\text{mach}}$ is the unit roundoff. The affected eigenvalues that lie near the shift ν anyways get removed by the next step. On 36 test cases with matrix condition numbers up to 10^9, switching to the eigendecomposition changed the estimates of $\operatorname{tr}(f(\boldsymbol{A}))$ returned by accelerated funNyström++ by at most 2.8×10^{-14} relative in double precision. Because RandLAPACK is templated for single and double precision, we also checked single precision, where the change reached 1.3×10^{-4}. So, we use the eigendecomposition route only in double precision.

Algorithm 5. Gram-matrix based Nyström factorization, based on (Camaño et al., 2026, Alg. 2)

Input: Matrix–vector access to $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, embedding dimension k.

Output: Factors $\boldsymbol{U}, \boldsymbol{\Lambda}$ of the Nyström approximation $\widehat{\boldsymbol{A}} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top$.

Sample a random SparseStack matrix $\boldsymbol{\Omega} \in \mathbb{R}^{n\times k}$ ▷ Section B; our code uses a SASO
$\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{\Omega}$ ▷ k matvecs with $\boldsymbol{A}$
$\boldsymbol{G} = \boldsymbol{\Omega}^\top\boldsymbol{\Omega}, \boldsymbol{C} = \boldsymbol{\Omega}^\top\boldsymbol{Y}$
$\nu = \sqrt{n}\,\varepsilon_{\text{mach}}\|\boldsymbol{Y}\|_{\text{F}}$ ▷ Small shift
Factor $\boldsymbol{C} + \nu\boldsymbol{G} = \boldsymbol{R}\boldsymbol{R}^\top$ ▷ Cholesky decomposition
$\boldsymbol{S} = \boldsymbol{Y}^\top\boldsymbol{Y} + 2\nu\boldsymbol{C} + \nu^2\boldsymbol{G}$ ▷ $\boldsymbol{S} = (\boldsymbol{Y} + \nu\boldsymbol{\Omega})^\top(\boldsymbol{Y} + \nu\boldsymbol{\Omega})$
$\boldsymbol{E} = \boldsymbol{R}^{-1}\boldsymbol{S}\boldsymbol{R}^{-\top}$ ▷ Using triangular solves
Factor $\boldsymbol{E} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^\top$ ▷ Eigendecomposition
$\boldsymbol{U} = (\boldsymbol{Y} + \nu\boldsymbol{\Omega})(\boldsymbol{R}^{-\top}\boldsymbol{V}\boldsymbol{\Lambda}^{-1/2})$
$\boldsymbol{\Lambda} \leftarrow \max\{\boldsymbol{\Lambda} - \nu\mathbf{I}, 0\}$ ▷ Remove the shift
return $\boldsymbol{U}, \boldsymbol{\Lambda}$

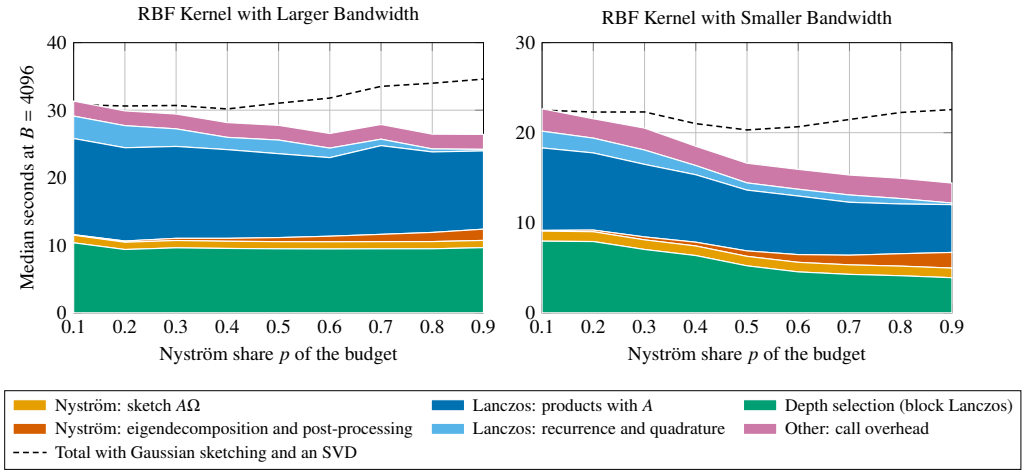


Figure 3: Algorithm performance breakdown of accelerated funNyström++. We plot the wall-clock time it took for accelerated funNyström++ to estimate the Von Neumann entropy on two kernel matrices from the Vendi Score experiments using 4,096 matvecs. The theory of accelerated funNyström++ only requires that the rank k is linearly proportional to the product ℓt. Here, we plot how the runtime changes as we adjust the fraction p of total matvecs that get allocated to Nyström instead of Girard-Hutchinson, and show how much time is spent on each subroutine as p changes. Across all runs, the methods achieved error between $\varepsilon = 2.8 \cdot 10^{-4}$ and $\varepsilon = 9.6 \cdot 10^{-4}$, meaning that little change. We see that investing more into sparse sketching can significantly reduce the total runtime. The dashed line is the total time with Gaussian sketching and an SVD in place of SASO sketching and the eigendecomposition (Section D.4). Hence, we see that increasing the rank k makes much less difference to the runtime when we have neither sparse sketching nor Gram-based Nyström eigendecomposition. One Ice Lake node with 48 threads; depth selection is charged to the budget; median time across 10 runs is shown.

Algorithm performance breakdown. In Figure 3, the products of $\boldsymbol{A}$ with the sample vectors in the Lanczos phase take the largest share of the time at every p, followed by depth selection; the Nyström approximation grows with p but stays below 20% of the total. Relative to Gaussian sketching and an SVD (the dashed line), SASO sketching and the eigendecomposition together save up to 24% of the time on the kernel with the larger bandwidth and up to 36% on the kernel with the smaller bandwidth, both at $p = 0.9$; at $p = 0.1$ the two totals are within 2% of each other. The savings grow with p because both choices speed up only the Nyström approximation, whose rank k grows with p.

D.5 Choosing the Lanczos depth adaptively

Every fixed-depth method in our experiments takes its Lanczos depth from a depth-selection run (Section D.2), and Algorithm 6 gives the procedure for block Lanczos. It runs block Lanczos on four

sample vectors and, after each step, forms the block Gauss and Gauss-Radau estimates of Section C; it stops at the first depth at which the two agree to the target accuracy. For operator monotone functions, Theorem C.3 guarantees that the Gauss estimate is then accurate to the same tolerance and that the gap between the two estimates shrinks monotonically with the depth. For single-vector Lanczos, depth selection runs the same test on each of the four sample vectors separately and takes the median of the four stopping depths, rounded up. Figure 4 shows the bracket on three matrices, including the entropy function $f(x) = -x \log x$ of the Vendi experiments, which is not operator monotone: in every case the exact value lies between the two estimates at every depth, and the relative gap between them exceeds the true error of the Gauss estimate, so stopping on the gap is conservative.

Algorithm 6. Adaptive depth selection for block Lanczos-QF

Input: Matrix–vector access to $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, function f, tolerance $\eta > 0$, maximum depth $t_{\max}$.
Output: Depth t to run block Lanczos with.

Sample $\boldsymbol{B} \in \mathbb{R}^{n\times 4}$ with columns drawn uniformly from the sphere of radius $\sqrt{n}$ ▷ Block size 4 is an arbitrary default
Initialize $\hat{\boldsymbol{Q}}_{-1} = \boldsymbol{0}$
$\hat{\boldsymbol{Q}}_0, \boldsymbol{\beta}_{-1} = \text{QR}(\boldsymbol{B})$
for $i = 0, \ldots, t_{\max} - 1$ **do**
  $\boldsymbol{Y}_{i+1} = \boldsymbol{A}\hat{\boldsymbol{Q}}_i - \hat{\boldsymbol{Q}}_{i-1}\boldsymbol{\beta}_{i-1}^\top$
  $\boldsymbol{\alpha}_i = \hat{\boldsymbol{Q}}_i^\top \boldsymbol{Y}_{i+1}$
  $\boldsymbol{Z}_{i+1} = \boldsymbol{Y}_{i+1} - \hat{\boldsymbol{Q}}_i\boldsymbol{\alpha}_i$
  Reorthogonalize $\boldsymbol{Z}_{i+1}$ against $\hat{\boldsymbol{Q}}_0, \ldots, \hat{\boldsymbol{Q}}_i$ ▷ Block modified Gram-Schmidt
  $\hat{\boldsymbol{Q}}_{i+1}, \boldsymbol{\beta}_i = \text{QR}(\boldsymbol{Z}_{i+1})$
  Construct the Gauss estimate $\boldsymbol{L}_{i+1}$ ▷ Algorithm 3
  Construct the Gauss-Radau estimate $\tilde{\boldsymbol{L}}_{i+1}$ ▷ Algorithm 4
  if $|\operatorname{tr}(\boldsymbol{L}_{i+1}) - \operatorname{tr}(\tilde{\boldsymbol{L}}_{i+1})| \le \eta \max\{|\operatorname{tr}(\boldsymbol{L}_{i+1})|, |\operatorname{tr}(\tilde{\boldsymbol{L}}_{i+1})|\}$ **then** ▷ η is the target accuracy
    return $t = i + 1$ ▷ Converged at depth $i + 1$
  end if
end for
return BUDGET EXCEEDED

The self-tuned variant of accelerated funNyström++ (Algorithm 7), which needs only a matvec budget, builds on the same depth selection: it chooses the Lanczos depth this way, charges this cost to the budget, and splits the remaining budget evenly between the Nyström approximation and the probes. RandLAPACK implements it with the single-vector form of depth selection described above.

Algorithm 7. Self-tuned accelerated funNyström++

Input: Matrix–vector access to $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, function f, matvec budget m.
Output: Approximation to $\operatorname{tr}(f(\boldsymbol{A}))$

Run Algorithm 6 with $t_{\max} = \min\{n, \lfloor m/32 \rfloor\}$; let t be the depth it returns, or $t_{\max}$ if it returns BUDGET EXCEEDED ▷ $4t \le m/8$ matvecs
$m \leftarrow m - 4t$
if Algorithm 6 returned BUDGET EXCEEDED **then**
  $t \leftarrow \min\{t, \lfloor m/8 \rfloor\}$ ▷ Keeps at least four sample vectors
end if
$\ell = \lfloor m/2t \rfloor$, $k = m - \ell t$ ▷ Half of the budget for each phase
if $k > n/2$ **then**
  $k \leftarrow \lfloor n/2 \rfloor$, $\ell \leftarrow \lfloor (m-k)/t \rfloor$ ▷ Rank cap; the surplus buys sample vectors
end if
Let $\widehat{\operatorname{tr}}_{\text{fN++}}$ be the result of Algorithm 1 with parameters k, ℓ, and t.
return $\widehat{\operatorname{tr}}_{\text{fN++}}$

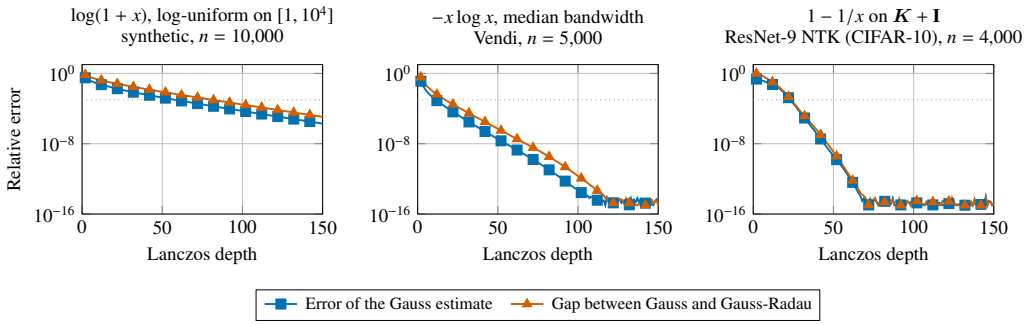


Figure 4: The Gauss-Radau gap bounds the error of the Gauss estimate. For one block Lanczos run with four sample vectors $\boldsymbol{B}$ per panel, we plot against the depth t the relative error $|\operatorname{tr}(\boldsymbol{L}_t) - \operatorname{tr}(\boldsymbol{B}^\top f(\boldsymbol{A})\boldsymbol{B})|/\operatorname{tr}(\boldsymbol{B}^\top f(\boldsymbol{A})\boldsymbol{B})$ of the block Gauss estimate, with the exact value computed from an eigendecomposition of $\boldsymbol{A}$, and the relative gap $|\operatorname{tr}(\boldsymbol{L}_t) - \operatorname{tr}(\tilde{\boldsymbol{L}}_t)|/|\operatorname{tr}(\boldsymbol{L}_t)|$ between the Gauss and Gauss-Radau estimates, which depth selection tests; the dotted line marks the target accuracy 10^{-3}. Left: $\log(1+x)$ on a diagonal matrix with $n = 10{,}000$ and eigenvalues log-uniform on $[1, 10^4]$. Middle: $-x\log x$ on the Vendi kernel matrix with $n = 5{,}000$ and the median bandwidth. Right: $1 - 1/x$ on $\boldsymbol{K} + \mathbf{I}$, where $\boldsymbol{K}$ is the leading $4{,}000 \times 4{,}000$ block of the empirical NTK of a ResNet-9 (He et al., 2016) trained on CIFAR-10 (Krizhevsky, 2009), from Ameli et al. (2025). For all three of these increasing positive concave functions, even though the second and third are not operator monotone, we see that the Gauss-Radau quadrature systematically overestimate the Gauss quadrature error, showing that the adaptive stopping criteria is reliable.

D.6 Block versus single-vector Lanczos

In RandLAPACK, the Lanczos phase of our method can run block Lanczos, as in all our other figures, or single-vector Lanczos on each sample vector; Figure 5 compares the two at equal matvec budgets. Block Lanczos is more accurate at every budget on the separated spectrum and the three real matrices, by up to 3.2 times, and within a factor of 1.7 either way on the log-uniform spectra. It reaches the target accuracy at a smaller depth, so the same budget buys more sample vectors (depth 63 against 117 on the Vendi matrix with $n = 50{,}000$ and the larger bandwidth), and it applies $\boldsymbol{A}$ to all sample vectors at once, which makes each matvec cheaper (*Fair baselines* in Section D.3).

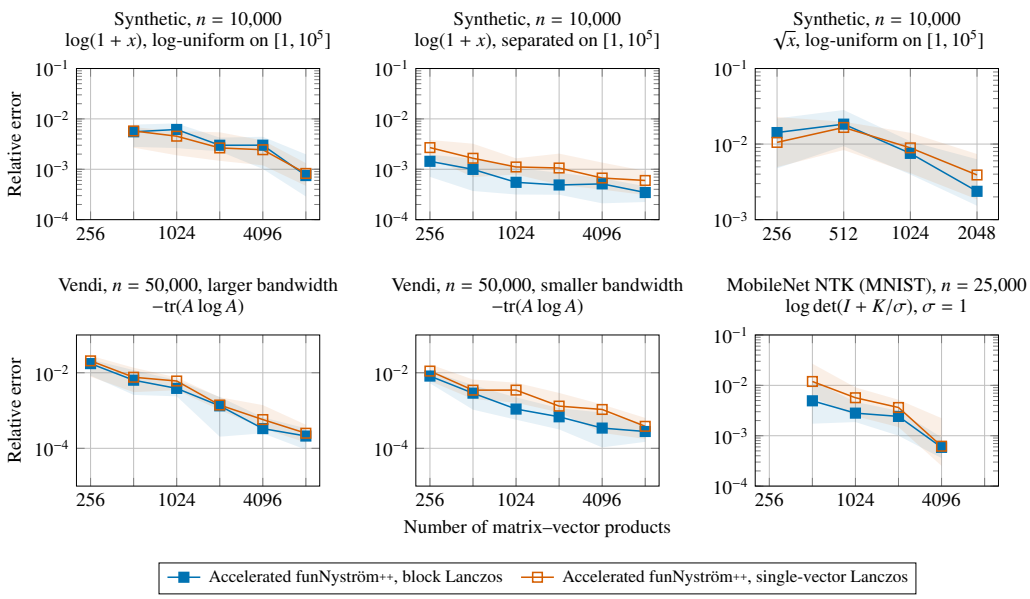


Figure 5: Block versus single-vector Lanczos in accelerated funNyström++. Median over 20 trials (lines) and 25th to 75th percentiles (shaded) of the relative error $|\widehat{\mathrm{tr}}_{\mathrm{fN}++} - \mathrm{tr}(f(\boldsymbol{A}))|/\,\mathrm{tr}(f(\boldsymbol{A}))$ against the matvec budget, with the Lanczos phase run by block Lanczos or by single-vector Lanczos, each at the depth chosen by its own depth selection. Top: diagonal matrices with $n = 10{,}000$ from Figure 6. Bottom: the Vendi kernel matrices with $n = 50{,}000$ and the MobileNet NTK on MNIST from Ameli et al. (2025) ($n = 25{,}000$, $\sigma = 1$). While the improvement is not always massive,

D.7 Synthetic spectra

Figure 6 compares all methods on diagonal matrices with prescribed spectra, which isolate the effect of the spectrum on the number of matvecs a method needs. The rows vary the placement of the eigenvalues (log-uniform on $[1, \kappa]$ or on $[1/\kappa, 1]$, or a few large eigenvalues separated from the rest) and the function, and the columns vary the condition number κ from 10^3 to 10^7.

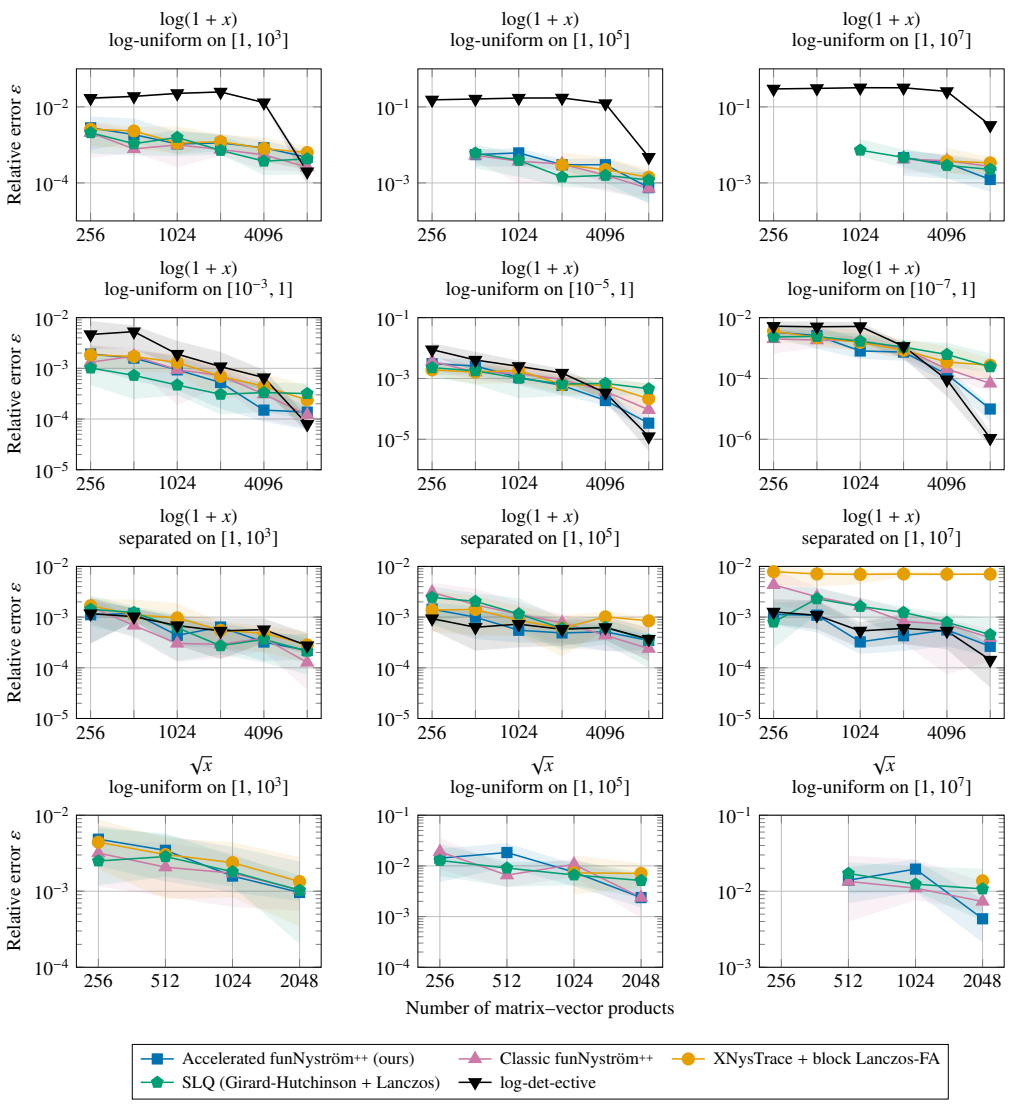


Figure 6: Matvec complexity on synthetic spectra. Median relative error over 20 trials (lines) and 25th to 75th percentiles (shaded) against the matvec budget, on diagonal matrices with $n = 10{,}000$. First row: $\mathrm{tr}(\log(\mathbf{I} + \boldsymbol{A}))$ with eigenvalues log-uniform on $[1, \kappa]$, for $\kappa = 10^3, 10^5, 10^7$ from left to right. Second row: the same, with eigenvalues log-uniform on $[1/\kappa, 1]$. Third row: the same function on a separated spectrum, with 40 eigenvalues log-uniform on $[\kappa/10, \kappa]$ and the others log-uniform on $[1, 10]$. Fourth row: $\mathrm{tr}(\boldsymbol{A}^{1/2})$ with eigenvalues log-uniform on $[1, \kappa]$. A missing point is a budget too small for the depth chosen by the method's depth selection.

D.8 Wall-clock comparisons

Figure 7 plots error against wall-clock time on the real matrices used in our experiments, following the protocol in Section D.2. Each marker is one matvec budget, so a curve further to the lower left reaches the same error in less time.

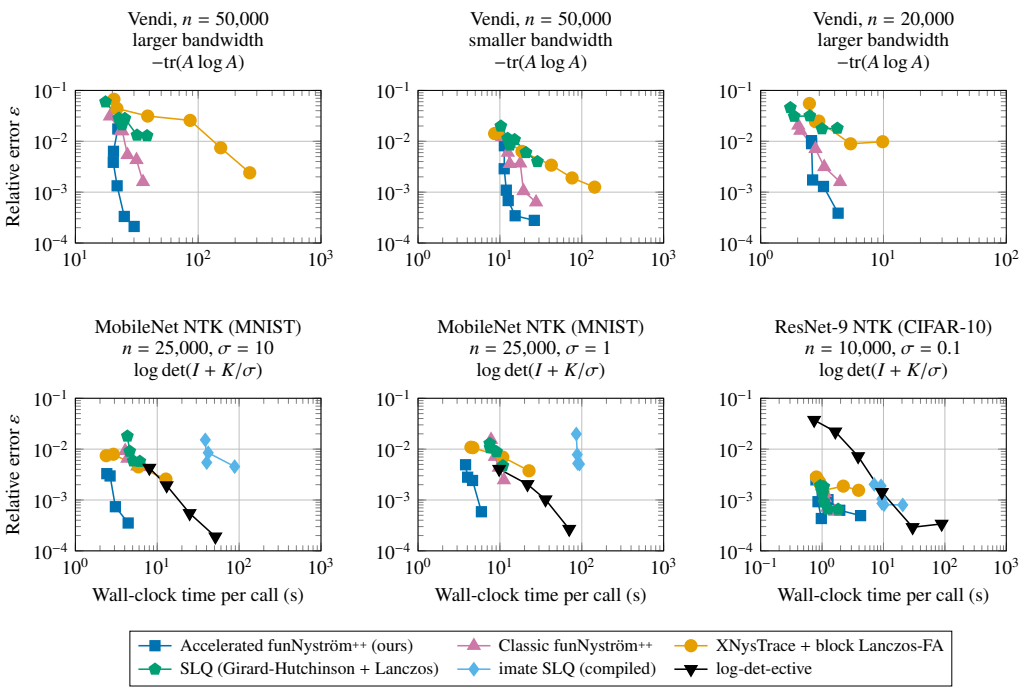


Figure 7: Wall-clock time against error on real matrices. Median relative error over 20 trials against the median wall-clock time per call over three timed trials, with one marker per matvec budget. Top: $-\mathrm{tr}(\boldsymbol{A} \log \boldsymbol{A})$ on the Vendi kernel matrices with n = 50,000 (larger and smaller bandwidth) and n = 20,000 (larger bandwidth). Bottom: $\log\det(\mathbf{I} + \boldsymbol{K}/\sigma)$ on empirical NTKs from Ameli et al. (2025): that of a MobileNet (Howard et al., 2017) trained on 2,500 MNIST images (n = 25,000; σ = 10 and σ = 1) and that of a ResNet-9 (He et al., 2016) trained on 1,000 CIFAR-10 images (n = 10,000; σ = 0.1). The timing protocol is described in Section D.2; the imate times come from imate's internal timer, which excludes loading the matrix and starting Python.