Iterative Methods Final Report

Matthew Louis, Srikanth Avasarala

December 2024

1 Introduction

Functional trace estimation of a matrix is commonly seen in many applications in fields like Machine learning, Computational biology, Quantum Chemistry, and signal processing. Given a symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ with an eigendecomposition $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top$, where $\boldsymbol{\Lambda} = \operatorname{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$ and λ_i, $i = 1, \dots, n$, are the eigenvalues of $\boldsymbol{A}$, the matrix function $f(\boldsymbol{A})$ is defined as:

$$f(\boldsymbol{A}) = \boldsymbol{U} f(\boldsymbol{\Lambda}) \boldsymbol{U}^\top,$$

where $f(\boldsymbol{\Lambda}) = \operatorname{diag}(f(\lambda_1), f(\lambda_2), \dots, f(\lambda_n))$. The trace estimation problem can then be formulated as follows: given a symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, compute an approximation of the trace of the matrix function $f(\boldsymbol{A})$, i.e.,

$$\operatorname{tr}(f(\boldsymbol{A})) = \sum_{i=1}^{n} f(\lambda_i),$$

where λ_i, $i = 1, \dots, n$, are the eigenvalues of $\boldsymbol{A}$, and f is the desired function. A naive approach to estimate the trace of matrix functions is to compute this trace directly from the eigenvalues of the matrix, which is expensive.

A variety of methods have been developed to address trace estimation problems. The stochastic trace estimator has been applied in various contexts, such as diagonal estimation, eigenvalue counting, log-determinant computation, and nuclear norm approximation[1, 6, 13]. For log-determinants, techniques like Chebyshev[11] and Taylor series expansions[16], as well as rational approximations using the Cauchy integral formula, have been used, although these can be computationally expensive due to requirements like solving multiple linear systems.

The Lanczos algorithm offers a compelling alternative for trace estimation of matrix functions, with advantages such as faster convergence compared to polynomial (e.g., Chebyshev or Taylor) and rational approximation methods. Unlike other methods, Lanczos does not require prior knowledge of the matrix spectrum and provides a matrix-dependent quadrature interpretation, leading to better convergence. However, it has practical drawbacks, such as higher storage requirements and reorthogonalization costs[10]. While polynomial methods are more storage-efficient and allow for easier a posteriori error estimates, they may converge slowly, particularly for functions with steep derivatives or discontinuities. Numerical experiments demonstrate the superior performance of the Lanczos method over other approaches, making it a robust choice for trace estimation problems.

In this project, we extend the Stochastic Lanczos Quadrature (SLQ) method, a widely used approach for functional trace estimation [15], to a block-based formulation aimed at improving the estimator's performance through preconditioning. We conduct an exploratory study using the large sparse matrix bcsstk09 from the SuiteSparse library[4] to evaluate the impact of blocking on the SLQ method and its variant that incorporates the Hutch++ estimator for trace estimation. The results of our analysis provide insights into the effects of blocking on the accuracy and computational efficiency of these methods, and we outline potential directions for further research.

1.1 Stochastic Trace Estimators

1.1.1 The Hutchinson Estimator

Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and consider a set of iid isotropic random vectors $\{\boldsymbol{\omega}_i\}_{i=1}^{s} \sim \textsc{uniform}\{\pm 1\}^n$, then the functional trace estimate of $\boldsymbol{A}$ for a given f is stated as

$$\hat{\operatorname{tr}}_H\left(f(\boldsymbol{A})\right) = \frac{1}{s}\sum_{i=1}^{s} \boldsymbol{\omega}_i^\top f(\boldsymbol{A})\boldsymbol{\omega}_i \tag{1}$$

1.1.2 The Hutch++ Estimator

One variance reduced estimator, proposed by Meyer [12] is the Hutch++ estimator. If assume s is even and let $\boldsymbol{\Omega}_1$ be the matrix whose columns are the first $s/2$ samples, and $\boldsymbol{\Omega}_2$ the matrix formed by the remaining $s/2$ samples, then the Hutch++ estimator is defined as follows

$$\begin{gathered} \boldsymbol{Y} := \boldsymbol{A}\boldsymbol{\Omega}_1 \qquad \boldsymbol{Q} := \operatorname{Orth}(\boldsymbol{Y}) \\ \hat{\operatorname{tr}}_{H++}\left(f(\boldsymbol{A})\right) = \operatorname{tr}\left(\boldsymbol{Q}^\top f(\boldsymbol{A})\boldsymbol{Q}\right) + \frac{1}{s/2}\operatorname{tr}\left(\boldsymbol{\Omega}_2^\top\left(\boldsymbol{I}-\boldsymbol{Q}\boldsymbol{Q}^\top\right)f(\boldsymbol{A})\left(\boldsymbol{I}-\boldsymbol{Q}\boldsymbol{Q}^\top\right)\boldsymbol{\Omega}_2\right) \end{gathered} \tag{2}$$

where the first term in $\hat{\operatorname{tr}}_i$ corresponds to a low rank approximation of $\boldsymbol{A}$ formed with all samples but the ith, and the second term to the trace of the residual, which is estimated by the Hutchinson estimator with the remaining samples $\boldsymbol{\Omega}_2$.

1.1.3 A Blocked Hutchinson Estimator

If we block the samples $\boldsymbol{\omega}_i \in \mathbb{R}^{n\times b}$ into blocks of size b, represented by the blocks $\boldsymbol{\Omega}_i$, a blocked variant of the Hutchinson estimator may be written as

$$\hat{\operatorname{tr}}_H(f(\boldsymbol{A})) = \frac{1}{s}\sum_{i=1}^{s/b}\operatorname{tr}\left(\boldsymbol{\Omega}_i^\top f(\boldsymbol{A})\boldsymbol{\Omega}_i\right) \tag{3}$$

2 Block Krylov Subspace Methods for Matrix Functions

To develop a blocked variant of the Hutchinson estimator, it is necessary to compute *block* matrix vector products with $f(\boldsymbol{A})$, i.e. quantities of the form $f(\boldsymbol{A})\boldsymbol{B}$ for $\boldsymbol{B} \in \mathbb{R}^{n\times b}$.

2.1 Block Arnoldi

Consider a block generalization of the Arnoldi process, where we compute an orthogonal basis for the *block* Krylov subspace $\mathcal{K}_k(\boldsymbol{A},\boldsymbol{B}) = \operatorname{span}\left(\{\boldsymbol{B}, \boldsymbol{A}\boldsymbol{B}, \cdots, \boldsymbol{A}^{k-1}\boldsymbol{B}\}\right)$ where we note this block Krylov subspace is (assuming no early breakdown) of dimension kb. First we orthogonalize the starting block $\boldsymbol{B}$ via a QR factorization $\boldsymbol{B} = \boldsymbol{Q}\boldsymbol{R}$. Then, if we let $\boldsymbol{\mathcal{Q}}_k \in \mathbb{R}^{n\times kb}$ be the matrix which contains the orthogonal blocks that form the basis for $\mathcal{K}_k(\boldsymbol{A},\boldsymbol{B})$, i.e. $\boldsymbol{\mathcal{Q}}_k = [\boldsymbol{Q}_1, \cdots, \boldsymbol{Q}_k]$, where $\boldsymbol{Q}_1 = \boldsymbol{Q}$, and we let

$$\boldsymbol{\mathcal{Q}}_k^\top \boldsymbol{A}\boldsymbol{\mathcal{Q}}_k = \boldsymbol{\mathcal{H}}_k \in \mathbb{R}^{kb\times kb}$$

where $\boldsymbol{\mathcal{H}}_k$ is the block upper Hessenberg matrix obtained by the Arnoldi process, i.e.

$$\boldsymbol{\mathcal{H}}_k = \begin{pmatrix} \boldsymbol{H}_{11} & \boldsymbol{H}_{12} & \cdots & \boldsymbol{H}_{1,k} \\ \boldsymbol{H}_{21} & \boldsymbol{H}_{22} & \cdots & \boldsymbol{H}_{2,k} \\ & \ddots & \ddots & \vdots \\ & & \boldsymbol{H}_{k,k-1} & \boldsymbol{H}_{k,k} \end{pmatrix}$$

Then, we also have the generalization of the Arnoldi recurrence for computing $\boldsymbol{\mathcal{H}}_k$

$$\boldsymbol{A}\boldsymbol{\mathcal{Q}}_k = \boldsymbol{\mathcal{Q}}_k\boldsymbol{\mathcal{H}}_k + \boldsymbol{Q}_{k+1}\boldsymbol{H}_{k+1,k}\hat{\boldsymbol{E}}_k^\top$$

where $\hat{\boldsymbol{E}}_m$ is a "block unit vector", and extracts the columns of a matrix corresponding to the kth block, i.e.

$$\hat{\boldsymbol{E}}_m = \hat{\mathbf{e}}_m^k \otimes \boldsymbol{I}_b = \begin{pmatrix} \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \boldsymbol{I}_b \end{pmatrix} \in \mathbb{R}^{kb\times b}$$

where $\boldsymbol{I}_b \in \mathbb{R}^{b\times b}$ denotes the identity matrix of dimension b and $\hat{\mathbf{e}}_m^k \in \mathbb{R}^k$ denotes the mth standard basis vector in $\mathbb{R}^k$. We may then use the block Arnoldi upper Hessenburg form $\boldsymbol{\mathcal{H}}_k$ to approximate $f(\boldsymbol{A})$ via [5]

$$f(\boldsymbol{A}) \approx \mathcal{Q}_k f(\boldsymbol{\mathcal{H}}_k)\hat{\boldsymbol{E}}_1\boldsymbol{R} \tag{4}$$

and the block Arnoldi algorithm is given by

Algorithm 1 Block Arnoldi

```
$\boldsymbol{Q}_1, \boldsymbol{R} = \mathrm{qr}(\boldsymbol{B})$
for $m = 1 \cdots m_{\max}$ do
    for $i = 1 \cdots m$ do
        $\boldsymbol{H}_{i,m} = \boldsymbol{Q}_i^\top \boldsymbol{A}\boldsymbol{Q}_m$
    end for
    $\boldsymbol{Q}_{m+1}, \boldsymbol{H}_{m+1,m+1} = \mathrm{qr}\left(\boldsymbol{A}\boldsymbol{Q}_m - \boldsymbol{\mathcal{Q}}_m\boldsymbol{\mathcal{H}}_m\hat{\boldsymbol{E}}_m\right)$
end for
$\boldsymbol{\mathcal{H}}_m = \begin{pmatrix} \boldsymbol{H}_{11} & \boldsymbol{H}_{12} & \cdots & \boldsymbol{H}_{1,m_{\max}} \\ \boldsymbol{H}_{21} & \boldsymbol{H}_{22} & \cdots & \boldsymbol{H}_{2,m_{\max}} \\ & \ddots & \ddots & \vdots \\ & & \boldsymbol{H}_{m_{\max},m_{\max}-1} & \boldsymbol{H}_{m_{\max},m_{\max}} \end{pmatrix}$
return $\boldsymbol{\mathcal{H}}_m$
```

2.2 Block Lanczos

If $\boldsymbol{A}$ is symmetric, then $\boldsymbol{\mathcal{H}}_k \equiv \boldsymbol{\mathcal{T}}_k$ is *block*-tridiagonal. Note, since each of the $\boldsymbol{H}_{i+1,i}$ subdiagonal blocks are obtained as R factors from a QR factorization, and hence they are upper triangular, the matrix $\boldsymbol{\mathcal{T}}_k$ is actually a banded matrix with bandwidth b.

$$\boldsymbol{\mathcal{T}}_k = \begin{pmatrix} \boldsymbol{\mathcal{A}}_1 & \boldsymbol{\mathcal{B}}_1^\top & & \\ \boldsymbol{\mathcal{B}}_1 & \boldsymbol{\mathcal{A}}_2 & \ddots & \\ & \ddots & \ddots & \boldsymbol{\mathcal{B}}_{k-1}^\top \\ & & \boldsymbol{\mathcal{B}}_{k-1} & \boldsymbol{\mathcal{A}}_m k \end{pmatrix}$$

For these matrices, the product $\boldsymbol{\mathcal{Q}}_k\boldsymbol{\mathcal{H}}_k\hat{\boldsymbol{E}}_k \equiv \boldsymbol{\mathcal{Q}}_k\boldsymbol{\mathcal{T}}_m\hat{\boldsymbol{E}}_k$ contains mostly zeros, and may be written as

$$\begin{aligned} \boldsymbol{\mathcal{Q}}_k\boldsymbol{\mathcal{T}}_k\hat{\boldsymbol{E}}_k &= \begin{pmatrix} \boldsymbol{Q}_1 & \cdots & \boldsymbol{Q}_{k-1} & \boldsymbol{Q}_k \end{pmatrix} \begin{pmatrix} \mathbf{0} \\ \vdots \\ \boldsymbol{\mathcal{B}}_{k-1}^\top \\ \boldsymbol{\mathcal{A}}_k \end{pmatrix} \\ &= \boldsymbol{Q}_{k-1}\boldsymbol{\mathcal{B}}_{k-1}^\top + \boldsymbol{Q}_k\boldsymbol{\mathcal{A}}_k \end{aligned}$$

which gives the following three term recurrence

$$\boldsymbol{Q}_{k+1}, \boldsymbol{\mathcal{B}}_k = \mathrm{qr}\left(\boldsymbol{A}\boldsymbol{Q}_k - \boldsymbol{Q}_{k-1}\boldsymbol{\mathcal{B}}_{k-1}^\top + \boldsymbol{Q}_k\boldsymbol{\mathcal{A}}_k\right)$$

and so we have the following algorithm

Algorithm 2 Block Lanczos

```
$\boldsymbol{Q}_1, \boldsymbol{R} = \mathrm{qr}(\boldsymbol{B})$
for $k = 1 \cdots k_{\max}$ do
    $\boldsymbol{\mathcal{A}}_k = \boldsymbol{Q}_k^\top \boldsymbol{A}\boldsymbol{Q}_k$
    $\boldsymbol{Q}_{k+1}, \boldsymbol{\mathcal{B}}_k = \mathrm{qr}\left(\boldsymbol{A}\boldsymbol{Q}_k - \boldsymbol{Q}_{k-1}\boldsymbol{\mathcal{B}}_{k-1}^\top + \boldsymbol{Q}_k\boldsymbol{\mathcal{A}}_k\right)$
end for
$\boldsymbol{\mathcal{T}}_k = \begin{pmatrix} \boldsymbol{\mathcal{A}}_1 & \boldsymbol{\mathcal{B}}_1^\top & & \\ \boldsymbol{\mathcal{B}}_1 & \boldsymbol{\mathcal{A}}_2 & \ddots & \\ & \ddots & \ddots & \boldsymbol{\mathcal{B}}_{k_{\max}-1}^\top \\ & & \boldsymbol{\mathcal{B}}_{k_{\max}-1} & \boldsymbol{\mathcal{A}}_{k_{\max}} \end{pmatrix}$
return $\boldsymbol{\mathcal{T}}_k$
```

where we note in the above algorithm $\boldsymbol{Q}_{-1}, \boldsymbol{\mathcal{B}}_{k-1} \equiv \mathbf{0}$ (formally accessed at the first iteration) as notational convenience.

2.3 The Block Lanczos Quadrature

When computing matrix functions, it is often that case that $\boldsymbol{A}$ is symmetric and positive definite, as we will assume henceforth. For such $\boldsymbol{A}$ it is natural to use the Lanczos method to estimate block matrix vector products with $f(\boldsymbol{A})$. For computing our blocked Hutchinson estimator (Eq. 3), we must estimate (using Eq. 4)

$$\mathrm{tr}\left(\boldsymbol{\Omega}_i^\top f(\boldsymbol{A})\boldsymbol{\Omega}_i\right) \approx \mathrm{tr}\left(\boldsymbol{\Omega}_i^\top \boldsymbol{\mathcal{Q}}_k f(\boldsymbol{\mathcal{T}}_k)\hat{\boldsymbol{E}}_1\boldsymbol{R}\right)$$

where $\boldsymbol{\Omega}_i = \boldsymbol{\mathcal{Q}}_k\hat{\boldsymbol{E}}_1\boldsymbol{R}$ and

$$\begin{aligned} \mathrm{tr}\left(\boldsymbol{\Omega}_i^\top \boldsymbol{\mathcal{Q}}_k f(\boldsymbol{\mathcal{T}}_k)\hat{\boldsymbol{E}}_1\boldsymbol{R}\right) &= \mathrm{tr}\left(\left(\boldsymbol{R}^\top\hat{\boldsymbol{E}}_1^\top\boldsymbol{\mathcal{Q}}_k^\top\right)\boldsymbol{\mathcal{Q}}_k f(\boldsymbol{\mathcal{T}}_k)\hat{\boldsymbol{E}}_1\boldsymbol{R}\right) \\ &= \mathrm{tr}\left(\boldsymbol{R}^\top\hat{\boldsymbol{E}}_1^\top f(\boldsymbol{\mathcal{T}}_k)\hat{\boldsymbol{E}}_1\boldsymbol{R}\right) \end{aligned}$$

If we assume that $f(\boldsymbol{\mathcal{T}}_k)$ is defined whenever $f(\boldsymbol{A})$ is (the conditions for which are rigorously proved in [9]), we may diagonalize $\boldsymbol{\mathcal{T}}_k$ (which is symmetric by construction) via $\boldsymbol{\mathcal{T}}_k = \boldsymbol{U}\boldsymbol{\Theta}\boldsymbol{U}^\top$ so that $f(\boldsymbol{\mathcal{T}}_k) = \boldsymbol{U}f(\boldsymbol{\Theta})\boldsymbol{U}^\top$. Now let $\boldsymbol{U} = [\boldsymbol{u}_1, \cdots, \boldsymbol{u}_{kb}]$ then we have

$$\begin{aligned} \mathrm{tr}\left(\boldsymbol{R}^\top\hat{\boldsymbol{E}}_1^\top f(\boldsymbol{\mathcal{T}}_k)\hat{\boldsymbol{E}}_1\boldsymbol{R}\right) &= \mathrm{tr}\left(\boldsymbol{R}^\top\hat{\boldsymbol{E}}_1^\top\boldsymbol{U}f(\boldsymbol{\Theta})\boldsymbol{U}^\top\hat{\boldsymbol{E}}_1\boldsymbol{R}\right) \\ &= \mathrm{tr}\left(\left(\boldsymbol{R}^\top\hat{\boldsymbol{E}}_1^\top\boldsymbol{U}\right)^\top\left(\boldsymbol{R}^\top\hat{\boldsymbol{E}}_1^\top\boldsymbol{U}\right)f(\boldsymbol{\Theta})\right) \qquad \text{By the cyclic property of tr} \end{aligned}$$

which we may write (noting some properties of the Frobenius norm) column-wise as

$$\begin{aligned} \mathrm{tr}\left(\boldsymbol{\Omega}_i^\top f(\boldsymbol{A})\boldsymbol{\Omega}_i\right) &\approx \sum_{i=1}^{kb}\left\|\boldsymbol{R}^\top\hat{\boldsymbol{E}}_1\boldsymbol{u}_i\right\|^2 f(\theta_i) \\ &\equiv \sum_{i=1}^{kb} w_i f(\theta_i) \end{aligned} \tag{5}$$

where $\hat{\boldsymbol{E}}_1\boldsymbol{u}_i = (\boldsymbol{u}_i)_{1:b} = \boldsymbol{U}_{1:b,i}$ and we note that our functional trace approximation takes the form of a *quadrature* (an elegant classical result). It turns out that the Lanczos iteration gives the quadrature weights and points for approximating some integral that gives exactly $\operatorname{tr}\left(\boldsymbol{\Omega}_i^\top f(\boldsymbol{A})\boldsymbol{\Omega}_i\right)$. And we note, for a unit block size $b = 1$, we have

$$\begin{aligned}\operatorname{tr}\left(\boldsymbol{\omega}_i^\top f(\boldsymbol{A})\boldsymbol{\omega}_i\right) = \boldsymbol{\omega}_i^\top f(\boldsymbol{A})\boldsymbol{\omega}_i &\approx \sum_{i=1}^{k} \left\| \|\boldsymbol{\omega}_i\| \mathbf{e}_1^\top \boldsymbol{u}_i \right\|^2 f(\theta_i) \\ &= \|\boldsymbol{\omega}_i\|^2 \sum_{i=1}^{m} |U_{1,i}|^2 f(\theta_i)\end{aligned}$$

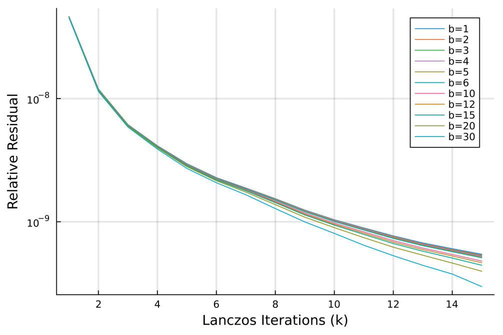
which is the classical stochastic Lanczos quadrature given in most treatments [14].

3 Numerical Results

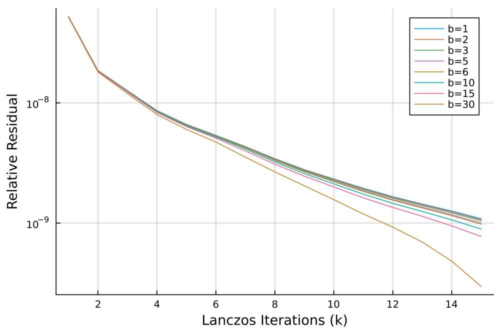
To benchmark the performance of the blocked Stochastic Lanczos iteration for functional trace estimation, we chose two large sparse spd benchmark matrices from suite sparse [5] (`bcsstk09`), and estimated the functional trace $\operatorname{tr}(f(\boldsymbol{A}))$ with $f(x) \coloneqq \ln(1+x) + x$. The code for this project is available here.

3.1 Block-Dependence of Convergence

First, to demonstrate the preconditioning abilities of the blocked Hutchinson and Hutch++ estimators, the convergence to the Hutchinson estimate $\hat{\operatorname{tr}}_H$ for a given set of samples as a function of the number of Lanczos iterates was computed for the `bcsstk09` test case. The results are shown in Fig. 1. As we might expect from a blocked algorithm, it converges faster with larger block sizes for the same number of Lanczos iterates. And, as will become the trend, the Hutch++ estimates trend similarly, which we would expect since the stochastic component of the Hutch++ estimator is given computed with the Hutchinson estimator.



(a) Hutchinson Estimator



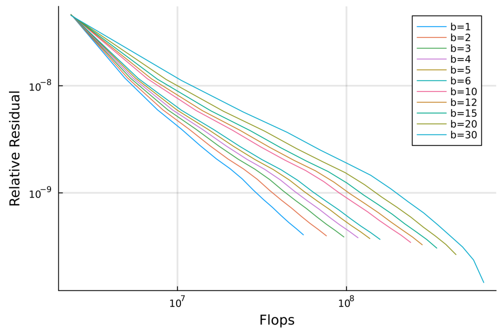
(b) Huch++ Estimator

Figure 1: Block-dependence of estimator convergence with respect to Lanczos iterations ($s = 60$)

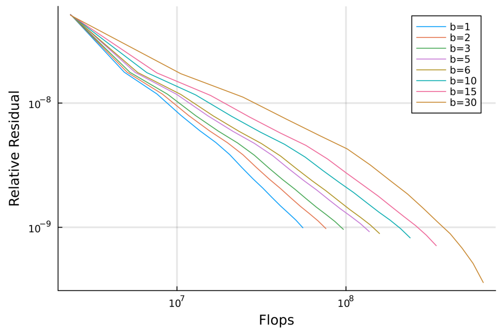
Although Fig. 1 is clean, it is not a fair comparison, because in practice, the Lanczos iteration requires some reorthogonalization scheme to maintain stability, which introduces a term that depends on the square of the block size into the computational complexity, so the algorithm scales as $\mathcal{O}(kb\operatorname{nnz}(\boldsymbol{A}) + b^2k^2n)$ (for each estimation of $\operatorname{tr}(\boldsymbol{\Omega}_i^\top f(\boldsymbol{A})\boldsymbol{\Omega}_i)$) and hence requires more flops for larger block sizes. To determine the convergence of the estimate with respect to the number of flops, we estimated the number of flops for the Lanczos iteration as

$$\hat{F} = \frac{s}{b}\left(2k\operatorname{nnz}(\boldsymbol{A})b + \frac{3}{2}b^2nk(k-1) + 2kbn\right)$$

where we have assumed *full* reorthogonalization. Using this estimated number of flops, we get the results in Fig. 2, which suggest that there are no benefits to blocking (in terms of computational complexity), and nonblocked methods are more efficient per flop.



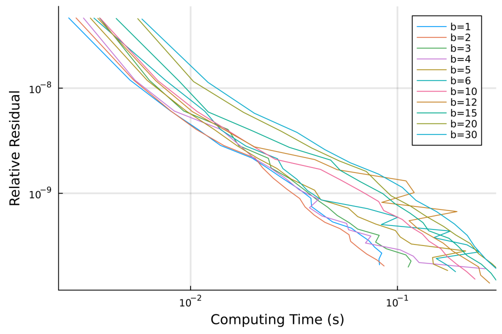
(a) Hutchinson Estimator



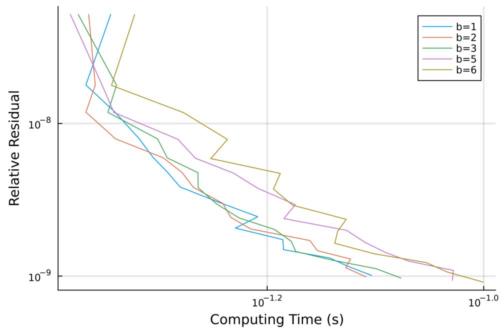
(b) Huch++ Estimator

Figure 2: Block-dependence of estimator convergence with respect to flops

However, blocked methods are often chosen over nonblocked methods because they enable the use of BLAS-3 [7], which is more cache efficient in many cases, and can lead to a significant speedup. One would then hope that this speedup might be enough to offset the increased number of flops. To investigate this, compute time was estimated using BenchmarkTools.jl [3], and the results are shown in Fig. 3. While the stratification is not as Stark as in Fig. 2, larger block sizes tend to require larger compute time for both the Hutchinson and Hutch++ estimators.



(a) Hutchinson Estimator



(b) Huch++ Estimator

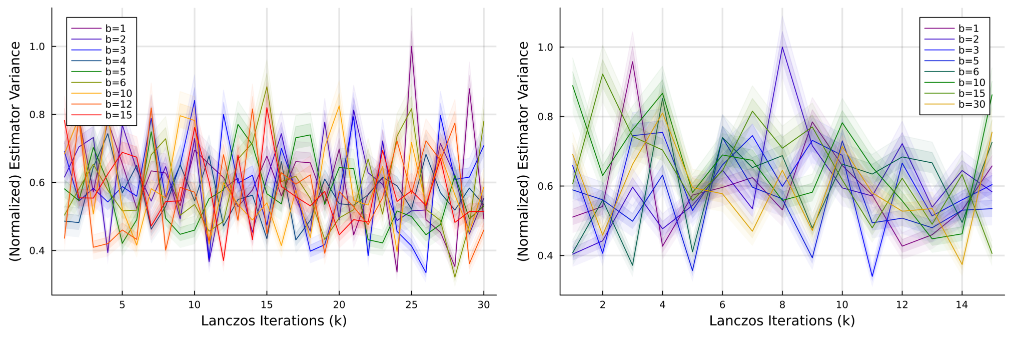
Figure 3: Block-dependence of estimator convergence with respect to compute time

3.2 Block-Dependence of Estimator Variance

For a perfectly well-converged estimate ($\hat{\text{tr}}_H$ or $\hat{\text{tr}}_{H++}$), the variance of the Hutchinson (and by extension Hutch++) estimator is known apriori, and decreases at the Monte-Carlo rate $\sim s^{-1}$ [14]. However, for a finite precision computation of $\hat{\text{tr}}_H$ by the block stochastic Lanczos quadrature, there is a potential for the blocking of samples to affect the estimator variance. In addition, it is not clear how this affect will depend on the number of samples s and the number of Lanczos iterates k. At the very least, we would like to ensure that blocking does not *increase* the estimator variance, and so the estimator variance was estimated by the sample variance S^2 over N samples whose estimated standard deviation is given (approximately) by

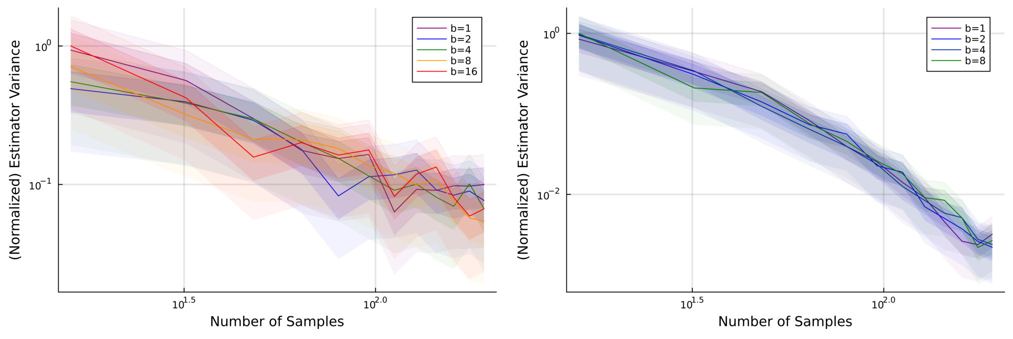
$$\hat{\sigma}_{S^2} = S^2\sqrt{\frac{2}{N-1}}$$

The results are shown in Fig. 5, 4. In all cases, the block size seems to have no stong effect either as a function of samples or Lanczos iterates, and in Fig. 5 we see the expected decay of the estimator variance with the number of samples s, and the slightly quicker decay (in this case) with the variance-reduced Hutch++ estimator.



(a) Hutchinson Estimator (b) Huch++ Estimator

Figure 4: Block-dependence of estimator variance with respect to Lanczos iterations, $s = 60$ (shaded regions indicate $\pm 1\sigma$ confidence)



(a) Hutchinson Estimator (b) Huch++ Estimator

Figure 5: Block-dependence of estimator variance with respect to the number of samples s, $k = 20$ (shaded regions indicate $\pm 1\sigma$ confidence)

4 Conclusions

We have introduced a block preconditioner for the stochastic Lanczos quadrature for functional trace estimation and empirically investigated its convergence properties and its affect on the estimator variance. For the cases considered, although there is a strong block dependence (suggesting that the block preconditioner is effective for an equal number of Lanczos iterates), the blocked algorithm converges more slowly per flop, and despite the increased cache efficiency of BLAS-3, even converges more slowly per compute time. It's not clear, however, whether there exist regimes where blocking does lead to quicker convergence (per compute time), as only the `bcsstk09` case was considered, and a more expansive study is needed yet to determine whether blocking can be an effective preconditioner for functional trace estimation.

In addition, it is not clear how blocking might affect exchangable functional trace estimators (like XTrace [8]), as blocking formally breaks exchangability. We would expect that, since the stochastic component of most exchangable estimators is computed via the Hutchinson estimator, if the Lanczos iteration is well converged, the blocking should have no effect on the estimator variance (as Figs. 5, 4 show), though confirmatory empirical calculations may still be insightful.

References

[1] Costas Bekas, Effrosyni Kokiopoulou, and Yousef Saad. An estimator for the diagonal of a matrix. *Applied numerical mathematics*, 57(11-12):1214–1229, 2007.

[2] George Casella and Roger Berger. *Statistical inference*. CRC Press, 2024.

[3] Jiahao Chen and Jarrett Revels. Robust benchmarking in noisy environments. *arXiv preprint arXiv:1608.04295*, 2016.

[4] Timothy A Davis and Yifan Hu. The university of florida sparse matrix collection. *ACM Transactions on Mathematical Software (TOMS)*, 38(1):1–25, 2011.

[5] Timothy A Davis and Yifan Hu. The university of florida sparse matrix collection. *ACM Transactions on Mathematical Software (TOMS)*, 38(1):1–25, 2011.

[6] Edoardo Di Napoli, Eric Polizzi, and Yousef Saad. Efficient estimation of eigenvalue counts in an interval. *Numerical Linear Algebra with Applications*, 23(4):674–692, 2016.

[7] J. J. Dongarra, Jeremy Du Croz, Sven Hammarling, and I. S. Duff. A set of level 3 basic linear algebra subprograms. *ACM Trans. Math. Softw.*, 16(1):1–17, March 1990.

[8] 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.

[9] Andreas Frommer, Kathryn Lund, and Daniel B Szyld. Block krylov subspace methods for functions of matrices. *Electronic Transactions on Numerical Analysis*, 47:100–126, 2017.

[10] Gene H Golub and Gérard Meurant. *Matrices, moments and quadrature with applications*, volume 30. Princeton University Press, 2009.

[11] Insu Han, Dmitry Malioutov, and Jinwoo Shin. Large-scale log-determinant computation through stochastic chebyshev expansions. In *International Conference on Machine Learning*, pages 908–917. PMLR, 2015.

[12] Raphael A Meyer, Cameron Musco, Christopher Musco, and David P Woodruff. Hutch++: Optimal stochastic trace estimation. In *Symposium on Simplicity in Algorithms (SOSA)*, pages 142–155. SIAM, 2021.

[13] Michael L Stein, Jie Chen, and Mihai Anitescu. Stochastic approximation of score functions for gaussian processes. 2013.

[14] Joel A Tropp. Randomized algorithms for matrix computations, 2020.

[15] 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.

[16] Yunong Zhang and William E Leithead. Approximate implementation of the logarithm of the matrix determinant in gaussian process regression. *Journal of Statistical Computation and Simulation*, 77(4):329–348, 2007.