

Improved variants of the Hutch++ algorithm for trace estimation*

David Persson* Alice Cortinovis* Daniel Kressner*

May 9, 2022

Abstract

This paper is concerned with two improved variants of the Hutch++ algorithm for estimating the trace of a square matrix, implicitly given through matrix-vector products. Hutch++ combines randomized low-rank approximation in a first phase with stochastic trace estimation in a second phase. In turn, Hutch++ only requires $O(\varepsilon^{-1})$ matrix-vector products to approximate the trace within a relative error ε with high probability, provided that the matrix is symmetric positive semidefinite. This compares favorably with the $O(\varepsilon^{-2})$ matrix-vector products needed when using stochastic trace estimation alone. In Hutch++, the number of matrix-vector products is fixed a priori and distributed in a prescribed fashion among the two phases. In this work, we derive an adaptive variant of Hutch++, which outputs an estimate of the trace that is within some prescribed error tolerance with a controllable failure probability, while splitting the matrix-vector products in a near-optimal way among the two phases. For the special case of a symmetric positive semi-definite matrix, we present another variant of Hutch++, called Nyström++, which utilizes the so called Nyström approximation and requires only one pass over the matrix, as compared to two passes with Hutch++. We extend the analysis of Hutch++ to Nyström++. Numerical experiments demonstrate the effectiveness of our two new algorithms.

1 Introduction

Computing or estimating the trace of a large symmetric matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$,

$$\text{tr}(\mathbf{A}) := \sum_{i=1}^n \mathbf{A}_{ii},$$

*This work has been supported by the SNSF research project *Fast algorithms from low-rank updates*, grant number: 200020_178806. Institute of Mathematics, EPF Lausanne, 1015 Lausanne, Switzerland. E-mails: david.persson@epfl.ch, alice.cortinovis@epfl.ch, daniel.kressner@epfl.ch

is an important problem that arises in a wide variety of applications, such as triangle counting in graphs [2], Frobenius norm estimation [5, 14], quantum chromodynamics [29], computing the Estrada index of a graph [8, 9], computing the log-determinant [1, 6, 27, 33] and many more. For an excellent overview of the applications to this problem we refer to [31].

It can be surprisingly difficult to compute the trace. This difficulty arises if one does not have direct access to the entries of \mathbf{A} , but can only access \mathbf{A} through matrix-vector products. This appears when, for example, \mathbf{A} is a function of another matrix \mathbf{B} , such as $\mathbf{A} = \exp(\mathbf{B})$, $\mathbf{A} = \log(\lambda\mathbf{I} + \mathbf{B})$, $\mathbf{A} = \mathbf{B}^{-1}$ or $\mathbf{A} = \mathbf{B}^3$. Computing \mathbf{A} (or even only its diagonal entries) explicitly in these situations is typically too expensive and may require up to $O(n^3)$ operations. On the other hand, computing (approximate) matrix-vector products $\mathbf{A}\mathbf{x}$ is tractable using, for example, Lanczos methods [16, 17].

Hutchinson's method [18] for trace estimation builds on the following observation: If \mathbf{x} is a random vector of length n satisfying $\mathbb{E}\mathbf{x}\mathbf{x}^T = \mathbf{I}$ then

$$\mathbb{E}\mathbf{x}^T \mathbf{A} \mathbf{x} = \text{tr}(\mathbf{A}).$$

Therefore, sampling m such quadratic forms and computing the sample mean yields the following unbiased estimator of the trace:

$$\text{tr}_m(\mathbf{A}) := \frac{1}{m} \sum_{i=1}^m \mathbf{x}_i^T \mathbf{A} \mathbf{x}_i = \frac{1}{m} \text{tr}(\mathbf{X}^T \mathbf{A} \mathbf{X}) \approx \text{tr}(\mathbf{A}), \quad (1)$$

where $\mathbf{X} = [\mathbf{x}_1 \ \cdots \ \mathbf{x}_m]$ contains m independent copies of \mathbf{x} . Common choices for the random vector \mathbf{x} are standard Gaussians; the entries in \mathbf{x} are independent identically distributed (i.i.d.) samples from $N(0, 1)$, and Rademacher vectors; the entries in \mathbf{x} are independently chosen to be -1 or $+1$ with equal probability. In this work, we choose \mathbf{x} to be standard Gaussian. In this case, the variance of $\text{tr}_m(\mathbf{A})$ is given by

$$\text{Var}(\text{tr}_m(\mathbf{A})) = \frac{2}{m} \|\mathbf{A}\|_F^2. \quad (2)$$

Under the assumption that \mathbf{A} is symmetric positive semi-definite, one can derive bounds on m that guarantee a small relative error with high probability:

$$\mathbb{P}(|\text{tr}_m(\mathbf{A}) - \text{tr}(\mathbf{A})| \leq \varepsilon \text{tr}(\mathbf{A})) \geq 1 - \delta; \quad (3)$$

see, e.g., [3, 14, 25, 26]. When \mathbf{A} is indefinite, aiming for such a relative bound is unrealistic, as can be easily seen for a non-zero matrix \mathbf{A} with $\text{tr}(\mathbf{A}) = 0$. Instead, one aims at deriving bounds on m that guarantee a small *absolute* error:

$$\mathbb{P}(|\text{tr}_m(\mathbf{A}) - \text{tr}(\mathbf{A})| \leq \varepsilon) \geq 1 - \delta. \quad (4)$$

It is well known that the number of samples needed to attain (3) or (4) grows at a rate proportional to ε^{-2} as $\varepsilon \rightarrow 0$. To reduce the number of samples (and, in

Algorithm 1 Hutch++

input: Symmetric $\mathbf{A} \in \mathbb{R}^{n \times n}$. Number of matrix-vector products $m \in \mathbb{N}$ (multiple of 3).

output: An approximation to $\text{tr}(\mathbf{A}) : \text{tr}_m^{\text{h}++}(\mathbf{A})$.

- 1: Sample $\mathbf{\Omega} \in \mathbb{R}^{n \times \frac{m}{3}}$ with i.i.d. $N(0, 1)$ or Rademacher entries.
 - 2: Compute $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$.
 - 3: Get an orthonormal basis $\mathbf{Q} \in \mathbb{R}^{n \times \frac{m}{3}}$ for $\text{range}(\mathbf{Y})$.
 - 4: Sample $\mathbf{\Psi} \in \mathbb{R}^{n \times \frac{m}{3}}$ with i.i.d. $N(0, 1)$ or Rademacher entries.
 - 5: **return** $\text{tr}_m^{\text{h}++}(\mathbf{A}) = \text{tr}(\mathbf{Q}^T \mathbf{A} \mathbf{Q}) + \frac{3}{m} \text{tr}(\mathbf{\Psi}^T (\mathbf{I} - \mathbf{Q} \mathbf{Q}^T) \mathbf{A} (\mathbf{I} - \mathbf{Q} \mathbf{Q}^T) \mathbf{\Psi})$
-

turn, the number of matrix-vector products), different variance reduction techniques were studied [12, 23, 34]. These methods aim at finding a decomposition

$$\text{tr}(\mathbf{A}) = \text{tr}(\mathbf{A}_1) + \text{tr}(\mathbf{A}_2), \quad (5)$$

such that $\text{tr}(\mathbf{A}_1)$ can be computed explicitly and the stochastic estimator for $\text{tr}(\mathbf{A}_2)$ has reduced variance, which – in view of (2) – means that \mathbf{A}_2 has reduced Frobenius norm. Among these techniques, the Hutch++ algorithm presented in [23] guarantees an ε -relative error, as in (3), with only $O(\varepsilon^{-1})$ matrix-vector products, provided that \mathbf{A} is symmetric positive semidefinite. In Hutch++, the matrix \mathbf{A}_1 in (5) is chosen to be a low-rank approximation of \mathbf{A} obtained with the randomized SVD [15], and $\mathbf{A}_2 = \mathbf{A} - \mathbf{A}_1$. The resulting method is presented in Algorithm 1. Hutch++ consists of two phases. The first phase is concerned with obtaining a low-rank approximation $\mathbf{A} \approx \mathbf{Q} \mathbf{Q}^T \mathbf{A}$ and exploits the cyclic property of the trace: $\text{tr}(\mathbf{Q} \mathbf{Q}^T \mathbf{A}) = \text{tr}(\mathbf{Q}^T \mathbf{A} \mathbf{Q})$. It uses $\frac{2m}{3}$ matrix-vector products with \mathbf{A} : $\mathbf{A}\mathbf{\Omega}$ in line 2 of Algorithm 1 and $\mathbf{A}\mathbf{Q}$ to compute $\text{tr}(\mathbf{Q}^T \mathbf{A} \mathbf{Q})$ in line 5. The second phase is concerned with estimating $\text{tr}(\mathbf{A} - \mathbf{Q} \mathbf{Q}^T \mathbf{A}) = \text{tr}((\mathbf{I} - \mathbf{Q} \mathbf{Q}^T) \mathbf{A} (\mathbf{I} - \mathbf{Q} \mathbf{Q}^T))$ via the stochastic trace estimator (1). It uses the remaining $\frac{m}{3}$ matrix-vector products with \mathbf{A} to compute $\mathbf{A}((\mathbf{I} - \mathbf{Q} \mathbf{Q}^T) \mathbf{\Psi})$ in line 5 of Algorithm 1.

1.1 Contributions

The effectiveness of the two phases of Hutch++ depends on the singular values of \mathbf{A} . When \mathbf{A} admits an accurate low-rank approximation (e.g., when its singular values decay quickly), it would be sufficient to perform the approximation $\text{tr}(\mathbf{A}) \approx \text{tr}(\mathbf{A}_1)$, as suggested by [27] and skip the second phase of Hutch++. On the other hand, when all singular values of \mathbf{A} are nearly equal, the variance reduction achieved during the first phase of Hutch++ is insignificant and all effort should be spent on the second phase, the stochastic trace estimator (1). One can easily perceive a situation where it is preferable to spend maybe not all but most of the matrix-vector products on the stochastic trace estimator. Algorithm 1 does not recognize such situations; the number of matrix-vector products is fixed a priori and distributed in a prescribed fashion among the two phases.

Furthermore, the results in [23] are of significant theoretical importance, but since the $O(\varepsilon^{-1})$ bound comes without explicit constants it gives practitioners little indication of how many matrix-vector products to use when estimating the trace of a given matrix \mathbf{A} . One can work out the constants, for example by using results in [15] if Gaussian random vectors are used, and conclude that, for fixed failure probability δ , $m = C/\varepsilon$ matrix-vector products are sufficient to get an estimate of the trace with a relative error at most ε with high probability, where C is a constant depending only on δ . However, this bound is in some cases a significant overestimation of the number of required matrix-vector products. To see this, consider the case when \mathbf{A} has rapidly decaying singular values. In this case it would be sufficient to perform the approximation $\text{tr}(\mathbf{A}) \approx \text{tr}(\mathbf{A}_1)$, with potentially much fewer matrix-vector products than suggested by the C/ε bound. On the other hand, when all singular values of \mathbf{A} are nearly equal, the standard deviation of the stochastic trace estimator, which is proportional to $\|\mathbf{A}\|_F$, is much smaller than $\text{tr}(\mathbf{A})$. Therefore, the relative error of the estimate produced by the stochastic trace estimator with only a few matrix-vector products, potentially much fewer than suggested by the C/ε bound, will give a sufficiently accurate estimate of the trace with high probability.¹

In this work, we develop an adaptive version of Hutch++ to address the above mentioned issues. We start with developing a prototype algorithm which given a prescribed tolerance ε and failure probability δ outputs an estimate of the trace of \mathbf{A} , denoted $\text{tr}_{\text{adap}}(\mathbf{A})$, such that

$$|\text{tr}_{\text{adap}}(\mathbf{A}) - \text{tr}(\mathbf{A})| \leq \varepsilon \tag{6}$$

holds, provably, with probability at least $1 - \delta$. At the same time, our algorithm attempts to minimize the overall number of matrix-vector products by distributing them between the two phases in a near-optimal fashion. Then we modify the prototype algorithm to develop a more efficient adaptive trace estimation algorithm, which will be A-Hutch++. Note, however, that the potential for improving Hutch++ is limited, in [23] the $O(\varepsilon^{-1})$ bound mentioned above is proven to be optimal up to a $\log(\varepsilon^{-1})$ factor. In practice, we observe that our adaptive version of Hutch++ is never worse than the original Hutch++ and often outperforms it. Possibly more importantly, the output of our prototype algorithm comes with a probabilistic guarantee on the error of the estimate of $\text{tr}(\mathbf{A})$ without requiring the user to know a priori how many matrix-vector products are needed. Our algorithm does not assume that \mathbf{A} is positive definite, which is why we focus on estimating $\text{tr}(\mathbf{A})$ up to a given absolute error.

Another aspect we address in this work is that the Hutch++ algorithm requires several passes over the matrix \mathbf{A} ; in Algorithm 1 the matrix-vector products carried out in line 5 depend on earlier ones. In the streaming model it is desirable to design an algorithm that requires only one pass over \mathbf{A} and if the matrix of interest is modified by a linear update $\mathbf{A} + \mathbf{E}$ one does not have to

¹To see this, recall that the standard deviation of the stochastic trace estimator with m samples equals $\sqrt{2/m}\|\mathbf{A}\|_F$. This can be much smaller than $\varepsilon \text{tr}(\mathbf{A})$ with m potentially much smaller than C/ε , provided ε is not too small.

revisit \mathbf{A} to update the output of the algorithm. Such a single pass property also increases parallelism. A single pass trace estimation algorithm was presented in [23] and we will call it *Single Pass Hutch++* in this work. For a symmetric positive semidefinite matrix this algorithm comes with nearly the same theoretical guarantees as Hutch++, but performs worse in practice. In the case of a symmetric positive semidefinite matrix we develop a variation of Hutch++, Nyström++, utilizing the Nyström approximation [13]. Nyström++ requires only one pass over \mathbf{A} and satisfies, up to constants, the theoretical guarantees of Hutch++. This new variation of Hutch++ significantly outperforms Single Pass Hutch++ and often outperforms Hutch++.

Remark. Note that the word *adaptive* is used differently in [23], where Hutch++ itself is already called *adaptive* because the matrix-vector products $\mathbf{A}\mathbf{Q}$ depend on (and thus adapt to) the previously computed $\mathbf{A}\mathbf{\Omega}$. In this work, we follow the convention where the term *adaptive* refers to an algorithm that adapts to a desired error bound. The Single Pass Hutch++ mentioned above is called *NA-Hutch++* (non-adaptive variant of Hutch++) in [23].

1.2 Notation

For a vector $\mathbf{x} \in \mathbb{R}^n$ we let $\|\mathbf{x}\|_2 = \left(\sum_{i=1}^n x_i^2\right)^{1/2}$ denote the Euclidean norm of \mathbf{x} . We let $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n \geq 0$ denote the singular values of \mathbf{A} . Thus, we have $\|\mathbf{A}\|_2 = \sigma_1$ and $\|\mathbf{A}\|_F^2 = \sigma_1^2 + \dots + \sigma_n^2$. The nuclear norm of \mathbf{A} is defined as $\|\mathbf{A}\|_* = \sigma_1 + \dots + \sigma_n$. We let $\rho(\mathbf{A}) = \frac{\|\mathbf{A}\|_F^2}{\|\mathbf{A}\|_2^2}$ denote the stable rank of \mathbf{A} . Furthermore, for a matrix $\mathbf{B} \in \mathbb{R}^{m \times p}$, $p \geq m$, with linearly independent rows we let $\mathbf{B}^\dagger := \mathbf{B}^T(\mathbf{B}\mathbf{B}^T)^{-1}$ denote the Moore-Penrose pseudoinverse. We say that a random $n \times k$ matrix $\mathbf{\Omega}$ with i.i.d. $N(0, 1)$ entries is a *standard Gaussian matrix*. In the case of $k = 1$ we say that it is a *standard Gaussian vector*.

2 Adaptive variants of Hutch++

The aim of this section is to develop adaptive variants of Hutch++ (Algorithm 1). In a first step, we derive a prototype algorithm that aims at minimizing the number of matrix-vector products and comes with a guaranteed bound on the failure probability. The latter requires to estimate the variance or, equivalently (see (2)), the Frobenius norm, and this estimate needs additional matrix-vector products. Our final algorithm A-Hutch++ reuses these matrix-vector products for trace estimation and chooses the number of them in an adaptive fashion. In turn, this creates dependencies that complicate the analysis but do not lead to observed failure probabilities that are above the prescribed failure probability.

2.1 Derivation of adaptive Hutch++

The first phase of Algorithm 1 requires $2r$ matrix-vector products with \mathbf{A} to obtain a rank- r approximation $\mathbf{Q}^{(r)}\mathbf{Q}^{(r)T}\mathbf{A}$, where we have added a superscript to emphasize the dependence on r . Let $M(r)$ be the number of matrix-vector products with \mathbf{A} in the second phase such that the stochastic trace estimator of

$$\mathbf{A}_{\text{rest}}^{(r)} := (\mathbf{I} - \mathbf{Q}^{(r)}\mathbf{Q}^{(r)T})\mathbf{A}(\mathbf{I} - \mathbf{Q}^{(r)}\mathbf{Q}^{(r)T}) \quad (7)$$

attains a prescribed accuracy and success probability. Then the total number of matrix-vector products with \mathbf{A} is

$$m(r) = 2r + M(r). \quad (8)$$

We aim at minimizing $m(r)$ in order to obtain a near-optimal distribution of matrix-vector products between the two phases.² For this purpose, we first derive a suitable expression for $M(r)$.

2.1.1 Analysis of trace estimation

The tightest tail bound available in the literature for the stochastic trace estimator $\text{tr}_m(\mathbf{B})$ for a symmetric matrix \mathbf{B} is [6, Theorem 1], which states that

$$\mathbb{P}(|\text{tr}_m(\mathbf{B}) - \text{tr}(\mathbf{B})| \geq \varepsilon) \leq 2 \exp\left(-m \frac{\varepsilon^2}{4\|\mathbf{B}\|_F^2 + 4\varepsilon\|\mathbf{B}\|_2}\right). \quad (9)$$

In most situations of interest, the term involving $\|\mathbf{B}\|_2$ will be insignificant. The following lemma is a variation of (9) that suppresses this term for sufficiently large m , similar to [23, Lemma 2.1]. We note in passing that (9) as well as the following lemma can be improved; see Appendix A.

Lemma 2.1. *Given $\ell > 0$ assume that $m \geq \frac{4(1+\ell)\log(2/\delta)}{\ell^2\rho(\mathbf{B})}$. Then the inequality*

$$|\text{tr}_m(\mathbf{B}) - \text{tr}(\mathbf{B})| \leq 2\sqrt{1+\ell}\ell\sqrt{\frac{\log(2/\delta)}{m}}\|\mathbf{B}\|_F \quad (10)$$

holds with probability at least $1 - \delta$.

²In practice we perform randomized low-rank approximations. Consequently, $\mathbf{A}_{\text{rest}}^{(r)}$ is random and therefore the function m is a random variable. Hence, it can be ambiguous what it means to minimize m . To clarify this, first note that we always assume $r \leq n$, where \mathbf{A} is $n \times n$, since when $r = n$ we are able to exactly compute $\text{tr}(\mathbf{A})$. Therefore, we will never sample more than n random vectors to obtain a low-rank approximation. Thus, let $\mathbf{\Omega} \in \mathbb{R}^{n \times n}$ be the random matrix from which we can construct $\mathbf{Q}^{(1)}, \mathbf{Q}^{(2)}, \dots, \mathbf{Q}^{(n)}$. Conditioned on $\mathbf{\Omega}$ the function m becomes deterministic and has a minimum, which is what we aim to find. We will describe a heuristic strategy to find the minimum in Section 2.1.2.

Proof. Inserting the right-hand side of (10), $\varepsilon := 2\sqrt{1+\ell}\sqrt{\frac{\log(2/\delta)}{m}}\|\mathbf{B}\|_F$, into (9) one obtains the desired result:

$$\begin{aligned} \mathbb{P}(|\operatorname{tr}_m(\mathbf{B}) - \operatorname{tr}(\mathbf{B})| \geq \varepsilon) &\leq 2 \exp\left(-\frac{(1+\ell)\log(2/\delta)\|\mathbf{B}\|_F}{\|\mathbf{B}\|_F + 2\sqrt{1+\ell}\sqrt{\frac{\log(2/\delta)}{m}}\|\mathbf{B}\|_2}\right) \\ &\leq 2 \exp\left(-\frac{(1+\ell)\log(2/\delta)\|\mathbf{B}\|_F}{(1+\ell)\|\mathbf{B}\|_F}\right) = \delta, \end{aligned}$$

where the second inequality utilizes

$$\ell\|\mathbf{B}\|_F \geq 2\sqrt{1+\ell}\sqrt{\frac{\log(2/\delta)}{m}}\|\mathbf{B}\|_2,$$

a consequence of the assumption on m . \square

Let

$$C(\varepsilon, \delta) := 4(1+\ell)\varepsilon^{-2}\log(2/\delta). \quad (11)$$

By Lemma 2.1, for sufficiently small ε , $C(\varepsilon, \delta)\|\mathbf{B}\|_F^2$ samples are sufficient to achieve $|\operatorname{tr}_m(\mathbf{B}) - \operatorname{tr}(\mathbf{B})| \leq \varepsilon$ with probability at least $1 - \delta$. In practice one cannot assume to know, or be able to compute, the stable rank appearing in the condition $m \geq \frac{4(1+\ell)\log(2/\delta)}{\ell^2\rho(\mathbf{B})}$. Since the stable rank is always larger than 1, requiring $m \geq \frac{4(1+\ell)\log(2/\delta)}{\ell^2}$ would be sufficient to ensure that $m \geq \frac{4(1+\ell)\log(2/\delta)}{\ell^2\rho(\mathbf{B})}$. However, in practice we set $\ell = 0$ and completely omit the side condition $m \geq \frac{4(1+\ell)\log(2/\delta)}{\ell^2\rho(\mathbf{B})}$. While not justified by Lemma 2.1, we observe no significant loss in the success probabilities of our algorithm, see Section 2.3.1.

2.1.2 Finding the minimum of $m(r)$

Applying the results above to $\mathbf{B} = \mathbf{A}_{\text{rest}}^{(r)}$ implies that a suitable choice for the function $m(r)$ in (8) is given by

$$m(r) = 2r + C(\varepsilon, \delta)\|\mathbf{A}_{\text{rest}}^{(r)}\|_F^2. \quad (12)$$

In the idealistic scenario that $\mathbf{Q}^{(r)}$ contains the dominant r singular vectors, we have $\|\mathbf{A}_{\text{rest}}^{(r)}\|_F^2 = \sigma_{r+1}^2 + \dots + \sigma_n^2$. This implies that the differences $m(r) - m(r-1) = 2 - C(\varepsilon, \delta)\sigma_{r+1}^2$ are monotonically increasing and switch sign at most once. In turn, r^* is a global minimum whenever it is a local minimum, that is, $m(r^* \pm 1) \geq m(r^*)$. Since $\mathbf{Q}^{(r)}$ only approximates the space spanned by the dominant r singular vectors of \mathbf{A} , these relations are not guaranteed to hold. In practice, we have observed $m(r^* \pm 1) \geq m(r^*)$ to remain a reliable criterion; see Figure 1 for an example.

Evaluating $m(r)$ involves the quantity $\|\mathbf{A}_{\text{rest}}^{(r)}\|_F^2$, which is too expensive to evaluate. Using the symmetry of \mathbf{A} and the unitary invariance of the Frobenius norm we get

$$\|\mathbf{A}_{\text{rest}}^{(r)}\|_F^2 = \|\mathbf{A}\|_F^2 + \|\mathbf{Q}^{(r)T}\mathbf{A}\mathbf{Q}^{(r)}\|_F^2 - 2\|\mathbf{A}\mathbf{Q}^{(r)}\|_F^2. \quad (13)$$

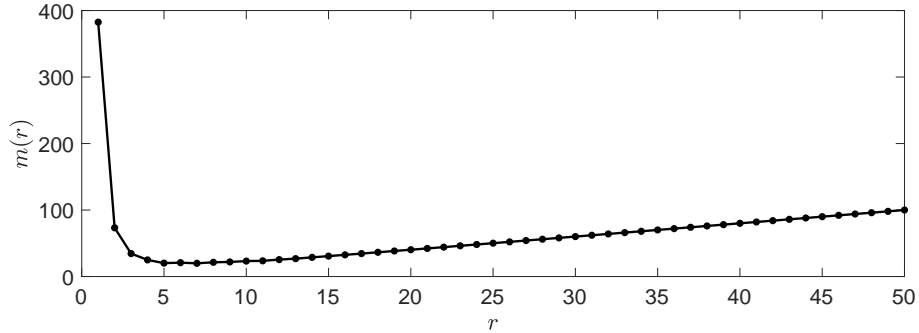


Figure 1: In this example we let $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T \in \mathbb{R}^{1000 \times 1000}$ where \mathbf{U} is a random orthogonal matrix and $\mathbf{\Lambda}$ is a diagonal matrix with $\mathbf{\Lambda}_{ii} = 1/i^2$. The x-axis shows the rank r , and the y-axis is the function $m(r)$ defined in (12) with $\delta = 0.01$, $\varepsilon = 0.05 \operatorname{tr}(\mathbf{A})$ and $\ell = 0$. The function has its minimum at $r^* = 7$.

In turn, $m(r)$ and the function

$$\tilde{m}(r) := 2r + C(\varepsilon, \delta) (\|\mathbf{Q}^{(r)T} \mathbf{A} \mathbf{Q}^{(r)}\|_F^2 - 2\|\mathbf{A} \mathbf{Q}^{(r)}\|_F^2) \quad (14)$$

have the same minimum. The latter can be cheaply computed by recursive updating, without any additional matrix-vector products with \mathbf{A} .

To summarize, we adapt the randomized SVD to build $\mathbf{Q}^{(r)}$ column-by-column, similar to as described in [15, Section 4.4], and stop the loop whenever a minimum of $\tilde{m}(r)$ is detected. By the heuristics discussed above, it is safe to stop at $r = r^*$ when $\tilde{m}(r^*) > \tilde{m}(r^* - 1) > \tilde{m}(r^* - 2)$.

2.1.3 Estimating the Frobenius norm of the remainder

Having found an approximate minimum r^* of $\tilde{m}(r)$ and computed $\mathbf{Q} \equiv \mathbf{Q}^{(r^*)}$, it remains to apply stochastic trace estimation to $\mathbf{A}_{\text{rest}} \equiv \mathbf{A}_{\text{rest}}^{(r^*)}$. By Lemma 2.1 it suffices to use $M \geq C(\varepsilon, \delta) \|\mathbf{A}_{\text{rest}}\|_F^2$ samples. Because computing $\|\mathbf{A}_{\text{rest}}\|_F$ is too expensive, we need to resort (once more) to a stochastic estimator utilizing only matrix-vector products. The following result is essential for that purpose.

Lemma 2.2. *Let $\mathbf{\Omega} \in \mathbb{R}^{n \times k}$ be a standard Gaussian matrix and let $\mathbf{B} \in \mathbb{R}^{n \times n}$. For any $\alpha \in (0, 1)$ it holds that*

$$\mathbb{P} \left(\frac{1}{k} \|\mathbf{B}\mathbf{\Omega}\|_F^2 < \alpha \|\mathbf{B}\|_F^2 \right) \leq \mathbb{P}(X < \alpha) = \frac{\gamma(k/2, \alpha k/2)}{\Gamma(k/2)},$$

where $X \sim \Gamma(k/2, k/2)$ (gamma distribution with shape and rate parameter $k/2$), $\gamma(s, x) := \int_0^x t^{s-1} e^{-t} dt$ is the lower incomplete gamma function and $\Gamma(s)$ is the standard gamma function.

Proof. It is well known that

$$\frac{1}{k} \|\mathbf{B}\mathbf{\Omega}\|_F^2 = \frac{1}{k} \sum_{j=1}^n \sigma_j^2 Z_j, \quad (15)$$

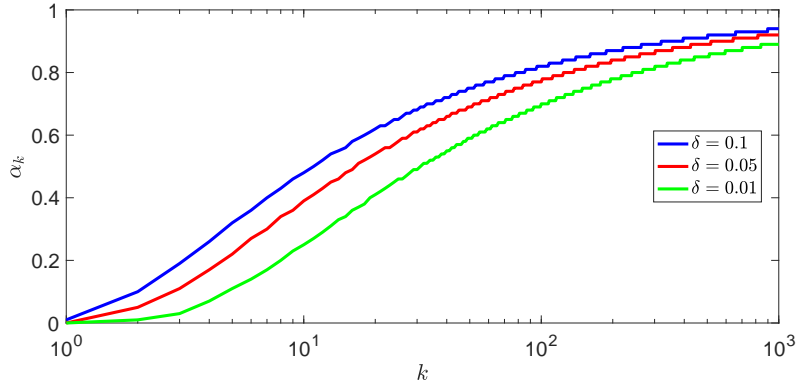


Figure 2: For different choices of δ , this plot demonstrates the relationship between k and the largest choice of α such that $\frac{\gamma(k/2, \alpha k/2)}{\Gamma(k/2)} \leq \delta$.

where Z_j , $j = 1, \dots, n$, denote i.i.d. χ_k^2 random variables; see, e.g., [14, Section 2]. Setting $X_j := \frac{1}{k} Z_j \sim \Gamma(k/2, k/2)$ and $\lambda_j = \frac{\sigma_j^2}{\|\mathbf{B}\|_F^2}$ for $j = 1, \dots, n$ we rewrite

$$\mathbb{P}\left(\frac{1}{k} \|\mathbf{B}\boldsymbol{\Omega}\|_F^2 < \alpha \|\mathbf{B}\|_F^2\right) = \mathbb{P}\left(\sum_{j=1}^n \lambda_j X_j < \alpha\right). \quad (16)$$

By [26, Theorem 2.2] the right-hand side is bounded for every $\alpha \in (0, 1)$ by $\mathbb{P}(X_1 < \alpha)$, which completes the proof. \square

Lemma 2.2 states that if $\frac{\gamma(k/2, \alpha k/2)}{\Gamma(k/2)} \leq \delta$ then $\frac{1}{k\alpha} \|\mathbf{B}\boldsymbol{\Omega}\|_F^2 > \|\mathbf{B}\|_F^2$ with probability at least $1 - \delta$. Hence, using $M := \lceil C(\varepsilon, \delta) \cdot \frac{1}{k\alpha} \|\mathbf{A}_{\text{rest}}\boldsymbol{\Omega}\|_F^2 \rceil$ samples ensures an error of at most ε with low failure probability. See Figure 2 for the relationship between k , α and δ .

2.1.4 A prototype algorithm

Combining the results presented above we obtain the prototype algorithm presented in Algorithm 2. To reduce the number of passes over the matrix \mathbf{A} the algorithm can be implemented in a block-wise fashion, which can in turn lead to a reduction of wall-clock time. For block-size $b = 1$ we use the heuristic stopping criteria for the low-rank approximation described above. For larger block-sizes it is sufficient to use $m(r^* - b) < m(r^*)$ as a stopping criteria.

Algorithm 2 Prototype algorithm

input: Symmetric $\mathbf{A} \in \mathbb{R}^{n \times n}$. Error tolerance $\varepsilon > 0$. Failure probability $\delta \in (0, 1)$. Parameter $\ell > 0$. Block-size b .

output: An approximation to $\text{tr}(\mathbf{A})$: $\text{tr}_{\text{adap}}(\mathbf{A})$.

- 1: $\mathbf{Y}^{(b)} = \mathbf{A}\mathbf{\Omega}^{(b)}$ where $\mathbf{\Omega}^{(b)} \in \mathbb{R}^{n \times b}$ has i.i.d. $N(0, 1)$ entries.
 - 2: Obtain orthonormal basis $\widehat{\mathbf{Q}}^{(b)}$ for range $(\mathbf{Y}^{(b)})$.
 - 3: $\mathbf{Q}^{(1)} = \widehat{\mathbf{Q}}^{(1)}$
 - 4: $\text{trest}_1 = \text{tr} \left(\widehat{\mathbf{Q}}^{(1)T} \left(\mathbf{A} \widehat{\mathbf{Q}}^{(1)} \right) \right)$
 - 5: Compute $\tilde{m}(b)$.
 - 6: $r = b$
 - 7: **while** A minimum of $\tilde{m}(r)$ not detected **do**
 - 8: $\mathbf{Y}^{(r+b)} = \mathbf{A}\mathbf{\Omega}^{(r+b)}$ where $\mathbf{\Omega}^{(r+b)} \in \mathbb{R}^{n \times b}$ has i.i.d. $N(0, 1)$ entries.
 - 9: $\tilde{\mathbf{Q}}^{(r+b)} = (\mathbf{I} - \mathbf{Q}^{(r)}\mathbf{Q}^{(r)T})\mathbf{Y}^{(r+b)}$
 - 10: Obtain orthonormal basis $\widehat{\mathbf{Q}}^{(r+b)}$ for range $(\tilde{\mathbf{Q}}^{(r+b)})$.
 - 11: $\mathbf{Q}^{(r+b)} = \begin{bmatrix} \mathbf{Q}^{(r)} & \widehat{\mathbf{Q}}^{(r+b)} \end{bmatrix}$
 - 12: $\text{trest}_1 = \text{trest}_1 + \text{tr} \left(\widehat{\mathbf{Q}}^{(r+b)T} \left(\mathbf{A} \widehat{\mathbf{Q}}^{(r+b)} \right) \right)$
 - 13: Update $\tilde{m}(r+b)$ recursively.
 - 14: $r = r + b$
 - 15: **end while**
 - 16: Let $\mathbf{Q} = \mathbf{Q}^{(r)}$ and $\mathbf{A}_{\text{rest}} = (\mathbf{I} - \mathbf{Q}\mathbf{Q}^T)\mathbf{A}(\mathbf{I} - \mathbf{Q}\mathbf{Q}^T)$. $\triangleright \mathbf{A}_{\text{rest}}$ is never formed explicitly.
 - 17: Choose $(k, \alpha) \in \mathbb{N} \times (0, 1)$ such that $\frac{\gamma(k/2, \alpha k/2)}{\Gamma(k/2)} \leq \delta$.
 - 18: $M = \max \left\{ \frac{4(1+\ell)\log(2/\delta)}{\ell^2}, \lceil C(\varepsilon, \delta) \cdot \frac{1}{k\alpha} \|\mathbf{A}_{\text{rest}} \mathbf{\Psi}\|_F^2 \rceil \right\}$ where $\mathbf{\Psi} \in \mathbb{R}^{n \times k}$ is a standard Gaussian matrix .
 - 19: $\text{trest}_2 = \text{tr}_M(\mathbf{A}_{\text{rest}})$
 - 20: **return** $\text{tr}_{\text{adap}}(\mathbf{A}) = \text{trest}_1 + \text{trest}_2$
-

A simple probabilistic analysis yields the following result on the success probability of Algorithm 2:

Lemma 2.3. *The output of Algorithm 2 satisfies $|\text{tr}_{\text{adap}}(\mathbf{A}) - \text{tr}(\mathbf{A})| \leq \varepsilon$ with probability at least $1 - 2\delta$.*

Proof. For the moment, let us consider \mathbf{Q} fixed and, hence, \mathbf{A}_{rest} deterministic. For a fixed arbitrary integer N let us consider the event

$$S_N := \{|\text{tr}_N(\mathbf{A}_{\text{rest}}) - \text{tr}(\mathbf{A}_{\text{rest}})| \leq \varepsilon\}.$$

Let M be the random variable defined in line 18 of Algorithm 2. Therefore, S_M is the event that the estimate of $\text{tr}(\mathbf{A}_{\text{rest}})$ from Algorithm 2 has an error at most ε . That is,

$$S_M = \{|\text{tr}_M(\mathbf{A}_{\text{rest}}) - \text{tr}(\mathbf{A}_{\text{rest}})| \leq \varepsilon\} = \bigcup_{N \geq 1} [S_N \cap \{M = N\}]$$

The analysis of $\mathbb{P}(S_M)$ is complicated by the fact that the integer M defined in line 18 of Algorithm 2 is also random. Letting

$$M_1 := \max \left\{ \frac{4(1+\ell) \log(2/\delta)}{\ell^2 \rho(\mathbf{A}_{\text{rest}})}, C(\varepsilon, \delta) \|\mathbf{A}_{\text{rest}}\|_F^2 \right\},$$

we know from Lemma 2.2 that $\mathbb{P}(M \geq M_1) \geq 1 - \delta$ and from (11) that $\mathbb{P}(S_N) \geq 1 - \delta$ for $N \geq M_1$. Moreover, it is important to remark that the events S_N and $M = N$ are independent. In particular, this implies $\mathbb{P}(S_M | M = N) = \mathbb{P}(S_N)$. Combining these observations yields

$$\begin{aligned} \mathbb{P}(S_M) &\geq \mathbb{P}(S_M \cap \{M \geq M_1\}) \\ &= \sum_{N \geq M_1} \mathbb{P}(S_M \cap \{M = N\}) = \sum_{N \geq M_1} \mathbb{P}(S_M | M = N) \mathbb{P}(M = N) \\ &= \sum_{N \geq M_1} \mathbb{P}(S_N) \mathbb{P}(M = N) \geq (1 - \delta) \sum_{N \geq M_1} \mathbb{P}(M = N) \\ &= (1 - \delta) \mathbb{P}(M \geq M_1) \geq (1 - \delta)^2 \geq 1 - 2\delta, \end{aligned}$$

which holds independently of \mathbf{Q} and thus completes the proof. \square

2.2 A-Hutch++

To turn Algorithm 2 into a practical method, we need to address the choice of the pair (k, α) in line 17 and apply further modification to increase its efficiency by reusing the matrix vector products in the Frobenius norm estimation in line 18 in the trace estimation in line 19 of Algorithm 2.

For fixed k , it makes sense to choose α as large as possible because M decreases with increasing α ; see line 18. Thus, we set

$$\alpha_k := \sup \left\{ \alpha \in (0, 1) : \frac{\gamma(k/2, \alpha k/2)}{\Gamma(k/2)} \leq \delta \right\}. \quad (17)$$

Lemma 2.4. *The sequence $\{\alpha_k\}_{k \in \mathbb{N}}$ defined by (17) increases monotonically and converges to 1.*

Proof. Letting $X := \frac{1}{k} \sum_{i=1}^k X_i \sim \Gamma(k/2, k/2)$ for i.i.d. χ_1^2 random variables X_i , we set

$$p_k(\alpha) := \mathbb{P}(X \leq \alpha) = \frac{\gamma(k/2, \alpha k/2)}{\Gamma(k/2)}.$$

By [26, Theorem 2.1] $p_{k+1}(\alpha) \leq p_k(\alpha)$ for every $\alpha \in (0, 1]$. Furthermore, by continuity of p_k in α and monotonicity of $p_k(\alpha)$ in k we have

$$\delta = p_k(\alpha_k) = p_{k+1}(\alpha_{k+1}) \leq p_k(\alpha_{k+1}).$$

Thus, by monotonicity of p_k in α we have $\alpha_k \leq \alpha_{k+1}$, which proves the monotonicity of the sequence $\{\alpha_k\}_{k \in \mathbb{N}}$.

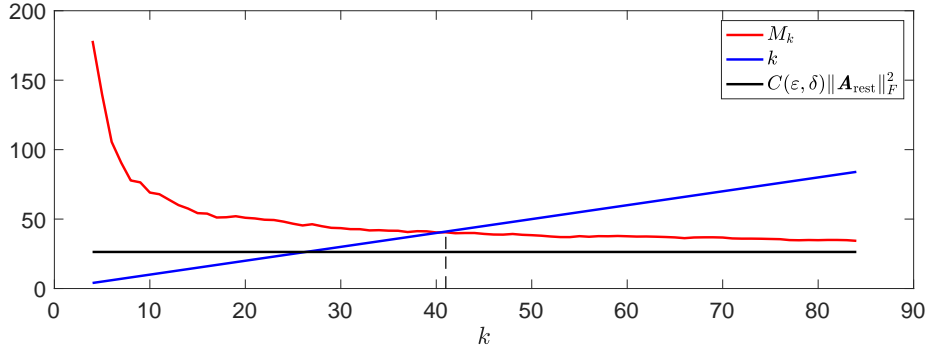


Figure 3: In this example we let $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T \in \mathbb{R}^{1000 \times 1000}$ where \mathbf{U} is a random orthogonal matrix and $\mathbf{\Lambda}$ is a diagonal matrix with $\Lambda_{ii} = 1/i^{1.5}$. We run Algorithm 2 with $\varepsilon = 0.01 \text{tr}(\mathbf{A})$, $\delta = 0.05$ and $\ell = 0$ to obtain \mathbf{A}_{rest} defined in line 16. The x-axis shows the number of matrix-vector products with \mathbf{A}_{rest} . The red line shows the evolution of the sequence M_k defined in (18), the blue line shows the linear line k against k and the black line is the number of matrix-vector products with \mathbf{A}_{rest} to guarantee an error less than ε with probability at least $1 - \delta$. We stop the while loop in Algorithm 3 once the red and blue line cross.

To show $\alpha_k \rightarrow 1$ as $k \rightarrow +\infty$, let $\alpha_\varepsilon := 1 - \varepsilon > 0$ for fixed arbitrary $0 < \varepsilon < 1$. By the law of large numbers, $p_k(\alpha_\varepsilon) \rightarrow 0$ and by the argument above this convergence is monotonic. Let $k_{\varepsilon, \delta} = \min\{k \in \mathbb{N} : p_k(\alpha_\varepsilon) \leq \delta\}$. Let $k \geq k_{\varepsilon, \delta}$. Then, $\delta \geq p_{k_{\varepsilon, \delta}}(\alpha_\varepsilon) \geq p_k(\alpha_\varepsilon)$. Thus, for all $k \geq k_{\varepsilon, \delta}$ we have $1 \geq \alpha_k \geq \alpha_\varepsilon \geq 1 - \varepsilon$, as required. \square

Furthermore, define the following random sequence M_k :

$$M_k := C(\varepsilon, \delta) \cdot \frac{1}{k\alpha_k} \|\mathbf{A}_{\text{rest}} \mathbf{\Psi}^{(k)}\|_F^2, \quad \mathbf{\Psi}^{(k)} = [\mathbf{\Psi}^{(k-1)} \quad \boldsymbol{\psi}^{(k)}], \quad \boldsymbol{\psi}^{(k)} \sim N(\mathbf{0}, \mathbf{I}). \quad (18)$$

By the law of large numbers we have $M_k \rightarrow C(\varepsilon, \delta) \|\mathbf{A}_{\text{rest}}\|_F^2$ almost surely as $k \rightarrow +\infty$. If we reuse the matrix vector products from line 18 in line 19 the total number of performed matrix vector products in the second phase of Algorithm 2 is

$$\max\{k, \lceil M_k \rceil\}. \quad (19)$$

Because of the monotonicity of α_k , and as seen in Figure 3, M_k is expected to decrease in k . Hence, in order to minimize (19) we choose k such that $k = \lceil M_k \rceil$. Thus, we evaluate M_k inside a while loop and stop the while loop once we detect $k > M_k$ for the first time. At this point we reuse the computation $\mathbf{A}_{\text{rest}} \mathbf{\Psi}^{(k)}$ to estimate $\text{tr}(\mathbf{A}_{\text{rest}})$. The resulting algorithm is presented in Algorithm 3. As with the prototype algorithm, Algorithm 3 can also be implemented to perform matrix-vector products in a block-wise fashion.

Algorithm 3 A-Hutch++

input: Symmetric $\mathbf{A} \in \mathbb{R}^{n \times n}$. Error tolerance $\varepsilon > 0$. Failure probability $\delta \in (0, 1)$. Block-size b .

output: An approximation to $\text{tr}(\mathbf{A}) : \text{tr}_{\text{adap}}(\mathbf{A})$.

- 1: Perform lines 1–16 in Algorithm 2 to get \mathbf{Q} , trest_1 and \mathbf{A}_{rest} .
 - 2: Initialize $\Psi^{(0)} = \square$ and $\mathbf{C}^{(0)} = \square$.
 - 3: Initialize $M_0 = \infty$ and $k = 0$.
 - 4: **while** $M_k > k$ **do**
 - 5: $k = k + b$
 - 6: $\alpha_k = \sup \left\{ \alpha \in (0, 1) : \frac{\gamma(\frac{k}{2}, \alpha \frac{k}{2})}{\Gamma(\frac{k}{2})} \leq \delta \right\}$
 - 7: Generate a random matrix $\widehat{\Psi}^{(k)} \in \mathbb{R}^{n \times b}$ and append $\Psi^{(k)} = \begin{bmatrix} \Psi^{(k-b)} & \widehat{\Psi}^{(k)} \end{bmatrix}$.
 - 8: Compute $\widehat{\mathbf{C}}^{(k)} = \mathbf{A}_{\text{rest}} \widehat{\Psi}^{(k)}$ and append $\mathbf{C}^{(k)} = \begin{bmatrix} \mathbf{C}^{(k-b)} & \widehat{\mathbf{C}}^{(k)} \end{bmatrix}$.
 - 9: Over-estimate $\|\mathbf{A}_{\text{rest}}\|_F^2$ with $\text{estFrob}_k = \frac{1}{k\alpha_k} \|\mathbf{C}^{(k)}\|_F^2$.
 - 10: Define $M_k = C(\varepsilon, \delta) \text{estFrob}_k$.
 - 11: **end while**
 - 12: **return** $\text{tr}_{\text{adap}}(\mathbf{A}) = \text{trest}_1 + \frac{1}{k} \text{tr}(\Psi^{(k)T} \mathbf{C}^{(k)})$
-

Due to the lack of independence between the Frobenius norm estimation and the stochastic trace estimation, the proof of Lemma 2.3 does not extend to Algorithm 3. In turn, this algorithm does not come with the same type of success guarantee. However, as presented in Section 2.3.1 the empirical failure probabilities remain well below the prescribed failure probability.

2.3 Numerical experiments

All numerical experiments in this paper have been performed in Matlab, version R2020a; our implementation of Algorithm 3 is available at <https://github.com/davpersson/A-Hutch-> together with the scripts to reproduce all figures and tables in this paper.

For a variety of matrices from [4, 11, 23, 27], we compare the newly proposed A-Hutch++ algorithm with Hutch++. In A-Hutch++ we fix $\delta = 0.05$ in all our experiments and we let $\varepsilon = \frac{|\text{tr}(\mathbf{A})|}{2^p}$ for $p = 2, 3, \dots, 10$, except in Figure 7b where we let $p = 3, 4, \dots, 11$. The error of the estimate produced by A-Hutch++ implemented in a block-wise fashion is essentially identical to the unblocked version of A-Hutch++, i.e. $b = 1$, as long as the block-size is small compared to the number of required matrix-vector products. Therefore, for simplicity, we set the block-size to $b = 1$ in all experiments. Furthermore, as discussed in Section 2.1.1 we set $\ell = 0$ and omit the side condition on m . For each considered matrix, for each value of ε , we first run Algorithm 3 and count the number of matrix-vector products that have been used to obtain the estimate, then we run Algorithm 1 with the same number of matrix-vector products. For each value of ε we repeat this 100 times and plot the average relative error on the

y-axis and the average required matrix-vector products on the x-axis. In each figure, the blue line is the average relative error from A-Hutch++, the red line is the average relative error from Hutch++, with the same number of matrix vector products, and the black dashed line is the ε that was used as the input tolerance of A-Hutch++. For matrices with slow eigenvalue decay we have also included the average relative error from the Hutchinson estimator (1), see the green line in Figures 4a,4b,7b, and 8a. The shaded blue area shows the 10th to 90th percentiles³ of the results from A-Hutch++, and the shaded red area shows the 10th to 90th percentiles of the results from Hutch++, see e.g. Figure 4.

In the numerical experiments we observe that A-Hutch++ performs better compared to Hutch++ for matrices with slower singular value decay; see e.g. Figure 4a, in which A-Hutch++ achieves an average relative error of 0.001827 using an average of 74.41 matrix-vector products (6th blue point in the figure). In comparison, Hutch++ achieves an average relative error of 0.001804 using an average of 237.7 matrix-vector products (7th red point in the figure). Hence, in these cases the adaptivity does improve the performance compared to Hutch++. For faster singular value decay the two algorithms perform similarly. However, in no case does Hutch++ perform noticeably better compared to A-Hutch++.

2.3.1 Synthetic matrices

We create matrices with algebraically decaying singular values as in [23], i.e. $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T \in \mathbb{R}^{5000 \times 5000}$ where \mathbf{U} is a random orthogonal matrix and $\mathbf{\Lambda}$ is a diagonal matrix with $\mathbf{\Lambda}_{ii} = 1/i^c$ for $i = 1, \dots, 5000$, for a parameter $c \in \{0.1, 0.5, 1, 3\}$. The results are shown in Figure 4.

Furthermore, using these example matrices we also estimated the failure probability of A-Hutch++. Table 1 demonstrates the empirical failure probabilities from 100000 repeats of A-Hutch++ for different input pairs (ε, δ) . In all cases the empirical failure probabilities remain well below the prescribed failure probability.

In addition, to demonstrate that A-Hutch++ allocates more matrix-vector products to the Hutchinson estimator for matrices with slow eigenvalue decay and vice versa for matrices with fast eigenvalue decay, we also include a table displaying the distribution of the matrix-vector products between the two phases. See Table 2.

³We show the 90% percentile because, if we did not reuse the matrix-vector products of the Frobenius norm estimation for the Hutchinson trace estimator, Lemma 2.3 would ensure a failure probability of at most $2\delta = 10\%$.

$\varepsilon \backslash \delta$	0.1	0.05	0.01
0.1 $\text{tr}(\mathbf{A})$	0	0	0
0.01 $\text{tr}(\mathbf{A})$	0.00285	0.00076	0.00005
0.005 $\text{tr}(\mathbf{A})$	0.00686	0.00244	0.00015

(a) $c = 0.1$

$\varepsilon \backslash \delta$	0.1	0.05	0.01
0.1 $\text{tr}(\mathbf{A})$	0	0	0
0.01 $\text{tr}(\mathbf{A})$	0.00484	0.00126	0.00010
0.005 $\text{tr}(\mathbf{A})$	0.00855	0.00331	0.00032

(b) $c = 0.5$

$\varepsilon \backslash \delta$	0.1	0.05	0.01
0.1 $\text{tr}(\mathbf{A})$	0.00026	0.00002	0
0.01 $\text{tr}(\mathbf{A})$	0.00607	0.00186	0.00018
0.005 $\text{tr}(\mathbf{A})$	0.00804	0.00250	0.00030

(c) $c = 1$

$\varepsilon \backslash \delta$	0.1	0.05	0.01
0.1 $\text{tr}(\mathbf{A})$	0	0	0
0.01 $\text{tr}(\mathbf{A})$	0.00002	0	0
0.005 $\text{tr}(\mathbf{A})$	0.00006	0	0

(d) $c = 3$

Table 1: Empirical failure probabilities from 100000 repeats of applying A-Hutch++ on the synthetic matrices described in Section 2.3.1.

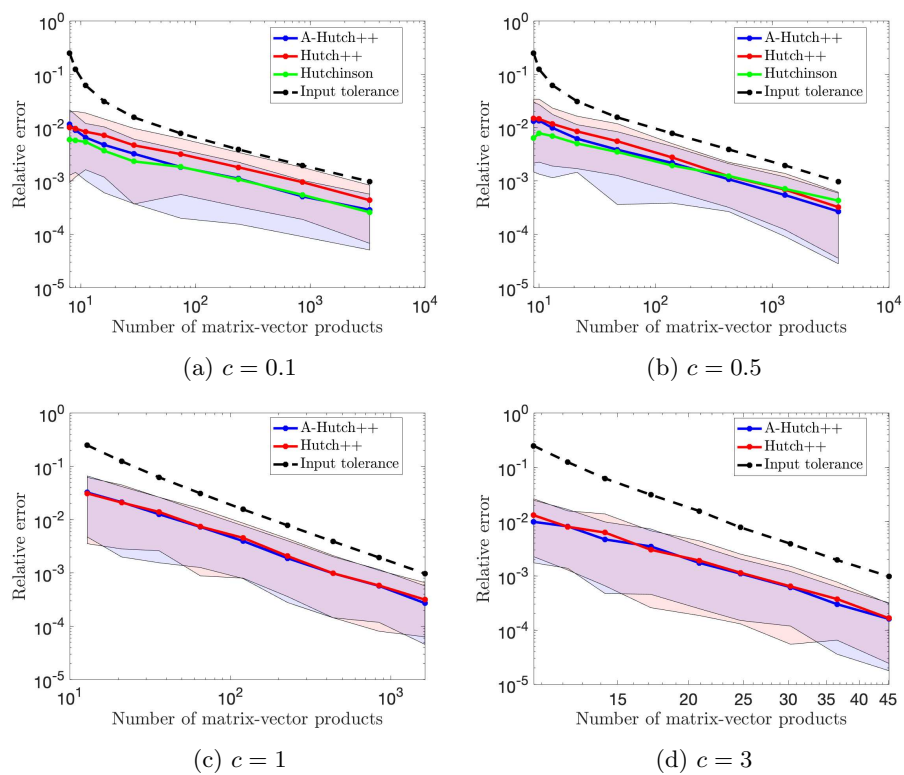


Figure 4: Comparison of A-Hutch++ and Hutch++ for the estimation of the trace of the synthetic matrices with algebraic decay from Section 2.3.1.

p	Total	Low rank approx.	Hutchinson est.	Ratio
2	8.00	6.00	2.00	0.25
3	9.00	6.00	3.00	0.33
4	11.00	6.00	5.00	0.45
5	16.00	6.00	10.00	0.63
6	29.04	6.00	23.04	0.79
7	74.41	6.00	68.41	0.92
8	237.66	6.00	231.66	0.97
9	858.13	6.00	852.13	0.99
10	3302.76	6.00	3296.76	1.00

(a) $c = 0.1$

p	Total	Low rank approx.	Hutchinson est.	Ratio
2	9.00	6.00	3.00	0.33
3	10.01	6.00	4.01	0.40
4	13.06	6.00	7.06	0.54
5	21.21	6.00	15.21	0.72
6	46.94	6.02	40.92	0.87
7	138.24	10.14	128.10	0.93
8	424.31	49.18	375.13	0.88
9	1287.60	206.96	1080.64	0.84
10	3688.39	914.18	2774.21	0.75

(b) $c = 0.5$

p	Total	Low rank approx.	Hutchinson est.	Ratio
2	12.86	6.16	6.70	0.52
3	21.07	8.86	12.21	0.58
4	36.02	15.08	20.94	0.58
5	65.15	27.68	37.47	0.58
6	120.04	52.84	67.20	0.56
7	228.02	101.18	126.84	0.56
8	436.75	199.14	237.61	0.54
9	843.98	396.32	447.66	0.53
10	1630.29	793.36	836.93	0.51

(c) $c = 1$

p	Total	Low rank approx.	Hutchinson est.	Ratio
2	10.66	8.20	2.46	0.23
3	12.24	8.88	3.36	0.27
4	14.24	10.76	3.48	0.24
5	17.16	12.44	4.72	0.28
6	20.91	15.22	5.69	0.27
7	24.70	18.28	6.42	0.26
8	30.28	22.50	7.78	0.26
9	36.57	27.68	8.89	0.24
10	45.14	34.50	10.64	0.24

(d) $c = 3$

Table 2: The average distribution of matrix-vector products between the low rank approximation phase and stochastic trace estimation phase of A-Hutch++ applied on the synthetic matrices with algebraic decay and input tolerance $\varepsilon = 2^{-p} \text{tr}(\mathbf{A})$ for $p = 2, 3, \dots, 10$. A-Hutch++ requires at least 6 matrix-vector products to detect a minimum of the function $\hat{m}(r)$ in (14).

2.3.2 Triangle counting

For an undirected graph with adjacency matrix \mathbf{B} , the number of triangles in the graph is equal to $\frac{1}{6} \text{tr}(\mathbf{B}^3)$; counting triangles arises for instance in data mining applications [2]. We apply A-Hutch++ and Hutch++ to $\mathbf{A} = \mathbf{B}^3$, where \mathbf{B} is the adjacency matrix of the following graphs:

- a Wikipedia vote network⁴ of size 7115 ($\text{tr}(\mathbf{B}^3) = 3650334$);
- an arXiv collaboration network⁵ of size 5242 ($\text{tr}(\mathbf{B}^3) = 289560$).

Note that one matrix-vector product with \mathbf{A} corresponds to three matrix-vector products with \mathbf{B} . The numerical results are shown in Figure 5.

⁴Accessed from <https://snap.stanford.edu/data/wiki-Vote.html>

⁵Accessed from <https://snap.stanford.edu/data/ca-GrQc.html>

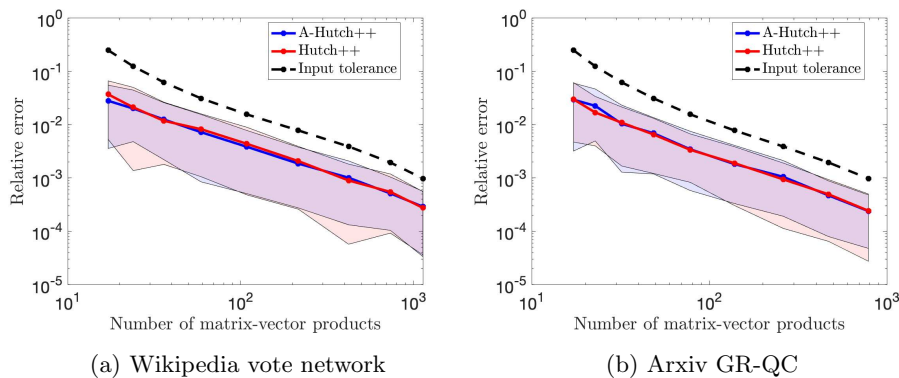


Figure 5: Comparison of A-Hutch++ and Hutch++ for the triangle counting examples from Section 2.3.2.

2.3.3 Estrada index

For an undirected graph with adjacency matrix \mathbf{B} , the Estrada index is defined as $\text{tr}(\exp(\mathbf{B}))$ and its applications include measuring the degree of protein protein folding [9] and network analysis [10]. As in [23], we estimate the Estrada index of Roget’s Thesaurus semantic graph adjacency matrix⁶. We approximate matrix-vector products with $\mathbf{A} = \exp(\mathbf{B})$ using 30 iterations of the Lanczos method [16, Chapter 13.2], after which the error from the approximated matrix-vector product is negligible. The results are shown in Figure 6.

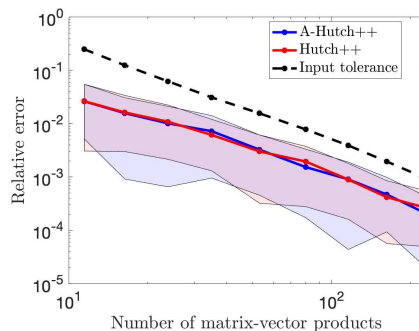
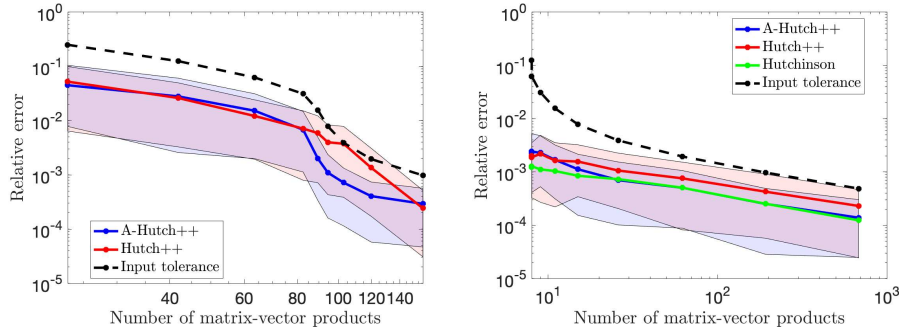


Figure 6: Comparison of A-Hutch++ and Hutch++ for the estimation of the Estrada index of the matrix from Section 2.3.3.

2.3.4 Log-determinant

The computation of the log-determinant of a symmetric positive definite matrix, which arises for instance in statistical learning [1] and Markov random fields

⁶Accessed from <http://vlado.fmf.uni-lj.si/pub/networks/data/>



(a) Estimating the log-determinant of the matrix from [27]. (b) Estimating the log-determinant of the matrix Thermomech TC.

Figure 7: Comparison of A-Hutch++ and Hutch++ for the log determinant estimation of the matrices from Section 2.3.4.

models [33], can be addressed by trace estimation exploiting the relation

$$\log \det(\mathbf{B}) = \text{tr}(\log(\mathbf{B})).$$

In our setting we apply A-Hutch++ and Hutch++ to $\mathbf{A} = \log(\mathbf{B})$ for the following symmetric positive definite matrices \mathbf{B} :

- $\mathbf{B} = \mathbf{I} + \sum_{j=1}^{40} \frac{10}{j^2} \mathbf{x}_j \mathbf{x}_j^T + \sum_{j=41}^{300} \frac{1}{j^2} \mathbf{x}_j \mathbf{x}_j^T$ where $\mathbf{x}_1, \dots, \mathbf{x}_{300} \in \mathbb{R}^{5000}$ are generated in Matlab using `sprandn(5000, 1, 0.025)`. This example comes from [27, 28]. \mathbf{B} has an eigenvalue gap at index 40. Matrix-vector products with $\mathbf{A} = \log(\mathbf{B})$ are approximated using 25 iterations of Lanczos method.
- \mathbf{B} is the Thermomech TC matrix⁷ from the SuiteSparse Matrix Collection [7]. Matrix-vector products with $\mathbf{A} = \log(\mathbf{B})$ are approximated using 35 iterations of Lanczos method.

The numerical results are shown in Figure 7.

2.3.5 Trace of inverses

We consider $\mathbf{A} = \mathbf{B}^{-1}$ for the following choices of \mathbf{B} :

- $\mathbf{B} = \text{tridiag}(-1, 4, -1)$ is a 10000×10000 tridiagonal matrix with 4 along the diagonal and -1 along the upper and lower subdiagonal (taken from [11]);

⁷Accessed from https://sparse.tamu.edu/Botonakis/thermomech_TC

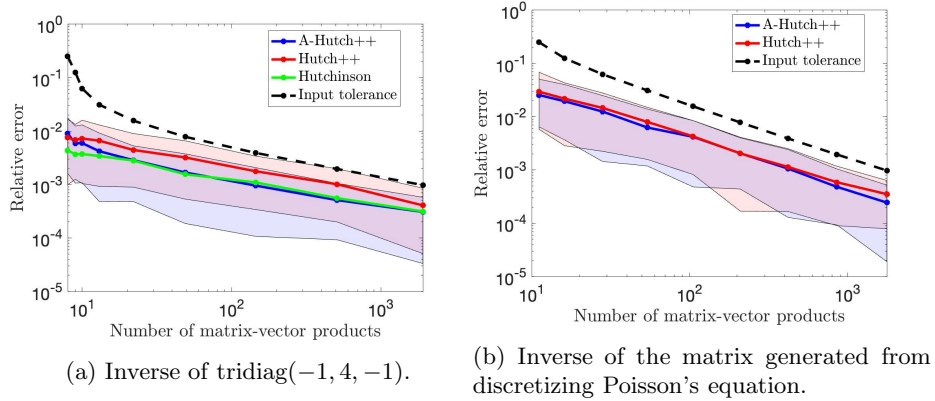


Figure 8: Comparison of A-Hutch++ and Hutch++ for the estimation of the trace of the inverse of the matrices described in Section 2.3.5.

- \mathbf{B} a block tridiagonal matrix of size $k^2 \times k^2$ generated from discretizing Poisson's equation with the 5-point operator on a $k \times k$ mesh, with $k = 100$ (taken from [4]).

Matrix-vector products with $\mathbf{A} = \mathbf{B}^{-1}$ are computed using backslash in Matlab. The results are shown in Figure 8.

3 Nyström++

As explained in the introduction, Hutch++ requires at least two passes over the matrix \mathbf{A} . In [23], Algorithm 4 was presented, and its analysis was improved in [19]. It requires only one pass over the input matrix, when computing the matrix vector products in line 3, and we thus call it *Single Pass Hutch++*. It

Algorithm 4 Single Pass Hutch++

input: Symmetric positive semi-definite $\mathbf{A} \in \mathbb{R}^{n \times n}$. Number of matrix-vector products $m \in \mathbb{N}$.

output: An approximation to $\text{tr}(\mathbf{A}) : \text{tr}_m^{\text{sph}++}(\mathbf{A})$

- 1: Fix positive constants c_1, c_2 and c_3 such that $c_1 < c_2$ and $c_1 + c_2 + c_3 = 1$
 - 2: Sample $\mathbf{\Omega} \in \mathbb{R}^{d \times c_1 m}$, $\mathbf{\Psi} \in \mathbb{R}^{d \times c_2 m}$, $\mathbf{\Phi} \in \mathbb{R}^{d \times c_3 m}$ with i.i.d. $N(0, 1)$ or Rademacher entries
 - 3: Compute $\begin{bmatrix} \mathbf{X} & \mathbf{Y} & \mathbf{Z} \end{bmatrix} = \mathbf{A} \begin{bmatrix} \mathbf{\Omega} & \mathbf{\Psi} & \mathbf{\Phi} \end{bmatrix}$
 - 4: **return** $\text{tr}_m^{\text{sph}++}(\mathbf{A}) = \text{tr}((\mathbf{\Omega}^T \mathbf{Y})^\dagger (\mathbf{X}^T \mathbf{Y})) + \frac{1}{c_3 m} (\text{tr}(\mathbf{\Phi}^T \mathbf{Z}) - \text{tr}(\mathbf{\Phi}^T \mathbf{Y} (\mathbf{\Omega}^T \mathbf{Y})^\dagger \mathbf{X}^T \mathbf{\Phi}))$
-

also fits the streaming model because an update $\mathbf{A} + \mathbf{E}$ of the input matrix

trivially translated into an update of the matrix-vector products, without having to revisit \mathbf{A} . It is similar to Hutch++ since it consists of a randomized low rank approximation phase and a stochastic trace estimation phase. The low rank approximation phase is performed by computing the low rank approximation $\mathbf{A}\Psi(\Omega^T\mathbf{A}\Psi)^\dagger(\mathbf{A}\Omega)^T = \mathbf{Y}(\Omega^T\mathbf{Y})^\dagger\mathbf{X}^T$, where \mathbf{X}, \mathbf{Y} and \mathbf{Z} are as in line 3 of Single Pass Hutch++. The trace of the low rank approximation equals $\text{tr}((\Omega^T\mathbf{Y})^\dagger(\mathbf{X}^T\mathbf{Y}))$ via the cyclic property of the trace. In the stochastic trace estimation phase the trace of $\mathbf{A} - \mathbf{Y}(\Omega^T\mathbf{Y})^\dagger\mathbf{X}^T$ is estimated, which is done by the stochastic trace estimator (1). Single Pass Hutch++ satisfies similar guarantees as Hutch++, but is observed to produce a less accurate trace estimate than Hutch++ with the same number of matrix-vector products. More formally, the following result was proved.

Theorem 3.1 ([19, Theorem 1.1]). *If Single Pass Hutch++ is implemented with $m = O\left(\varepsilon^{-1}\sqrt{\log(\delta^{-1})} + \log(\delta^{-1})\right)$ matrix-vector products then*

$$|\text{tr}_m^{\text{sph}++}(\mathbf{A}) - \text{tr}(\mathbf{A})| \leq \varepsilon \text{tr}(\mathbf{A}).$$

holds with probability at least $1 - \delta$.

On the other hand, the numerical experiments in [19] demonstrated that due to the single pass property, which allows for performing matrix-vector products in parallel, Single Pass Hutch++ outperforms Hutch++ in terms of wall-clock time.⁸

For symmetric positive semi-definite \mathbf{A} one can obtain a version of Hutch++ by utilizing the Nyström approximation $\mathbf{A} \approx \mathbf{A}\Omega(\Omega^T\mathbf{A}\Omega)^\dagger\Omega^T\mathbf{A}$ [13] instead. We call this algorithm Nyström++, see Algorithm 5. The idea of using the Nyström approximation in the context of trace estimation had previously been presented in [20, Section 4] in a broader context, but no analysis was presented. A version of Hutch++ using a similar low-rank approximation was also mentioned in [22]. Furthermore, Nyström++ also fits the streaming model. Another possible advantage of Nyström++ over Hutch++ is that while the Nyström approximation is less accurate than the randomized SVD, one can spend more matrix-vector products for both attaining a low-rank approximation of \mathbf{A} and on estimating the trace of $\mathbf{A} - \mathbf{A}\Omega(\Omega^T\mathbf{A}\Omega)^\dagger\Omega^T\mathbf{A}$.

⁸One needs to be careful how to implement the low-rank approximation in Single Pass Hutch++, since it is prone to numerical instabilities due to the pseudoinverse of $\Omega^T\mathbf{Y}$. In our implementation we follow the suggestion given in [24, Section 5.1]. We compute a thin QR-decomposition of $(\Omega^T\mathbf{Y})^T = \mathbf{Q}\mathbf{R}$ and let $\mathbf{S} = \mathbf{Y}\mathbf{Q}$ and $\mathbf{Z} = \mathbf{X}\mathbf{R}^{-1}$. Then $\mathbf{Y}(\Omega^T\mathbf{Y})^\dagger\mathbf{X}^T = \mathbf{S}\mathbf{Z}^T$.

Algorithm 5 Nyström++

input: Symmetric positive semi-definite $\mathbf{A} \in \mathbb{R}^{n \times n}$. Number of matrix-vector products $m \in \mathbb{N}$ (multiple of 2).

output: An approximation to $\text{tr}(\mathbf{A}) : \text{tr}_m^{n++}(\mathbf{A})$.

1: Sample $\mathbf{\Omega} \in \mathbb{R}^{n \times \frac{m}{2}}$, $\mathbf{\Phi} \in \mathbb{R}^{n \times \frac{m}{2}}$ with i.i.d. $N(0, 1)$ entries.

2: Compute $[\mathbf{X} \ \mathbf{Y}] = \mathbf{A} [\mathbf{\Omega} \ \mathbf{\Phi}]$.

3: **return** $\text{tr}_m^{n++}(\mathbf{A}) = \text{tr}((\mathbf{\Omega}^T \mathbf{X})^\dagger (\mathbf{X}^T \mathbf{X})) + \frac{2}{m} (\text{tr}(\mathbf{\Phi}^T \mathbf{Y}) - \text{tr}(\mathbf{\Phi}^T \mathbf{X} (\mathbf{\Omega}^T \mathbf{X})^\dagger \mathbf{X}^T \mathbf{\Phi}))$

Recall that the trace of the Nyström approximation $\mathbf{X}(\mathbf{\Omega}^T \mathbf{X})^\dagger \mathbf{X}^T$ equals $\text{tr}((\mathbf{\Omega}^T \mathbf{X})^\dagger (\mathbf{X}^T \mathbf{X}))$ via the cyclic property of the trace.

3.1 Analysis of Nyström++

In the following, we show that Algorithm 5 enjoys the same theoretical guarantees as Algorithm 1 [23, Theorem 1.1]. We begin with a result on the Frobenius norm error of the Nyström approximation.

Lemma 3.2. *Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be symmetric positive semidefinite and let $\mathbf{\Omega} \in \mathbb{R}^{n \times 2k}$ be a standard Gaussian matrix with $k \geq 5$. Then*

$$\|\mathbf{A} - \mathbf{A}\mathbf{\Omega}(\mathbf{\Omega}^T \mathbf{A}\mathbf{\Omega})^\dagger \mathbf{\Omega}^T \mathbf{A}\|_F \leq \frac{542}{\sqrt{k}} \text{tr}(\mathbf{A})$$

holds with probability at least $1 - 6e^{-k}$.

The proof of Lemma 3.2 builds on the following result.

Lemma 3.3 ([13, Theorem 3]). *For a symmetric positive semidefinite matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ of rank at least k , let $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ be a spectral decomposition with the eigenvalues in non-increasing order on the diagonal of $\mathbf{\Lambda}$. Partition $\mathbf{U} = [\mathbf{U}_1 \ \mathbf{U}_2]$ such that $\mathbf{U}_1 \in \mathbb{R}^{n \times k}$ and $\mathbf{\Lambda} = \begin{bmatrix} \mathbf{\Lambda}_1 & \\ & \mathbf{\Lambda}_2 \end{bmatrix}$ such that $\mathbf{\Lambda}_1 \in \mathbb{R}^{k \times k}$.*

Let $p \geq 1$ be an oversampling parameter and let $\mathbf{\Omega} \in \mathbb{R}^{n \times (k+p)}$ be such that $\mathbf{\Psi}_1 := \mathbf{U}_1^T \mathbf{\Omega}$ has full rank and define $\mathbf{\Psi}_2 := \mathbf{U}_2^T \mathbf{\Omega}$. Then

$$\|\mathbf{A} - \mathbf{A}\mathbf{\Omega}(\mathbf{\Omega}^T \mathbf{A}\mathbf{\Omega})^\dagger \mathbf{\Omega}^T \mathbf{A}\|_F \leq \|\mathbf{\Lambda}_2\|_F + \|\mathbf{\Lambda}_2^{1/2} \mathbf{\Psi}_2 \mathbf{\Psi}_1^\dagger\|_2 \left(\sqrt{2\|\mathbf{\Lambda}_2\|_*} + \|\mathbf{\Lambda}_2^{1/2} \mathbf{\Psi}_2 \mathbf{\Psi}_1^\dagger\|_F \right). \quad (20)$$

Proof of Lemma 3.2. By proceeding as in the beginning of the proof of [13, Lemma 7] with probability at least $1 - 3e^{-k}$ we have⁹

$$\begin{aligned} \|\mathbf{\Lambda}_2^{1/2} \mathbf{\Psi}_2 \mathbf{\Psi}_1^\dagger\|_2 &\leq \|\mathbf{\Lambda}_2^{1/2}\|_2 \left(\sqrt{\frac{3k}{k+1}} e + \frac{2e^2 k}{k+1} \right) + \|\mathbf{\Lambda}_2^{1/2}\|_F \frac{e^2 \sqrt{2k}}{k+1} \\ &\leq \sqrt{\|\mathbf{\Lambda}_2\|_2} (\gamma_1 + \gamma_2) + \sqrt{\|\mathbf{\Lambda}_2\|_*} \frac{\gamma_3}{\sqrt{k}} \end{aligned} \quad (21)$$

⁹In the setting of [13, Lemma 7], set the quantities $p = k$, $t = e$, $u = \sqrt{2k}$ and $\mathbf{D} = \mathbf{\Lambda}_2^{1/2}$ to obtain (21) and (22).

by letting $\gamma_1 := \sqrt{3}e$, $\gamma_2 := 2e^2$ and $\gamma_3 := \sqrt{2}e^2$, where $\|\cdot\|_*$ denotes the nuclear norm defined in Section 1.2. Similarly, we have with probability at least $1 - 3e^{-k}$

$$\begin{aligned} \|\mathbf{\Lambda}_2^{1/2} \mathbf{\Psi}_2 \mathbf{\Psi}_1^\dagger\|_F &\leq \|\mathbf{\Lambda}_2^{1/2}\|_F \sqrt{\frac{3k}{k+1}} t + \|\mathbf{\Lambda}_2^{1/2}\|_2 \frac{2e^2 k}{k+1} \\ &\leq \sqrt{\|\mathbf{\Lambda}_2\|_*} \gamma_1 + \sqrt{\|\mathbf{\Lambda}_2\|_2} \gamma_2. \end{aligned} \quad (22)$$

By the union bound, both (21) and (22) hold simultaneously with probability at least $1 - 6e^{-k}$.

$\mathbf{\Psi}_1 \in \mathbb{R}^{k \times 2k}$ is a standard Gaussian matrix and therefore has full row rank almost surely. We may therefore apply Lemma 3.3 combined with the bounds (21) and (22). Hence, with probability at least $1 - 6e^{-k}$ we have

$$\begin{aligned} &\|\mathbf{A} - \mathbf{A}\mathbf{\Omega}(\mathbf{\Omega}^T \mathbf{A}\mathbf{\Omega})^\dagger \mathbf{\Omega}^T \mathbf{A}\|_F \\ &\leq \|\mathbf{\Lambda}_2\|_F + \|\mathbf{\Lambda}_2^{1/2} \mathbf{\Psi}_2 \mathbf{\Psi}_1^\dagger\|_2 \left(\sqrt{2\|\mathbf{\Lambda}_2\|_*} + \|\mathbf{\Lambda}_2^{1/2} \mathbf{\Psi}_2 \mathbf{\Psi}_1^\dagger\|_F \right) \\ &\leq \|\mathbf{\Lambda}_2\|_F + \left(\sqrt{\|\mathbf{\Lambda}_2\|_2}(\gamma_1 + \gamma_2) + \sqrt{\|\mathbf{\Lambda}_2\|_*} \frac{\gamma_3}{\sqrt{k}} \right) \left(\sqrt{\|\mathbf{\Lambda}_2\|_*}(\sqrt{2} + \gamma_1) + \sqrt{\|\mathbf{\Lambda}_2\|_2} \gamma_2 \right) \\ &= \|\mathbf{\Lambda}_2\|_F + \sqrt{\|\mathbf{\Lambda}_2\|_2} \|\mathbf{\Lambda}_2\|_* \tilde{\gamma}_1 + \|\mathbf{\Lambda}_2\|_2 \tilde{\gamma}_2 + \|\mathbf{\Lambda}_2\|_* \frac{\tilde{\gamma}_3}{\sqrt{k}} \\ &\leq \frac{1 + \tilde{\gamma}_1 + \tilde{\gamma}_2 + \tilde{\gamma}_3}{\sqrt{k}} \text{tr}(\mathbf{A}) \end{aligned}$$

where we set

$$\tilde{\gamma}_1 := (\gamma_1 + \gamma_2)(\sqrt{2} + \gamma_1) + \frac{\gamma_2 \gamma_3}{\sqrt{k}}, \quad \tilde{\gamma}_2 := (\gamma_1 + \gamma_2) \gamma_2, \quad \tilde{\gamma}_3 := \gamma_3(\sqrt{2} + \gamma_1)$$

and use the norm inequalities

$$\begin{aligned} \|\mathbf{\Lambda}_2\|_2 &\leq \|\mathbf{\Lambda}_2\|_F \leq \sqrt{\|\mathbf{\Lambda}_2\|_2 \|\mathbf{\Lambda}_2\|_*} \leq \frac{1}{\sqrt{k}} \|\mathbf{\Lambda}\|_* = \frac{1}{\sqrt{k}} \text{tr}(\mathbf{A}) \\ \|\mathbf{\Lambda}_2\|_* &\leq \|\mathbf{\Lambda}\|_* = \text{tr}(\mathbf{A}) \end{aligned}$$

in the last step. The proof is completed by noting that $1 + \tilde{\gamma}_1 + \tilde{\gamma}_2 + \tilde{\gamma}_3 \leq 542$. \square

We can now proceed to extend the main result on Hutch++ [23, Theorem 1.1] to Nyström++.

Theorem 3.4. *Suppose that Algorithm 5 (Nyström++) is executed with $m = O(\varepsilon^{-1} \sqrt{\log(\delta^{-1})} + \log(\delta^{-1}))$ matrix-vector products and $\delta \in (0, 1/2)^{10}$. Then its output satisfies*

$$|\text{tr}_m^{n++}(\mathbf{A}) - \text{tr}(\mathbf{A})| \leq \varepsilon \text{tr}(\mathbf{A}) \quad (23)$$

with probability at least $1 - \delta$.

¹⁰This condition on δ allows us to bound all $\log(p\delta^{-1})$ terms that would otherwise appear in the proof (see e.g. Lemma 2.1 where the term $\log(2\delta^{-1})$ appears) from above with $c \log(\delta^{-1})$ for some sufficiently large constant c .

Proof. We follow the proof of [23, Theorem 1.1]. Let us first recall that $\mathbf{X} = \mathbf{A}\mathbf{\Omega}$, $\mathbf{Y} = \mathbf{A}\mathbf{\Phi}$ for $d \times m/2$ standard Gaussian random matrices $\mathbf{\Omega}, \mathbf{\Phi}$ in Algorithm 5. Throughout the proof, we assume that $m \geq c \log(\delta^{-1})$ for some (sufficiently large) constant c .

By Lemma 3.2, there is a constant C_1 such that

$$\|\mathbf{A} - \mathbf{X}(\mathbf{\Omega}^T \mathbf{X})^\dagger \mathbf{X}^T\|_F \leq C_1 m^{-1/2} \text{tr}(\mathbf{A}), \quad (24)$$

with probability at least $1 - \delta/2$. By Lemma 2.1 there is a constant C_2 such that

$$\begin{aligned} & |\text{tr}(\mathbf{A} - \mathbf{X}(\mathbf{\Omega}^T \mathbf{X})^\dagger \mathbf{X}^T) - \text{tr}_{m/2}(\mathbf{A} - \mathbf{X}(\mathbf{\Omega}^T \mathbf{X})^\dagger \mathbf{X}^T)| \\ & \leq C_2 m^{-1/2} \sqrt{\log(\delta^{-1})} \|\mathbf{A} - \mathbf{X}(\mathbf{\Omega}^T \mathbf{X})^\dagger \mathbf{X}^T\|_F. \end{aligned}$$

with probability at least $1 - \delta/2$. By the union bound it holds with probability at least $1 - \delta$ that

$$\begin{aligned} |\text{tr}_m^{\text{n}++}(\mathbf{A}) - \text{tr}(\mathbf{A})| &= |\text{tr}(\mathbf{A} - \mathbf{X}(\mathbf{\Omega}^T \mathbf{X})^\dagger \mathbf{X}^T) - \text{tr}_{m/2}(\mathbf{A} - \mathbf{X}(\mathbf{\Omega}^T \mathbf{X})^\dagger \mathbf{X}^T)| \\ &\leq C_2 m^{-1/2} \sqrt{\log(\delta^{-1})} \|\mathbf{A} - \mathbf{X}(\mathbf{\Omega}^T \mathbf{X})^\dagger \mathbf{X}^T\|_F \\ &\leq C_1 C_2 m^{-1} \sqrt{\log(\delta^{-1})} \text{tr}(\mathbf{A}). \end{aligned}$$

Hence, setting $m = O(\varepsilon^{-1} \sqrt{\log(\delta^{-1})} + \log(\delta^{-1}))$ implies the claim. \square

3.2 Adaptive Nyström++

It is natural to aim at designing an adaptive version of Nyström++. Following A-Hutch++ we would need to find the minimum of

$$m(r) = r + C(\varepsilon, \delta) \|\mathbf{A} - \mathbf{A}_n^{(r)}\|_F^2, \quad (25)$$

where $\mathbf{A}_n^{(r)}$ is the rank- r Nyström approximation. Such an adaptive version clearly does not fit the streaming model. Moreover, we lose another advantage of Nyström++, that it only needs to perform r matrix-vector products with \mathbf{A} to get a rank- r approximation, compared to $2r$ for the randomized SVD. Since we cannot compute $\|\mathbf{A} - \mathbf{A}_n^{(r)}\|_F^2$ we would need to decompose this term as done in (13). This yields $\|\mathbf{A} - \mathbf{A}_n^{(r)}\|_F^2 = \|\mathbf{A}\|_F^2 - 2 \text{tr}(\mathbf{A}\mathbf{A}_n^{(r)}) + \|\mathbf{A}_n^{(r)}\|_F^2$. However, evaluating the term $-2 \text{tr}(\mathbf{A}\mathbf{A}_n^{(r)}) + \|\mathbf{A}_n^{(r)}\|_F^2$ depending on r requires additional matrix-vector products with \mathbf{A} . In summary, there is little advantage of using such an adaptive version of Nyström.

3.3 Numerical results

To deal with potential numerical instabilities due to the appearance of the pseudoinverse in the Nyström approximation in line 3 of Algorithm 5, in our implementation we use [21, Algorithm 16]. This algorithm computes an eigenvalue decomposition $\mathbf{U}\mathbf{\Sigma}\mathbf{U}^T$ of the Nyström approximation of $\mathbf{A} + \nu \mathbf{I}$, where ν is a

small shift, without explicitly forming the Nyström approximation. Once the eigenvalue decomposition is obtained the algorithm removes the shift by setting $\mathbf{\Lambda} = \max\{0, \mathbf{\Sigma} - \nu \mathbf{I}\}$ and returns $\mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, in factored form, as the stabilized Nyström approximation. The shift is set as $\nu = \sqrt{n}\mathbf{eps}(\|\mathbf{A}\mathbf{\Omega}\|_2)$, where $\mathbf{eps}(x)$ returns the distance to the next larger double precision floating point number to $x \in \mathbb{R}$ and $\mathbf{\Omega}$ is as in Algorithm 5. For further details, we refer to [21, 30].

We compare Nyström++ with Hutch++ and Single Pass Hutch++. We consider $m = 12 + 48k$ for $k \in \{0, 1, 2, \dots, 20\}$ and for each value of m we run Hutch++, Single Pass Hutch++ and Nyström++ 100 times each. We run the experiments on the matrices from Section 2.3.1, Section 2.3.3, and Section 2.3.5. Moreover, we create two matrices with exponential decay, i.e. $\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T \in \mathbb{R}^{5000 \times 5000}$ where \mathbf{Q} is a random orthogonal matrix and $\mathbf{\Lambda}$ is the diagonal matrix with entries $\mathbf{\Lambda}_{ii} = \exp(-i/s)$ for $i = 1, \dots, 5000$, where s is a parameter controlling the rate of the decay. We let $s = 10$ and $s = 100$.

The results are displayed in Figures 9, 10, 11, and 12, respectively. In each figure, the blue line is the average relative error from Nyström++, the red line is the average relative error from Hutch++ and the green line is the average relative error from Single Pass Hutch++. The shaded blue area shows the 10th to 90th percentiles of the results from Nyström++, and the shaded red area shows the 10th to 90th percentiles of the results from Hutch++.

In all cases we observe that Single Pass Hutch++ is the weakest alternative. Moreover, in many cases Hutch++ and Nyström++ have similar performances, and in some cases Nyström++ outperforms Hutch++, see e.g. Figure 12.

4 Conclusion

We have presented an adaptive version of Hutch++, A-Hutch++, that will estimate the trace of a symmetric matrix \mathbf{A} while attempting to minimize the number of matrix-vector products with \mathbf{A} used overall. This algorithm also comes with the advantage that the user does not need to determine the number of matrix-vector products required to output an estimate of the trace that is within the prescribed error tolerance. We have tested A-Hutch++ on a variety of examples and we found that A-Hutch++ in many cases provided some improvement and in any case, it did not require more matrix-vector products compared to Hutch++ to achieve the same error. Furthermore, we presented a version of Hutch++ utilizing the Nyström approximation, which requires only one pass over the matrix. We proved that this algorithm satisfies the same theoretical guarantees of Hutch++. While this algorithm offers a similar performance as Hutch++, it performs significantly better than the previously proposed single pass algorithm Single Pass Hutch++.

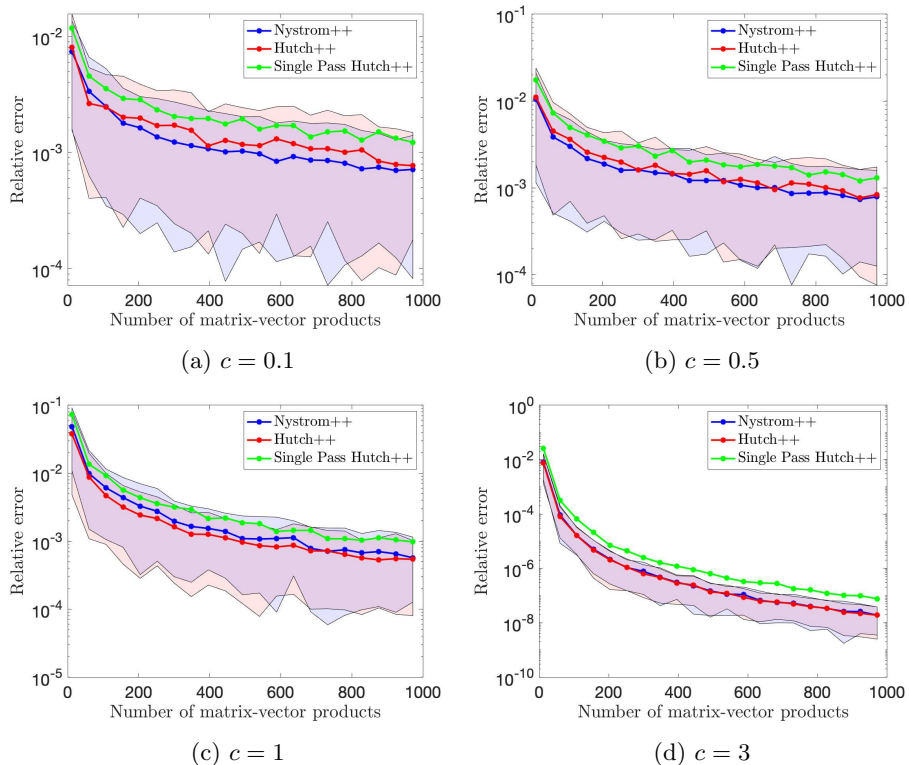


Figure 9: Comparison of Hutch++, Single Pass Hutch++ and Nyström++ for the estimation of the trace of the synthetic matrices with algebraic decay described in Section 2.3.1.

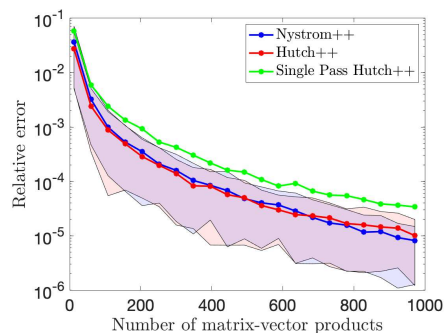


Figure 10: Comparison of Hutch++, Single Pass Hutch++ and Nyström++ for the estimation of the Estrada index as described in Section 2.3.3.

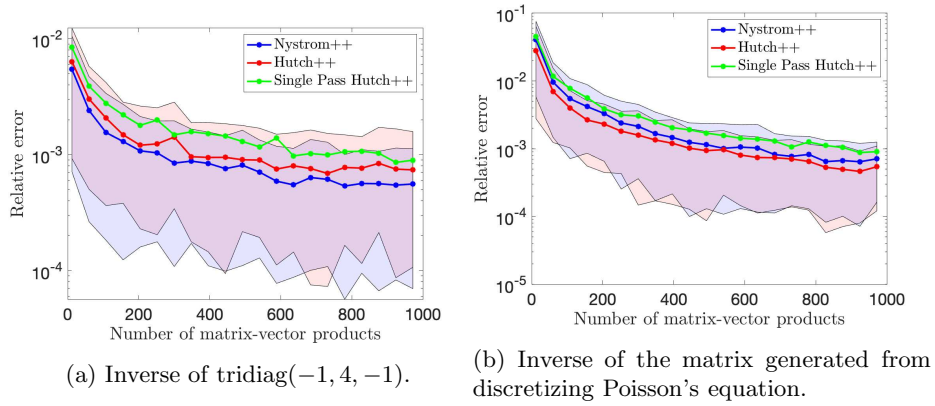


Figure 11: Comparison of Hutch++, Single Pass Hutch++ and Nyström++ for the estimation of the trace of the inverse of the matrices described in Section 2.3.5.

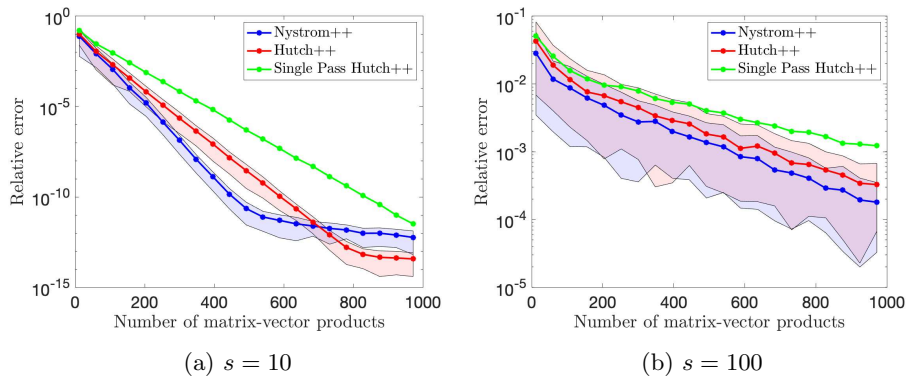


Figure 12: Comparison of Hutch++, Single Pass Hutch++ and Nyström++ for the estimation of the trace of the synthetic matrices with exponential decay described in Section 3.3.

Appendices

In this section we prove a version of the Hanson-Wright inequality and show that this is slightly stronger than the bound in [6, Theorem 1]. From this inequality we derive a version of Lemma 2.1. We conclude with proving that this bound is asymptotically optimal.

A Hanson-Wright Inequality

The Hanson-Wright Inequality we wish to prove is the following:

Theorem A.1. *Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be symmetric. Let $\boldsymbol{\omega}$ be a standard Gaussian vector of length n . Further, choose an arbitrary $c \in (0, 1/2)$ and define $C = -\frac{1}{c} - \frac{\log(1-2c)}{2c^2}$. Then we have*

$$\mathbb{P}(|\boldsymbol{\omega}^T \mathbf{A} \boldsymbol{\omega} - \text{tr}(\mathbf{A})| \geq \varepsilon) \leq 2 \exp\left(-\min\left\{\frac{\varepsilon^2}{4C\|\mathbf{A}\|_F^2}, \frac{c\varepsilon}{2\|\mathbf{A}\|_2}\right\}\right)$$

Theorem A.1 will be proved using tail bounds for sub-Exponential random variables.

Definition A.1 (Sub-Exponential Random Variable). A random variable X is called sub-Exponential with parameters $\nu^2, \alpha > 0$ if

$$\mathbb{E} \exp(t(X - \mathbb{E}X)) \leq \exp\left(\frac{t^2 \nu^2}{2}\right) \quad \text{for all } |t| \leq \frac{1}{\alpha}.$$

For sub-Exponential random variables one has the following result, which follows from a Chernoff bound.

Lemma A.2. ([32, Proposition 2.9]) *If X is a sub-Exponential random variable with parameters (ν^2, α) . Then*

$$\mathbb{P}(|X - \mathbb{E}X| > \varepsilon) \leq 2 \exp\left(-\frac{1}{2} \min\left\{\frac{\varepsilon^2}{\nu^2}, \frac{\varepsilon}{\alpha}\right\}\right).$$

In order to prove Theorem A.1 we require the following 3 lemmas. They are proved using basic calculus techniques are therefore omitted.

Lemma A.3. *If $C = -\frac{1}{c} - \frac{\log(1-2c)}{2c^2}$ for $c \in (0, 1/2)$, then $C > 1$ and $\lim_{c \rightarrow 0^+} C = 1$.*

Lemma A.4. *For $0 \leq x \leq c < \frac{1}{2}$ we have*

$$\frac{\exp(-x)}{\sqrt{1-2x}} \leq \exp(Cx^2)$$

where $C = -\frac{1}{c} - \frac{\log(1-2c)}{2c^2}$

Lemma A.5. For $x \geq 0$ we have $\frac{\exp(x)}{\sqrt{1+2x}} \leq \exp(x^2) \leq \exp(Cx^2)$ where C as in Lemma A.4. Furthermore, for $x \in (-1/2, 0]$ we have $\frac{\exp(x)}{\sqrt{1+2x}} \geq \exp(x^2)$. In particular, for $x \in [0, 1/2)$ we have $\frac{\exp(-x)}{\sqrt{1-2x}} \geq \exp(x^2)$.

We can now proceed to prove Theorem A.1.

Proof of Theorem A.1. Let $X = \boldsymbol{\omega}^T \mathbf{A} \boldsymbol{\omega}$ where $\boldsymbol{\omega} \sim N(\mathbf{0}, \mathbf{I})$. We will show that X is sub-Exponential with parameters $(2C\|\mathbf{A}\|_F^2, \frac{\|\mathbf{A}\|_2}{c})$. The final result will follow from Lemma A.2.

Note that since \mathbf{A} is symmetric we have

$$\mathbf{A} = \mathbf{Q} \boldsymbol{\Lambda} \mathbf{Q}^T$$

where \mathbf{Q} is orthogonal and

$$\boldsymbol{\Lambda} = \begin{bmatrix} \boldsymbol{\Lambda}_+ & & \\ & \mathbf{0} & \\ & & \boldsymbol{\Lambda}_- \end{bmatrix}$$

where $\boldsymbol{\Lambda}_\pm = \text{diag}(\pm\lambda_1^\pm, \dots, \pm\lambda_{l_\pm}^\pm)$ where $\lambda_i^\pm > 0 \quad \forall i = 1, \dots, l_\pm$.

Note that by unitary invariance of Gaussian vectors we have

$$X - \mathbb{E}X = \boldsymbol{\omega}^T \mathbf{A} \boldsymbol{\omega} - \text{tr}(\mathbf{A}) \stackrel{d}{=} \sum_{i=1}^{l_+} \lambda_i^+ ((\omega_i^+)^2 - 1) - \sum_{j=1}^{l_-} \lambda_j^- ((\omega_j^-)^2 - 1)$$

Hence,

$$\mathbb{E} \exp(t(X - \mathbb{E}X)) = \left(\prod_{i=1}^{l_+} \mathbb{E} \exp(t\lambda_i^+ ((\omega_i^+)^2 - 1)) \right) \left(\prod_{j=1}^{l_-} \mathbb{E} \exp(-t\lambda_j^- ((\omega_j^-)^2 - 1)) \right)$$

Note that

$$\mathbb{E} \exp(\pm t\lambda_k^\pm ((\omega_k^\pm)^2 - 1)) = \frac{\exp(\mp \lambda_k^\pm t)}{\sqrt{1 \mp 2\lambda_k^\pm t}} \quad (26)$$

provided $\pm t\lambda_k^\pm < 1/2$.

By Lemma A.4 and A.5 we have

$$\frac{\exp(\mp \lambda_k^\pm t)}{\sqrt{1 - 2\lambda_k^\pm t}} \leq \exp(C(\lambda_k^\pm)^2 t^2), \quad |t\lambda_i^\pm| \leq c < 1/2$$

Hence, by inserting this into (26) we get

$$\begin{aligned} \mathbb{E} \exp(t(X - \mathbb{E}X)) &\leq \left(\prod_{i=1}^{l_+} \exp(Ct^2(\lambda_i^+)^2) \right) \left(\prod_{j=1}^{l_-} \exp(Ct^2(\lambda_j^-)^2) \right) \\ &= \exp(Ct^2\|\mathbf{A}\|_F^2) \end{aligned}$$

provided

$$|t| \leq \frac{c}{\|\mathbf{A}\|_2}$$

Hence, X is sub-Exponential with parameters $\left(2C\|\mathbf{A}\|_F^2, \frac{\|\mathbf{A}\|_2}{c}\right)$. \square

The following corollary can be proved via the diagonal embedding trick [6, Theorem 1].

Corollary A.5.1. *Let \mathbf{A} be as in Theorem A.1. Let $\text{tr}_m(\mathbf{A})$ be the stochastic trace estimator with m samples of i.i.d. standard Gaussian vectors. Then we have*

$$\mathbb{P}(|\text{tr}_m(\mathbf{A}) - \text{tr}(\mathbf{A})| \geq \varepsilon) \leq 2 \exp\left(-m \min\left\{\frac{\varepsilon^2}{4C\|\mathbf{A}\|_F^2}, \frac{c\varepsilon}{2\|\mathbf{A}\|_2}\right\}\right)$$

One can now show that Theorem A.1 is slightly stronger than the corresponding tailbound shown in [6, Lemma 4].

Lemma A.6. *For all \mathbf{A} as in Theorem A.1 and $\varepsilon > 0$ there exists $c \in (0, 1/2)$ such that*

$$\min\left\{\frac{\varepsilon^2}{4C\|\mathbf{A}\|_F^2}, \frac{c\varepsilon}{2\|\mathbf{A}\|_2}\right\} > \frac{\varepsilon^2}{4(\|\mathbf{A}\|_F^2 + \varepsilon\|\mathbf{A}\|_2)}$$

Proof. Let $x = \frac{\|\mathbf{A}\|_F^2}{\varepsilon^2}$ and $y = \frac{\|\mathbf{A}\|_2}{\varepsilon}$. Hence, we need to show that there exists $c \in (0, 1/2)$ such that

$$\min\left\{\frac{1}{2Cx}, \frac{c}{y}\right\} > \frac{1}{2(x+y)} \quad (27)$$

Note that for any $z > 0$ there is $c \in (0, 1/2)$ such that $Cc = z$. Hence, choose c such that $Cc = \frac{y}{2x} \Leftrightarrow x = \frac{y}{2Cc}$. This choice will guarantee (27) since $1 > C(1 - 2c)$. \square

One also has the following version of Lemma 2.1.

Lemma A.7. *Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be symmetric with stable rank $\rho(\mathbf{A})$. Let $\text{tr}_m(\mathbf{A})$ be the stochastic trace estimator (1) with m matrix-vector multiplies with i.i.d. standard Gaussian random vectors. Let $c \in (0, \frac{1}{2})$ be arbitrary and define $C = -\frac{1}{c} - \frac{\log(1-2c)}{2c^2}$. Then, if $m \geq \frac{\log(2/\delta)}{c^2 C \rho(\mathbf{A})}$ we have that*

$$|\text{tr}_m(\mathbf{A}) - \text{tr}(\mathbf{A})| \leq 2\sqrt{C} \sqrt{\frac{\log(2/\delta)}{m}} \|\mathbf{A}\|_F \quad (28)$$

holds with probability at least $1 - \delta$.

Proof. In the setting of Corollary A.5.1 let $\varepsilon = 2\sqrt{C} \sqrt{\frac{\log(2/\delta)}{m}} \|\mathbf{A}\|_F$ and choose m sufficiently large. Proceed as in the proof of Lemma 2.1. \square

B Tight constants

In this section we prove that the smallest possible constant γ such that

$$|\mathrm{tr}_m(\mathbf{A}) - \mathrm{tr}(\mathbf{A})| \leq \gamma \sqrt{\frac{\log(2/\delta)}{m}} \|\mathbf{A}\|_F$$

holds with probability at least $1 - \delta$, is $\gamma = 2$. If we let $c \rightarrow 0$ in Lemma A.7 we note that $C \rightarrow 1$. So we expect that we have an upper bound of $\gamma = 2$. Lemma B.1 implies that $\gamma = 2$ is the lower bound, and Lemma A.7 implies that it can be asymptotically reached.

Lemma B.1. *Let \mathbf{A} be symmetric and $\gamma < 2$. Then, $\exists \delta^* > 0$ s.t. for all sufficiently large N we have*

$$\mathbb{P} \left(|\mathrm{tr}_N(\mathbf{A}) - \mathrm{tr}(\mathbf{A})| \leq \gamma \sqrt{\frac{\log(2/\delta^*)}{N}} \|\mathbf{A}\|_F \right) < 1 - \delta^*$$

For this we need the following lemma

Lemma B.2. *Let $f : (0, 1) \mapsto \mathbb{R}$ be a continuously differentiable function and $\lim_{x \rightarrow 0^+} f(x) = 1$. Suppose $\lim_{x \rightarrow 0^+} f'(x) = -\infty$. Then, $\exists \varepsilon > 0$ s.t. $x \in (0, \varepsilon) \Rightarrow f(x) < 1 - x$.*

Proof. Note that $\lim_{y \rightarrow 0^+} \int_y^x f'(t) dt = \int_0^x f'(t) dt = f(x) - 1 \Rightarrow f(x) = 1 + \int_0^x f'(t) dt$ and the integral is understood as taking the limit to 0 at the lower bound.

Since $\lim_{x \rightarrow 0^+} f'(x) = -\infty$ we know $\exists \varepsilon > 0$ s.t. $x \in (0, \varepsilon) \Rightarrow f'(x) < -2$.

Thus, $\forall x \in (0, \varepsilon)$ we have

$$\begin{aligned} 1 - x - f(x) &= 1 - x - 1 - \int_0^x f'(t) dt \\ &= - \int_0^x (f'(t) + 1) dt \\ &\geq - \int_0^x (-2 + 1) dt = \int_0^x dt = x > 0 \end{aligned}$$

as required. \square

We now proceed with proving Lemma B.1.

Proof of Lemma B.1. In fact we prove something stronger: Let S_N be the sample mean of N i.i.d. random variables with mean 0 and standard deviation σ . Then, $\exists \delta^* > 0$ such that for all sufficiently large N we have

$$\mathbb{P} \left(|S_N| \leq \gamma \sqrt{\frac{\log(2/\delta^*)}{N}} \frac{\sigma}{\sqrt{2}} \right) < 1 - \delta^* \quad (29)$$

The result in Lemma B.2 immediately follows from (29). Define

$$p_N = \mathbb{P} \left(|S_N| \leq \gamma \sqrt{\frac{\log(2/\delta)}{N}} \frac{\sigma}{\sqrt{2}} \right)$$

By the Central Limit Theorem we have

$$p_N \rightarrow \Phi \left(\frac{\gamma \sqrt{\log(2/\delta)} \sigma}{\sqrt{2} \sigma} \right) - \Phi \left(-\frac{\gamma \sqrt{\log(2/\delta)} \sigma}{\sqrt{2} \sigma} \right) = \operatorname{erf} \left(\frac{\gamma}{2} \sqrt{\log(2/\delta)} \right)$$

as $N \rightarrow \infty$, where Φ is the cumulative distribution function of $N(0, 1)$. Let $\nu = \frac{\gamma}{2} < 1$ and $p(\delta) := \operatorname{erf}(\frac{\gamma}{2} \sqrt{\log(2/\delta)})$. We will now show that $\exists \delta^* \in (0, 1/2)$ such that

$$p(\delta) < 1 - \delta^*$$

It is easy to see that

$$\lim_{\delta \rightarrow 0^+} p(\delta) = 1 = 1 - \delta|_{\delta=0}$$

For $\delta > 0$ we have

$$\begin{aligned} p'(\delta) &= -\frac{\nu}{\sqrt{\pi}} \frac{\exp(-\nu^2 \log(2/\delta))}{\delta \sqrt{\log(2/\delta)}} \\ &= -\frac{\nu}{\sqrt{\pi}} \left(\frac{\delta}{2} \right)^{\nu^2} \frac{1}{\delta \sqrt{\log(2/\delta)}} \\ &= -\frac{\nu}{2^{\nu^2} \sqrt{\pi}} \delta^{\nu^2-1} \frac{1}{\sqrt{\log(2/\delta)}} \\ &= -\frac{\nu}{2^{\nu^2} \sqrt{\pi}} \frac{1}{\delta^\beta \sqrt{\log(2/\delta)}} \end{aligned}$$

where $\beta = 1 - \nu^2$. Since $\nu < 1$ we have $\beta > 0$ which implies that $\delta^\beta \sqrt{\log(2/\delta)} \rightarrow 0$ as $\delta \rightarrow 0^+$ and therefore $\lim_{\delta \rightarrow 0^+} p'(\delta) = -\infty$. By Lemma B.2, there is a neighbourhood $(0, \varepsilon)$ s.t. $p(\delta) < 1 - \delta$ whenever $\delta \in (0, \varepsilon)$. Thus, there is a $\delta^* \in (0, \varepsilon)$ such that $p(\delta^*) < 1 - \delta^*$. Since, $\lim_{N \rightarrow +\infty} p_N(\delta^*) = p(\delta^*) \exists M \in \mathbb{N}$ s.t. $N > M \Rightarrow p_N(\delta^*) < 1 - \delta^*$. Then choose any $N > M$ will be sufficiently large. \square

References

- [1] R. H. Affandi, E. Fox, R. Adams, and B. Taskar. Learning the parameters of determinantal point process kernels. In *International Conference on Machine Learning*, pages 1224–1232. PMLR, 2014.

- [2] H. Avron. Counting triangles in large graphs using randomized matrix trace estimation. In *Workshop on Large-scale Data Mining: Theory and Applications*, volume 10, pages 10–9, 2010.
- [3] H. Avron and S. Toledo. Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. *J. ACM*, 58(2):Art. 8, 17, 2011.
- [4] Z. Bai, M. Fahey, and G. Golub. Some large-scale matrix computation problems. *J. Comput. Appl. Math.*, 74(1-2):71–89, 1996.
- [5] Z. Bujanović and D. Kressner. Norm and trace estimation with random rank-one vectors. *SIAM J. Matrix Anal. Appl.*, 42(1):202–223, 2021.
- [6] A. Cortinovis and D. Kressner. On randomized trace estimates for indefinite matrices with an application to determinants. *Found. Comput. Math.*, 2021.
- [7] T. A. Davis and Y. Hu. The University of Florida sparse matrix collection. *ACM Trans. Math. Software*, 38(1):Art. 1, 25, 2011.
- [8] J. A. de la Peña, I. Gutman, and J. Rada. Estimating the Estrada index. *Linear Algebra Appl.*, 427(1):70–76, 2007.
- [9] E. Estrada. Characterization of 3D molecular structure. *Chemical Physics Letters*, 319(5-6):713–718, 2000.
- [10] E. Estrada and D. J. Higham. Network properties revealed through matrix functions. *SIAM Rev.*, 52(4):696–714, 2010.
- [11] A. Frommer, C. Schimmel, and M. Schweitzer. Analysis of Probing Techniques for Sparse Approximation and Trace Estimation of Decaying Matrix Functions. *SIAM J. Matrix Anal. Appl.*, 42(3):1290–1318, 2021.
- [12] A. S. Gambhir, A. Stathopoulos, and K. Orginos. Deflation as a method of variance reduction for estimating the trace of a matrix inverse. *SIAM J. Sci. Comput.*, 39(2):A532–A558, 2017.
- [13] A. Gittens and M. W. Mahoney. Revisiting the Nyström method for improved large-scale machine learning. *J. Mach. Learn. Res.*, 17:Paper No. 117, 65, 2016.
- [14] S. Gratton and D. Tittley-Peloquin. Improved bounds for small-sample estimation. *SIAM J. Matrix Anal. Appl.*, 39(2):922–931, 2018.
- [15] N. Halko, P.-G. Martinsson, and J. A. Tropp. Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.*, 53(2):217–288, 2011.
- [16] N. J. Higham. *Functions of matrices*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2008. Theory and computation.

- [17] M. Hochbruck and C. Lubich. On Krylov subspace approximations to the matrix exponential operator. *SIAM J. Numer. Anal.*, 34(5):1911–1925, 1997.
- [18] M. F. Hutchinson. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Comm. Statist. Simulation Comput.*, 18(3):1059–1076, 1989.
- [19] S. Jiang, H. Pham, D. Woodruff, and R. Zhang. Optimal sketching for trace estimation. *Advances in Neural Information Processing Systems*, 34, 2021.
- [20] L. Lin. Randomized estimation of spectral densities of large matrices made accurate. *Numer. Math.*, 136(1):183–213, 2017.
- [21] P.-G. Martinsson and J. A. Tropp. Randomized numerical linear algebra: foundations and algorithms. *Acta Numer.*, 29:403–572, 2020.
- [22] R. A. Meyer. Updates for hutch++. <https://ram900.hosting.nyu.edu/hutchplusplus/#nystroulm-hutch>. Accessed: 3 February 2022.
- [23] R. A. Meyer, C. Musco, C. Musco, and D. P. Woodruff. Hutch++: Optimal stochastic trace estimation. In *Symposium on Simplicity in Algorithms (SOSA)*, pages 142–155. SIAM, 2021.
- [24] Y. Nakatsukasa. Fast and stable randomized low-rank matrix approximation. *arXiv preprint arXiv:2009.11392*, 2020.
- [25] F. Roosta-Khorasani and U. Ascher. Improved bounds on sample size for implicit matrix trace estimators. *Found. Comput. Math.*, 15(5):1187–1212, 2015.
- [26] F. Roosta-Khorasani, G. J. Székely, and U. M. Ascher. Assessing stochastic algorithms for large scale nonlinear least squares problems using extremal probabilities of linear combinations of gamma random variables. *SIAM/ASA J. Uncertain. Quantif.*, 3(1):61–90, 2015.
- [27] A. K. Saibaba, A. Alexanderian, and I. C. F. Ipsen. Randomized matrix-free trace and log-determinant estimators. *Numer. Math.*, 137(2):353–395, 2017.
- [28] D. C. Sorensen and M. Embree. A DEIM induced CUR factorization. *SIAM J. Sci. Comput.*, 38(3):A1454–A1482, 2016.
- [29] C. Thron, S. J. Dong, K. F. Liu, and H. P. Ying. Padé- Z_2 estimator of determinants. *Physical Review D*, 57(3):1642, 1998.

- [30] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher. Fixed-rank approximation of a positive-semidefinite matrix from streaming data. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1225–1234. Curran Associates, Inc., 2017.
- [31] S. Ubaru and Y. Saad. Applications of trace estimation techniques. In *International Conference on High Performance Computing in Science and Engineering*, pages 19–33. Springer, 2017.
- [32] M. J. Wainwright. *High-dimensional statistics*, volume 48 of *Cambridge Series in Statistical and Probabilistic Mathematics*. Cambridge University Press, Cambridge, 2019. A non-asymptotic viewpoint.
- [33] M. J. Wainwright and M. I. Jordan. Log-determinant relaxation for approximate inference in discrete Markov random fields. *IEEE Trans. Signal Process.*, 54(6):2099–2109, 2006.
- [34] L. Wu, J. Laeuchli, V. Kalantzis, A. Stathopoulos, and E. Gallopoulos. Estimating the trace of the matrix inverse by interpolating from the diagonal of an approximate inverse. *J. Comput. Phys.*, 326:828–844, 2016.