Hutch++: Optimal Stochastic Trace Estimation

Raphael A. Meyer* Cameron Musco† Christopher Musco* David P. Woodruff‡

Abstract

We study the problem of estimating the trace of a matrix $\boldsymbol{A}$ that can only be accessed through matrix-vector multiplication. We introduce a new randomized algorithm, Hutch++, which computes a $(1\pm\varepsilon)$ approximation to $\operatorname{tr}(\boldsymbol{A})$ for any positive semidefinite (PSD) $\boldsymbol{A}$ using just $O(1/\varepsilon)$ matrix-vector products. This improves on the ubiquitous *Hutchinson's estimator*, which requires $O(1/\varepsilon^2)$ matrix-vector products. Our approach is based on a simple technique for reducing the variance of Hutchinson's estimator using a low-rank approximation step, and is easy to implement and analyze. Moreover, we prove that, up to a logarithmic factor, the complexity of Hutch++ is optimal amongst all matrix-vector query algorithms, even when queries can be chosen adaptively. We show that it significantly outperforms Hutchinson's method in experiments. While our theory requires $\boldsymbol{A}$ to be positive semidefinite, empirical gains extend to applications involving non-PSD matrices, such as triangle estimation in networks.

1 Introduction

A ubiquitous problem in numerical linear algebra is that of approximating the trace of a $d \times d$ matrix $\boldsymbol{A}$ that can only be accessed via matrix-vector multiplication queries. In other words, we are given access to an oracle that can evaluate $\boldsymbol{A}\mathbf{x}$ for any $\mathbf{x} \in \mathbb{R}^d$, and the goal is to return an approximation to $\operatorname{tr}(\boldsymbol{A})$ using as few queries to this oracle as possible. An exact solution can be obtained with d queries because $\operatorname{tr}(\boldsymbol{A}) = \sum_{i=1}^{d} \boldsymbol{A}_{ii} = \sum_{i=1}^{d} \mathbf{e}_i^T \boldsymbol{A}\mathbf{e}_i$, where $\mathbf{e}_i$ denotes the i^{th} standard basis vector. The goal is thus to develop algorithms that use far fewer than d matrix-vector multiplications.

Known as *implicit* or *matrix free* trace estimation, this problem arises in applications that require the trace of a matrix $\boldsymbol{A}$, where $\boldsymbol{A}$ is itself a transformation of some other matrix $\boldsymbol{B}$. For example, $\boldsymbol{A} = \boldsymbol{B}^q$, $\boldsymbol{A} = \boldsymbol{B}^{-1}$, or $\boldsymbol{A} = \exp(\boldsymbol{B})$. In all of these cases, explicitly computing $\boldsymbol{A}$ would require roughly $O(d^3)$ time, whereas multiplication with a vector $\mathbf{x}$ can be implemented more quickly using iterative methods. For example, $\boldsymbol{B}^q\mathbf{x}$ can be computed in just $O(d^2)$ time for constant q, and for well-conditioned matrices, $\boldsymbol{B}^{-1}\mathbf{x}$ and $\exp(\boldsymbol{B})\mathbf{x}$ can also be computed in $O(d^2)$ time using the conjugate gradient or Lanczos methods [Hig08]. Implicit trace estimation is used to approximate matrix norms [HMAS17, MNS+18], spectral densities [LSY16, CKSV18, BKKS20], log-determinants [BDKZ15, HMS15], the Estrada index [US18, WSMB20], eigenvalue counts in intervals [DNPS16], triangle counts in graphs [Avr10], and much more [Che16, LSTZ20]. In these applications, we typically have that $\boldsymbol{A}$ is symmetric, and often positive semidefinite (PSD).

1.1 Hutchinson's Estimator

The most common method for implicit trace estimation is Hutchinson's stochastic estimator [Hut90]. This elegant randomized algorithm works as follows: let $\boldsymbol{G} = [\mathbf{g}_1, \ldots, \mathbf{g}_m] \in \mathbb{R}^{d\times m}$ be a matrix containing i.i.d. random variables with mean 0 and variance 1. A simple calculation shows that $\mathbb{E}[\mathbf{g}_i^T \boldsymbol{A}\mathbf{g}_i] = \operatorname{tr}(\boldsymbol{A})$ for each $\mathbf{g}_i \in \mathbb{R}^d$, and $\mathbf{g}_i^T \boldsymbol{A}\mathbf{g}_i$ can be computed with just one matrix-vector multiplication. So to approximate $\operatorname{tr}(\boldsymbol{A})$, Hutchinson's estimator returns the following average:

Hutchinson's Estimator:

$$\mathrm{H}_m(\boldsymbol{A}) = \frac{1}{m}\sum_{i=1}^{m} \mathbf{g}_i^T \boldsymbol{A}\mathbf{g}_i = \frac{1}{m}\operatorname{tr}(\boldsymbol{G}^T\boldsymbol{A}\boldsymbol{G}). \tag{1.1}$$

Hutchinson's original work suggests using random ± 1 sign vectors for $\mathbf{g}_1, \ldots, \mathbf{g}_m$, and an earlier paper by Girard suggests standard normal random variables [Gir87]. Both choices perform similarly, as both random variables are sub-Gaussian. For vectors with sub-Gaussian random entries, it can be proven that, when $\boldsymbol{A}$ is positive semidefinite, $(1-\varepsilon)\operatorname{tr}(\boldsymbol{A}) \le \mathrm{H}_m(\boldsymbol{A}) \le (1+\varepsilon)\operatorname{tr}(\boldsymbol{A})$ with probability $\ge 1-\delta$ if we use $m = O\left(\log(1/\delta)/\varepsilon^2\right)$ matrix-vector multiplication queries [AT11, RA15].[1] For constant δ (e.g., $\delta = 1/10$) the bound is $O(1/\varepsilon^2)$.

1.2 Our results

Since Hutchinson's work, and the non-asymptotic analysis in [AT11], there has been no improvement on this $O(1/\varepsilon^2)$ matrix-vector multiplica-

*New York University
†Univeristy of Massachusetts Amherst
‡Carnegie Mellon University

[1] For non-PSD matrices, this generalizes to $\operatorname{tr}(\boldsymbol{A}) - \varepsilon\|\boldsymbol{A}\|_F \le \mathrm{H}_m(\boldsymbol{A}) \le \operatorname{tr}(\boldsymbol{A}) + \varepsilon\|\boldsymbol{A}\|_F$, which implies the relative error bound since when $\boldsymbol{A}$ is PSD, $\|\boldsymbol{A}\|_F \le \operatorname{tr}(\boldsymbol{A})$.

tion bound for trace approximation. Our main contribution is a quadratic improvement: we provide a new algorithm, Hutch++, that obtains the same $(1\pm\varepsilon)$ guarantee with $O(1/\varepsilon)$ matrix-vector multiplication queries. This algorithm is nearly as simple as the original Hutchinson's method, and can be implemented in just a few lines of code.

Algorithm 1 Hutch++

input: Matrix-vector multiplication oracle for PSD matrix $\boldsymbol{A} \in \mathbb{R}^{d\times d}$. Number m of queries.

output: Approximation to $\operatorname{tr}(\boldsymbol{A})$.

1: Sample $\boldsymbol{S} \in \mathbb{R}^{d\times \frac{m}{3}}$ and $\boldsymbol{G} \in \mathbb{R}^{d\times \frac{m}{3}}$ with i.i.d. $\{+1,-1\}$ entries.
2: Compute an orthonormal basis $\boldsymbol{Q} \in \mathbb{R}^{d\times \frac{m}{3}}$ for the span of $\boldsymbol{AS}$ (e.g., via QR decomposition).
3: **return** Hutch++$(\boldsymbol{A}) = \operatorname{tr}(\boldsymbol{Q}^T\boldsymbol{AQ}) + \frac{3}{m}\operatorname{tr}(\boldsymbol{G}^T(\boldsymbol{I}-\boldsymbol{QQ}^T)\boldsymbol{A}(\boldsymbol{I}-\boldsymbol{QQ}^T)\boldsymbol{G})$.

Hutch++ requires m matrix-vector multiplications with $\boldsymbol{A}$: $m/3$ to compute $\boldsymbol{A}\cdot\boldsymbol{S}$, $m/3$ to compute $\boldsymbol{A}\cdot\boldsymbol{Q}$, and $m/3$ to compute $\boldsymbol{A}\cdot(\boldsymbol{I}-\boldsymbol{QQ}^T)\boldsymbol{G}$. It requires $O(dm^2)$ additional runtime to compute the basis $\boldsymbol{Q}$ and the product $(\boldsymbol{I}-\boldsymbol{QQ}^T)\boldsymbol{G} = \boldsymbol{G}-\boldsymbol{QQ}^T\boldsymbol{G}$. For concreteness, we state the method with random sign matrices, but the entries of $\boldsymbol{S}$ and $\boldsymbol{G}$ can be any sub-Gaussian random variables with mean 0 and variance 1, including e.g., standard Gaussians. Our main theorem on Hutch++ is:

THEOREM 1.1. *If Hutch++ is implemented with $m = O(\sqrt{\log(1/\delta)}/\varepsilon + \log(1/\delta))$ matrix-vector multiplication queries, then for any PSD $\boldsymbol{A}$, with probability $\geq 1-\delta$, the output Hutch++$(\boldsymbol{A})$ satisfies:*

$$(1-\varepsilon)\operatorname{tr}(\boldsymbol{A}) \leq \textit{Hutch++}(\boldsymbol{A}) \leq (1+\varepsilon)\operatorname{tr}(\boldsymbol{A}).$$

Hutch++ can be viewed as a natural *variance reduced* version of Hutchinson's estimator. The method starts by computing an orthonormal span $\boldsymbol{Q} \in \mathbb{R}^{d\times\frac{m}{3}}$ by running a single iteration of power method with a random start matrix $\boldsymbol{S}$. $\boldsymbol{Q}$ coarsely approximates the span of $\boldsymbol{A}$'s top eigenvectors. Then we separate $\boldsymbol{A}$ into its projection onto the subspace spanned by $\boldsymbol{Q}$, and onto that subspace's orthogonal compliment, writing $\operatorname{tr}(\boldsymbol{A}) = \operatorname{tr}(\boldsymbol{QQ}^T\boldsymbol{AQQ}^T) + \operatorname{tr}\left((\boldsymbol{I}-\boldsymbol{QQ}^T)\boldsymbol{A}(\boldsymbol{I}-\boldsymbol{QQ}^T)\right)$. By the cyclic property of the trace, the first term is equal to $\operatorname{tr}(\boldsymbol{Q}^T\boldsymbol{AQ})$, which is computed *exactly* by Hutch++ with $\frac{m}{3}$ matrix-vector multiplications. The second term is approximated using Hutchinson's estimator with the random vectors in $\boldsymbol{G}$.

Thus, the error in estimating $\operatorname{tr}(\boldsymbol{A})$ is entirely due to approximating this second term. The key observation is that the variance when estimating this term is much lower than when estimating $\operatorname{tr}(\boldsymbol{A})$ directly. Specifically, it is proportional to $\|(\boldsymbol{I}-\boldsymbol{QQ}^T)\boldsymbol{A}(\boldsymbol{I}-\boldsymbol{QQ}^T)\|_F^2$, which, using standard tools from randomized linear algebra [CEM+15, Woo14], we can show is bounded by $\varepsilon\operatorname{tr}(\boldsymbol{A})^2$ with good probability when $m = O(1/\varepsilon)$. This yields our improvement over Hutchinson's method applied directly to $\boldsymbol{A}$, which has variance bounded by $\operatorname{tr}(\boldsymbol{A})^2$. The full proof of Theorem 1.1 is in Section 3.

Algorithm 1 is *adaptive*: it multiplies $\boldsymbol{A}$ by a sequence of query vectors $\mathbf{r}_1,\ldots,\mathbf{r}_m$, where later queries depend on earlier ones. In contrast, Hutchinson's method is *non-adaptive*: $\mathbf{r}_1,\ldots,\mathbf{r}_m$ are chosen in advance, before computing any of the products $\boldsymbol{Ar}_1,\ldots,\boldsymbol{Ar}_m$. In addition to Algorithm 1, we give a non-adaptive variant of Hutch++ that obtains the same $O(1/\varepsilon)$ bound. We complement these results with a nearly matching lower bound, proven in Section 4. Specifically, via a reduction from the Gap-Hamming problem from communication complexity, we show that any matrix-vector query algorithm whose queries have bounded bit complexity requires $m = \Omega\left(\frac{1}{\varepsilon\log(1/\varepsilon)}\right)$ queries to estimate the trace of a PSD matrix up to a $(1\pm\varepsilon)$ multiplicative approximation. We also prove a tight $m = \Omega\left(\frac{1}{\varepsilon}\right)$ lower bound for non-adaptive algorithms in the real RAM model of computation.

Empirical Results. In Section 5 we complement our theoretical results with experiments on synthetic and real-world matrices, including applications of trace estimation to approximating log determinants, the graph Estrada index, and the number of triangles in a graph. We demonstrate that Hutch++ improves substantially on Hutchinson's estimator, and on related estimators based on approximating the top eigenvalues of $\boldsymbol{A}$. While our theory applies to positive semidefinite matrices, Hutch++ can be applied unmodified to non-positive semidefinite trace estimation, and continues to perform very well empirically. We note that Hutch++ is simple to implement and essentially parameter free – the only choice needed is the number of matrix-vector multiplication queries m.

1.3 Prior Work Upper bounds.

A nearly tight non-asymptotic analysis of Hutchinson's estimator for positive semidefinite matrices was given by Avron and Toledo using an approach based on reducing to Johnson-Lindenstrauss random projection [AT11, DG03, Ach03]. A slightly tighter approach from [RA15] obtains a $(1\pm\varepsilon)$ multiplicative error bound with $m = O(1/\varepsilon^2)$ matrix-vector multiplication queries. This bound is what we improve on with Hutch++.

A number of papers suggest variance reduction schemes for Hutchinson's estimator. Some take advan-

tage of sparsity structure in $\boldsymbol{A}$ [TS11, SLO13] and others use a "decomposition" approach similar to Hutch++ [APJ+18]. Most related to our work are two papers which, like Hutch++, perform the decomposition by projecting onto some $\boldsymbol{Q}$ that approximately spans $\boldsymbol{A}$'s top eigenspace [GSO17, Lin17]. The justification is that this method should perform much better than Hutchinson's when $\boldsymbol{A}$ is close to low-rank, because $\mathrm{tr}(\boldsymbol{Q}^T\boldsymbol{A}\boldsymbol{Q})$ will capture most of $\boldsymbol{A}$'s trace. Our contribution is an analysis of this approach which 1) improves on Hutchinson's *even when* $\boldsymbol{A}$ *is far from low-rank* and 2) shows that a very coarse approximation to the top eigenvectors suffices (computed using one iteration of the power method). Finally, we note two papers which directly use the approximation $\mathrm{tr}(\boldsymbol{A}) \approx \mathrm{tr}(\boldsymbol{Q}^T\boldsymbol{A}\boldsymbol{Q})$, where $\boldsymbol{Q}$ is computed with a randomized SVD method [SAI17, HL20]. Of course, this approach works best for nearly-low rank matrices.

Lower bounds. Our lower bounds extend a recent line of work on lower bounds for linear algebra problems in the "matrix-vector query model" [SEAR18, SWYZ19, BHSW20]. [WWZ14] proves a lower bound of $\Omega(1/\varepsilon^2)$ queries for PSD trace approximation in an alternative model that allows for adaptive "quadratic form" queries: $\mathbf{r}_1^T\boldsymbol{A}\mathbf{r}_1, \ldots, \mathbf{r}_m^T\boldsymbol{A}\mathbf{r}_m$. This model captures Hutchinson's estimator, but not Hutch++, which is why we are able to obtain an upper bound of $O(1/\varepsilon)$ queries.

2 Preliminaries

Notation. For $\mathbf{a} \in \mathbb{R}^d$, $\|\mathbf{a}\|_2 = (\sum_{i=1}^d a_i^2)^{1/2}$ denotes the ℓ_2 norm and $\|\mathbf{a}\|_1 = \sum_{i=1}^d |a_i|$ denotes the ℓ_1 norm. For $\boldsymbol{A} \in \mathbb{R}^{n\times d}$, $\|\boldsymbol{A}\|_F = (\sum_{i=1}^n \sum_{j=1}^d \boldsymbol{A}_{ij}^2)^{1/2}$ denotes the Frobenius norm. For square $\boldsymbol{A} \in \mathbb{R}^{d\times d}$, $\mathrm{tr}(\boldsymbol{A}) = \sum_{i=1}^d \boldsymbol{A}_{ii}$ denotes the trace. Our main results on trace approximation are proven for symmetric positive semidefinite (PSD) matrices, which are the focus of many applications. Any symmetric $\boldsymbol{A} \in \mathbb{R}^{d\times d}$ has eigendecomposition $\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T$, where $\boldsymbol{V} \in \mathbb{R}^{d\times d}$ is orthogonal and $\boldsymbol{\Lambda}$ is a real-valued diagonal matrix. We let $\boldsymbol{\lambda} = \mathrm{diag}(\boldsymbol{\Lambda})$ be a vector containing $\boldsymbol{A}$'s eigenvalues in descending order: $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$. When $\boldsymbol{A}$ is PSD, $\lambda_i \geq 0$ for all i. We use the identities $\mathrm{tr}(\boldsymbol{A}) = \|\boldsymbol{\lambda}\|_1$ and $\|\boldsymbol{A}\|_F = \|\boldsymbol{\lambda}\|_2$. We let $\boldsymbol{A}_k = \arg\min_{\boldsymbol{B},\mathrm{rank}(\boldsymbol{B})=k} \|\boldsymbol{A}-\boldsymbol{B}\|_F$ denote the optimal k-rank approximation to $\boldsymbol{A}$. For a PSD matrix $\boldsymbol{A}$, $\boldsymbol{A}_k = \boldsymbol{V}_k\boldsymbol{\Lambda}_k\boldsymbol{V}_k^T$, where $\boldsymbol{V}_k \in \mathbb{R}^{d\times k}$ contains the first k columns of $\boldsymbol{V}$ and $\boldsymbol{\Lambda}_k$ is the $k \times k$ top left submatrix of $\boldsymbol{\Lambda}$.

Hutchinson's Analysis. We require a standard bound on the accuracy of Hutchinson's estimator:

LEMMA 2.1. *Let* $\boldsymbol{A} \in \mathbb{R}^{d\times d}$, $\delta \in (0, 1/2]$, $\ell \in \mathbb{N}$. *Let* $\mathrm{H}_\ell(\boldsymbol{A})$ *be the* ℓ*-query Hutchinson estimator defined in* (1.1)*, implemented with mean 0, i.i.d. sub-Gaussian random variables with constant sub-Gaussian parameter. For fixed constants* c, C*, if* $\ell > c\log(1/\delta)$*, then with probability* $\geq 1-\delta$*,*

$$|\mathrm{H}_\ell(\boldsymbol{A}) - \mathrm{tr}(\boldsymbol{A})| \leq C\sqrt{\frac{\log(1/\delta)}{\ell}}\|\boldsymbol{A}\|_F.$$

So, if $\ell = O\left(\frac{\log(1/\delta)}{\varepsilon^2}\right)$ *then, with probability* $\geq 1-\delta$*,* $|\mathrm{H}_\ell(\boldsymbol{A}) - \mathrm{tr}(\boldsymbol{A})| \leq \varepsilon\|\boldsymbol{A}\|_F$.

We refer the reader to [RV+13] for a formal definition of sub-Gaussian random variables: both normal $\mathcal{N}(0,1)$ random variables and ± 1 random variables are sub-Gaussian with constant parameter. Lemma 2.1 is proven in Appendix A for completeness. It is slightly more general than prior work [RA15] in that it applies to non-PSD, and even asymmetric matrices, which will be important in the analysis of our non-adaptive algorithm. A similar result was recently shown in [CK20].

3 Complexity Analysis

We start by providing the technical intuition behind Hutch++. First note that, for a PSD matrix with eigenvalues $\boldsymbol{\lambda}$, $\|\boldsymbol{A}\|_F \leq \mathrm{tr}(\boldsymbol{A})$, so Lemma 2.1 immediately implies that Hutchinson's estimator obtains a relative error guarantee with $O(1/\varepsilon^2)$ queries. However, this bound is only tight when $\|\boldsymbol{\lambda}\|_2 \approx \|\boldsymbol{\lambda}\|_1$, i.e., when $\boldsymbol{A}$ has *significant mass concentrated on just a small number of eigenvalues.*

Hutch++ simply eliminates this possibility by approximately projecting off $\boldsymbol{A}$'s large eigenvalues using a projection $\boldsymbol{Q}\boldsymbol{Q}^T$. By doing so, it only needs to compute a stochastic estimate for the trace of $(\boldsymbol{I}-\boldsymbol{Q}\boldsymbol{Q}^T)\boldsymbol{A}(\boldsymbol{I}-\boldsymbol{Q}\boldsymbol{Q}^T)$. The error of this estimate is proportional to $\|(\boldsymbol{I}-\boldsymbol{Q}\boldsymbol{Q}^T)\boldsymbol{A}(\boldsymbol{I}-\boldsymbol{Q}\boldsymbol{Q}^T)\|_F$, which we show is *always much smaller than* $\mathrm{tr}(\boldsymbol{A})$. In particular, suppose that $\boldsymbol{Q} = \boldsymbol{V}_k$ exactly spanned the top k eigenvectors $\boldsymbol{A}$ and thus $(\boldsymbol{I}-\boldsymbol{Q}\boldsymbol{Q}^T)\boldsymbol{A}(\boldsymbol{I}-\boldsymbol{Q}\boldsymbol{Q}^T) = \boldsymbol{A}-\boldsymbol{A}_k$. Then we have:

LEMMA 3.1. *Let* $\boldsymbol{A}_k$ *be the best rank-*k *approximation to PSD matrix* $\boldsymbol{A}$*. Then,* $\|\boldsymbol{A}-\boldsymbol{A}_k\|_F \leq \frac{1}{\sqrt{k}}\mathrm{tr}(\boldsymbol{A})$.

Proof. We have $\lambda_{k+1} \leq \frac{1}{k}\sum_{i=1}^k \lambda_i \leq \frac{1}{k}\mathrm{tr}(\boldsymbol{A})$, so:

$$\|\boldsymbol{A}-\boldsymbol{A}_k\|_F^2 = \sum_{i=k+1}^d \lambda_i^2 \leq \lambda_{k+1}\sum_{i=k+1}^d \lambda_i \leq \frac{1}{k}\mathrm{tr}(\boldsymbol{A})\sum_{i=k+1}^d \lambda_i \leq \frac{1}{k}\mathrm{tr}(\boldsymbol{A})^2.$$

∎

This result immediately suggests the possibility of an algorithm with $O(1/\varepsilon)$ query complexity: Set $k = O(1/\varepsilon)$ and split $\operatorname{tr}(\boldsymbol{A}) = \operatorname{tr}(\boldsymbol{A}_k) + \operatorname{tr}(\boldsymbol{A} - \boldsymbol{A}_k)$. The first term can be computed exactly with $O(1/\varepsilon)$ matrix-vector multiplication queries if $\boldsymbol{V}_k$ is known, since $\operatorname{tr}(\boldsymbol{A}_k) = \operatorname{tr}(\boldsymbol{V}_k^T \boldsymbol{A} \boldsymbol{V}_k)$. By Lemma 3.1 combined with Lemma 2.1, the second can be estimated to error $\pm\varepsilon \operatorname{tr}(\boldsymbol{A})$ using just $O(1/\varepsilon)$ queries instead of $O(1/\varepsilon^2)$. Of course, we can't compute $\boldsymbol{V}_k$ exactly with a small number of matrix-vector multiplication queries, but this is easily resolved by using an approximate projection. Using standard tools from randomized linear algebra, $O(k)$ queries suffices to find a $\boldsymbol{Q}$ with $\|(\boldsymbol{I} - \boldsymbol{Q}\boldsymbol{Q}^T)\boldsymbol{A}(\boldsymbol{I} - \boldsymbol{Q}\boldsymbol{Q}^T)\|_F \leq O(\|\boldsymbol{A} - \boldsymbol{A}_k\|_F)$, which is all that is needed for a $O(1/\varepsilon)$ query result.

Concretely, we use Lemma 3.1 to prove the following general theorem, from which Theorem 1.1 and our non-adaptive algorithmic result will follow as direct corollaries.

THEOREM 3.1. *Let $\boldsymbol{A} \in \mathbb{R}^{d\times d}$ be PSD, $\delta \in (0, 1/2)$, $\ell \in \mathbb{N}$, $k \in \mathbb{N}$. Let $\tilde{\boldsymbol{A}}$ and $\boldsymbol{\Delta}$ be any matrices with:*

$$\operatorname{tr}(\boldsymbol{A}) = \operatorname{tr}(\tilde{\boldsymbol{A}}) + \operatorname{tr}(\boldsymbol{\Delta}) \quad \text{and} \quad \|\boldsymbol{\Delta}\|_F \leq 2\|\boldsymbol{A} - \boldsymbol{A}_k\|_F.$$

For fixed constants c, C, if $\ell > c\log(1/\delta)$, then with probability $1-\delta$, $Z = \left[\operatorname{tr}(\tilde{\boldsymbol{A}}) + \mathrm{H}_\ell(\boldsymbol{\Delta})\right]$ satisfies:

$$\left|Z - \operatorname{tr}(\boldsymbol{A})\right| \leq 2C\sqrt{\tfrac{\log(1/\delta)}{k\ell}} \cdot \operatorname{tr}(\boldsymbol{A}).$$

In particular, if $k = \ell = O\left(\frac{\sqrt{\log(1/\delta)}}{\varepsilon} + \log(1/\delta)\right)$, Z is a $(1\pm\varepsilon)$ error approximation to $\operatorname{tr}(\boldsymbol{A})$.

Proof. We have with probability $\geq 1-\delta$:

(since $Z = \operatorname{tr}(\tilde{\boldsymbol{A}}) + \mathrm{H}_\ell(\boldsymbol{\Delta})$ and $\operatorname{tr}(\boldsymbol{A}) = \operatorname{tr}(\tilde{\boldsymbol{A}}) + \operatorname{tr}(\boldsymbol{\Delta})$)
$$|Z - \operatorname{tr}(\boldsymbol{A})| = |\mathrm{H}_\ell(\boldsymbol{\Delta}) - \operatorname{tr}(\boldsymbol{\Delta})|$$
(by the standard Hutchinson's analysis, Lemma 2.1)
$$\leq C\sqrt{\tfrac{\log(1/\delta)}{\ell}}\|\boldsymbol{\Delta}\|_F$$
(by the assumption that $\|\boldsymbol{\Delta}\|_F \leq 2\|\boldsymbol{A} - \boldsymbol{A}_k\|_F$)
$$\leq 2C\sqrt{\tfrac{\log(1/\delta)}{\ell}}\|\boldsymbol{A} - \boldsymbol{A}_k\|_F$$
(by Lemma 3.1)
$$\leq 2C\sqrt{\tfrac{\log(1/\delta)}{k\ell}}\operatorname{tr}(\boldsymbol{A}).$$
∎

As discussed, Theorem 3.1 would immediately yield an $O(1/\varepsilon)$ query algorithm if we knew an optimal k-rank approximation for $\boldsymbol{A}$. Since computing one is infeasible, our first version of Hutch++ (Algorithm 1) instead uses a projection onto a subspace $\boldsymbol{Q}$ which is computed with one iteration of the power method. We have:

THEOREM 1.1 RESTATED 1. *If Algorithm 1 is implemented with $m = O(\sqrt{\log(1/\delta)}/\varepsilon + \log(1/\delta))$ matrix-vector multiplication queries, then for any PSD $\boldsymbol{A}$, with probability $\geq 1-\delta$, the output Hutch++$(\boldsymbol{A})$ satisfies: $(1-\varepsilon)\operatorname{tr}(\boldsymbol{A}) \leq$ Hutch++$(\boldsymbol{A}) \leq (1+\varepsilon)\operatorname{tr}(\boldsymbol{A})$.*

Proof. Let $\boldsymbol{S}$, $\boldsymbol{G}$, and $\boldsymbol{Q}$ be as in Algorithm 1. We instantiate Theorem 3.1 with $\tilde{\boldsymbol{A}} = \boldsymbol{Q}^T\boldsymbol{A}\boldsymbol{Q}$ and $\boldsymbol{\Delta} = (\boldsymbol{I} - \boldsymbol{Q}\boldsymbol{Q}^T)\boldsymbol{A}(\boldsymbol{I} - \boldsymbol{Q}\boldsymbol{Q}^T)$. Note that, since $\boldsymbol{Q}$ is orthogonal, $(\boldsymbol{I} - \boldsymbol{Q}\boldsymbol{Q}^T)$ is a projection matrix, so $(\boldsymbol{I} - \boldsymbol{Q}\boldsymbol{Q}^T) = (\boldsymbol{I} - \boldsymbol{Q}\boldsymbol{Q}^T)^2$. This fact, along with the cyclic property of the trace, gives:

$$\operatorname{tr}(\tilde{\boldsymbol{A}}) = \operatorname{tr}(\boldsymbol{A}\boldsymbol{Q}\boldsymbol{Q}^T) \quad \text{and} \quad \operatorname{tr}(\boldsymbol{\Delta}) = \operatorname{tr}(\boldsymbol{A}(\boldsymbol{I} - \boldsymbol{Q}\boldsymbol{Q}^T)),$$

and thus $\operatorname{tr}(\tilde{\boldsymbol{A}}) + \operatorname{tr}(\boldsymbol{\Delta}) = \operatorname{tr}(\boldsymbol{A})$ as required by Theorem 3.1. Furthermore, since multiplying by a projection matrix can only decrease Frobenius norm, $\|\boldsymbol{\Delta}\|_F^2 \leq \|\boldsymbol{A}(\boldsymbol{I} - \boldsymbol{Q}\boldsymbol{Q}^T)\|_F^2 = \|\boldsymbol{A} - \boldsymbol{A}\boldsymbol{Q}\boldsymbol{Q}\|_F^2$.

Recall that $\boldsymbol{Q}$ is an orthogonal basis for the column span of $\boldsymbol{A}\boldsymbol{S}$, where $\boldsymbol{S}$ is a random sign matrix with $\frac{m}{3}$ columns. $\boldsymbol{Q}$ is thus an orthogonal basis for a linear sketch of $\boldsymbol{A}$'s column space, and it is well known that $\boldsymbol{Q}$ will align with large eigenvectors of $\boldsymbol{A}$, and $\|\boldsymbol{A} - \boldsymbol{A}\boldsymbol{Q}\boldsymbol{Q}^T\|_F^2$ will be small [Sar06, Woo14]. Concretely, applying Corollary 7 and Claim 1 from [MM20], we have that, as long as $\frac{m}{3} \geq O(k + \log(1/\delta))$, with probability $\geq 1-\delta$:

$$\|\boldsymbol{A} - \boldsymbol{A}\boldsymbol{Q}\boldsymbol{Q}^T\|_F^2 \leq 2\|\boldsymbol{A} - \boldsymbol{A}_k\|_F^2.$$

Accordingly, $\|\boldsymbol{\Delta}\|_F \leq 2\|\boldsymbol{A} - \boldsymbol{A}_k\|_F^2$ as required by Theorem 3.1. The result then immediately follows by setting $k = O(\sqrt{\log(1/\delta)}/\varepsilon + \log(1/\delta))$ and noting that Hutch++$(\boldsymbol{A}) = \left[\operatorname{tr}(\tilde{\boldsymbol{A}}) + \mathrm{H}_\ell(\boldsymbol{\Delta})\right]$ where $\ell = O(\sqrt{\log(1/\delta)}/\varepsilon + \log(1/\delta))$. ∎

3.1 A Non-Adaptive Variant of Hutch++

As discussed in Section 1, Algorithm 1 is *adaptive*: it uses the result of computing $\boldsymbol{A}\boldsymbol{S}$ to compute $\boldsymbol{Q}$, which is then multiplied by $\boldsymbol{A}$ to compute the $\operatorname{tr}(\boldsymbol{Q}^T\boldsymbol{A}\boldsymbol{Q})$ term. Meanwhile, Hutchinson's estimator is *non-adaptive*: it samples a single random matrix upfront, batch-multiplies by $\boldsymbol{A}$ once, and computes an approximation to $\operatorname{tr}(\boldsymbol{A})$ from the result, without any further queries.

Not only is non-adaptivity an interesting theoretical property, but it can be practically useful, since parallelism or block iterative methods often make it faster to multiply an implicit matrix by many vectors at once. With these considerations in mind, we describe a non-adaptive variant of Hutch++, which we call NA-Hutch++. NA-Hutch++ obtains nearly the same theoretical guarantees as Algorithm 1, although it tends to perform slightly worse in our experiments.

We leverage a streaming low-rank approximation result of Clarkson and Woodruff [CW09] which shows that if $\boldsymbol{S} \in \mathbb{R}^{d\times m}$ and $\boldsymbol{R} \in \mathbb{R}^{d\times cm}$ are sub-Gaussian random matrices with $m = O(k\log(1/\delta))$ and $c > 1$ a fixed constant, then with probability $1-\delta$, the matrix $\tilde{\boldsymbol{A}} = \boldsymbol{AR}(\boldsymbol{S}^T\boldsymbol{AR})^+(\boldsymbol{AS})^T$ satisfies $\|\boldsymbol{A}-\tilde{\boldsymbol{A}}\|_F \le 2\|\boldsymbol{A}-\boldsymbol{A}_k\|_F$. Here $^+$ denotes the Moore-Penrose pseudoinverse. We can compute $\operatorname{tr}(\tilde{\boldsymbol{A}})$ efficiently without explicitly constructing $\tilde{\boldsymbol{A}} \in \mathbb{R}^{d\times d}$ by noting that it is equal to $\operatorname{tr}((\boldsymbol{S}^T\boldsymbol{AR})^+(\boldsymbol{AS})^T(\boldsymbol{AR}))$ via the cyclic property of the trace. This yields:

Algorithm 2 NA-Hutch++ (Non-Adaptive variant of Hutch++)

input: Matrix-vector multiplication oracle for PSD matrix $\boldsymbol{A} \in \mathbb{R}^{d\times d}$. Number m of queries.
output: Approximation to $\operatorname{tr}(\boldsymbol{A})$.

1: Fix constants c_1, c_2, c_3 such that $c_1 < c_2$ and $c_1 + c_2 + c_3 = 1$.
2: Sample $\boldsymbol{S} \in \mathbb{R}^{d\times c_1 m}$, $\boldsymbol{R} \in \mathbb{R}^{d\times c_2 m}$, and $\boldsymbol{G} \in \mathbb{R}^{d\times c_3 m}$ with i.i.d. $\{+1,-1\}$ entries.
3: Compute $\boldsymbol{Z} = \boldsymbol{AR}$ and $\boldsymbol{W} = \boldsymbol{AS}$.
4: **return** NA-Hutch++$(\boldsymbol{A}) = \operatorname{tr}((\boldsymbol{S}^T\boldsymbol{Z})^+(\boldsymbol{W}^T\boldsymbol{Z})) + \frac{3}{m}\left[\operatorname{tr}(\boldsymbol{G}^T\boldsymbol{AG}) - \operatorname{tr}(\boldsymbol{G}^T\boldsymbol{Z}(\boldsymbol{S}^T\boldsymbol{Z})^+\boldsymbol{W}^T\boldsymbol{G})\right]$

NA-Hutch++ requires m matrix-vector multiplications with $\boldsymbol{A}$. In our experiments, it works well with $c_1 = c_3 = 1/4$ and $c_2 = 1/2$. Assuming $m < d$, it requires $O(dm^2)$ further runtime, to perform the matrix multiplications on line 4 and to compute $(\boldsymbol{S}^T\boldsymbol{Z})^+$, which takes $O(dm^2 + m^3)$ time.

THEOREM 3.2. *If NA-Hutch++ is implemented with $m = O(\log(1/\delta)/\varepsilon)$ matrix-vector multiplication queries and $\frac{c_2}{c_1}$ a sufficiently large constant, then for any PSD $\boldsymbol{A}$, with probability $\ge 1-\delta$, the output NA-Hutch++$(\boldsymbol{A})$ satisfies: $(1-\varepsilon)\operatorname{tr}(\boldsymbol{A}) \le$ NA-Hutch++$(\boldsymbol{A}) \le (1+\varepsilon)\operatorname{tr}(\boldsymbol{A})$.*

Proof. We apply Theorem 3.1 with $\tilde{\boldsymbol{A}} = \boldsymbol{Z}(\boldsymbol{S}^T\boldsymbol{Z})^+\boldsymbol{W}^T$, $\boldsymbol{\Delta} = \boldsymbol{A} - \tilde{\boldsymbol{A}}$, $k = O(1/\varepsilon)$ and $\ell = c_3 m = O(\frac{\log(1/\delta)}{\varepsilon})$. $\operatorname{tr}(\boldsymbol{A}) = \operatorname{tr}(\tilde{\boldsymbol{A}}) + \operatorname{tr}(\boldsymbol{\Delta})$ and NA-Hutch++$(\boldsymbol{A}) = [\operatorname{tr}(\tilde{\boldsymbol{A}}) + \mathrm{H}_\ell(\boldsymbol{\Delta})]$. By Theorem 4.7 of [CW09], since $c_1 m = O(k\log(1/\delta))$, $\|\boldsymbol{\Delta}\|_F \le 2\|\boldsymbol{A}-\boldsymbol{A}_k\|_F$ with probability $\ge 1-\delta$ as required. ∎

4 Lower Bounds

A natural question is if the $O(1/\varepsilon)$ matrix-vector query bound of Theorem 1.1 and Theorem 3.2 is tight. In this section, we prove that it is up to a logarithmic factor, even for algorithms that perform *adaptive* queries like Hutch++. Our lower bound is via a reduction to communication complexity: we show that a better algorithm for PSD trace estimation would imply a better 2-party communication protocol for the Gap-Hamming problem, which would violate known adaptive lower bounds for that problem [CR12]. To prove this result we need to assume a fixed precision model of computation. Specifically we require that the entries in each query vector $\mathbf{r}$ are integers bounded in absolute value by 2^b, for some fixed constant b. By scaling, this captures the setting where the query vectors are non-integer, but have bounded precision. Formally, we prove in Section 4.1:

THEOREM 4.1. *Any algorithm that accesses a positive semidefinite matrix $\boldsymbol{A}$ via matrix-vector multiplication queries $\boldsymbol{A}\mathbf{r}_1, \ldots, \boldsymbol{A}\mathbf{r}_m$, where $\mathbf{r}_1, \ldots, \mathbf{r}_m$ are possibly adaptively chosen vectors with integer entries in $\{-2^b, \ldots, 2^b\}$, requires $m = \Omega\left(\frac{1}{\varepsilon(b+\log(1/\varepsilon))}\right)$ such queries to output an estimate t so that, with probability $> 2/3$, $(1-\varepsilon)\operatorname{tr}(\boldsymbol{A}) \le t \le (1+\varepsilon)\operatorname{tr}(\boldsymbol{A})$.*

For constant b our lower bound is $\Omega\left(\frac{1}{\varepsilon\log(1/\varepsilon)}\right)$, which matches Theorem 1.1 and Theorem 3.2 up to a $\log(1/\varepsilon)$ factor. We also provide an alternative lower bound which holds in the real RAM model of computation (all inputs and arithmetic operations involve real numbers). This second lower bound is tight up to constants, but only applies to non-adaptive algorithms. It is proven using different information theoretic techniques – we reduce to a hypothesis testing problem involving *negatively spiked* covariance matrices [CMW15, PWBM18]. Formally, we prove in Appendix B:

THEOREM 4.2. *Any algorithm that accesses a postive semidefinite matrix $\boldsymbol{A}$ through matrix-vector multiplication queries $\boldsymbol{A}\mathbf{r}_1, \ldots, \boldsymbol{A}\mathbf{r}_m$, where $\mathbf{r}_1, \ldots, \mathbf{r}_m$ are real valued* non-adaptively *chosen vectors requires $m = \Omega\left(\frac{1}{\varepsilon}\right)$ such queries to output an estimate t so that, with probability $> 3/4$, $(1-\varepsilon)\operatorname{tr}(\boldsymbol{A}) \le t \le (1+\varepsilon)\operatorname{tr}(\boldsymbol{A})$.*

4.1 Adaptive lower bound The proof of Theorem 4.1 is based on reducing the Gap-Hamming problem to trace estimation. This problem has been well studied in communication complexity since its introduction in [IW03].

PROBLEM 4.1. (GAP-HAMMING) *Let Alice and Bob be communicating parties who hold vectors $\mathbf{s} \in \{-1,1\}^n$ and $\mathbf{t} \in \{-1,1\}^n$, respectively. The Gap-Hamming problem asks Alice and Bob to return:*

$$1 \text{ if } \langle \mathbf{s}, \mathbf{t}\rangle \ge \sqrt{n} \quad \text{and} \quad -1 \text{ if } \langle \mathbf{s}, \mathbf{t}\rangle \le -\sqrt{n}.$$

A tight lower bound on the unbounded round, randomized communication complexity of this problem was first

proven in [CR12], with alternative proofs appearing in [Vid12, She12]. Formally:

LEMMA 4.1. (THEOREM 2.6 IN [CR12]) *The randomized communication complexity for solving Problem 4.1 with probability $\geq 2/3$ is $\Omega(n)$ bits.*

With Lemma 4.1 in place, we have all we need to prove Theorem 4.1.

Proof. [Proof of Theorem 4.1] Fix a perfect square $n \in \mathbb{N}$. Consider an instance of Problem 4.1 with inputs $\mathbf{s} \in \mathbb{R}^n$ and $\mathbf{t} \in \mathbb{R}^n$. Let $\boldsymbol{S} \in \mathbb{R}^{\sqrt{n}\times\sqrt{n}}$ and $\boldsymbol{T} \in \mathbb{R}^{\sqrt{n}\times\sqrt{n}}$ contain the entries of $\mathbf{s}$ and $\mathbf{t}$ rearranged into matrices (e.g., placed left-to-right, top-to-bottom). Let $\boldsymbol{Z} = \boldsymbol{S} + \boldsymbol{T}$ and let $\boldsymbol{A} = \boldsymbol{Z}^T\boldsymbol{Z}$. $\boldsymbol{A}$ is positive semidefinite and we have:

$$\begin{aligned}\operatorname{tr}(\boldsymbol{A}) = \|\boldsymbol{Z}\|_F^2 = \|\mathbf{s}+\mathbf{t}\|_2^2 &= \|\mathbf{s}\|_2^2 + \|\mathbf{t}\|_2^2 + 2\langle \mathbf{s}, \mathbf{t}\rangle \\ &= 2n + 2\langle \mathbf{s}, \mathbf{t}\rangle.\end{aligned}$$

If $\langle \mathbf{s}, \mathbf{t}\rangle \geq \sqrt{n}$ then we will have $\operatorname{tr}(\boldsymbol{A}) \geq 2(n+\sqrt{n})$ and if $\langle \mathbf{s}, \mathbf{t}\rangle \leq -\sqrt{n}$ then we will have $\operatorname{tr}(\boldsymbol{A}) \leq 2(n-\sqrt{n})$. So, if Alice and Bob can approximate $\operatorname{tr}(\boldsymbol{A})$ up to relative error $(1 \pm 1/\sqrt{n})$, then they can solve Problem 4.1. We claim that they can do so with just $O(m \cdot \sqrt{n}(\log n + b))$ bits of communication if there exists an m-query adaptive matrix-vector multiplication algorithm for positive semidefinite trace estimation achieving error $(1 \pm 1/\sqrt{n})$.

Specifically, Alice takes charge of running the query algorithm. To compute $\boldsymbol{A}\mathbf{r}$ for a vector $\mathbf{r}$, Alice and Bob first need to compute $\boldsymbol{Z}\mathbf{r}$. To do so, Alice sends $\mathbf{r}$ to Bob, which takes $O(\sqrt{n} \cdot b)$ bits since $\mathbf{r}$ has entries bounded by 2^b. Bob then computes $\boldsymbol{T}\mathbf{r}$, which has entries bounded by $\sqrt{n}2^b$. He sends the result to Alice, using $O(\sqrt{n}(b + \log n))$ bits. Upon receiving $\boldsymbol{T}\mathbf{r}$, Alice computes $\boldsymbol{Z}\mathbf{r} = \boldsymbol{S}\mathbf{r} + \boldsymbol{T}\mathbf{r}$. Next, they need to multiply $\boldsymbol{Z}\mathbf{r}$ by $\boldsymbol{Z}^T$ to obtain $\boldsymbol{A}\mathbf{r} = \boldsymbol{Z}^T\boldsymbol{Z}\mathbf{r}$. To do so, Alice sends $\boldsymbol{Z}\mathbf{r}$ to Bob (again using $O(\sqrt{n}(b+\log n))$ bits) who computes $\boldsymbol{T}^T\boldsymbol{Z}\mathbf{r}$. The entries in this vector are bounded by $2n2^b$, so Bob sends the result back to Alice using $O(\sqrt{n}(b + \log n))$ bits. Finally, Alice computes $\boldsymbol{S}^T\boldsymbol{Z}\mathbf{r}$ and adds the result to $\boldsymbol{T}^T\boldsymbol{Z}\mathbf{r}$ to obtain $\boldsymbol{Z}^T\boldsymbol{Z}\mathbf{r} = \boldsymbol{A}\mathbf{r}$. Given this result, Alice chooses the next query vector according to the algorithm and repeats.

Overall, running the full matrix-vector query algorithm requires $O(m \cdot \sqrt{n}(\log n + b))$ bits of communication. So, from Lemma 4.1 we have that $m = \Omega(\sqrt{n}/(\log n + b))$ queries are needed to approximate the trace to accuracy $1 \pm \varepsilon$ for $\varepsilon = 1/\sqrt{n}$, with probability $> 2/3$. ∎

5 Experimental Validation

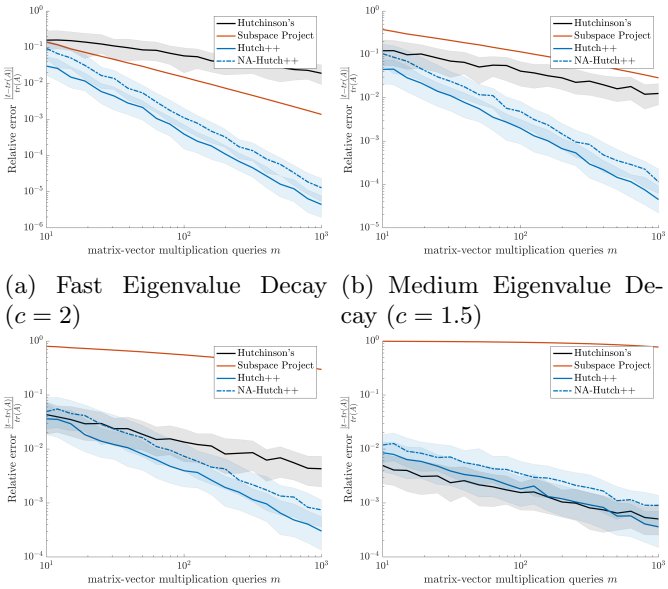
We complement our theory with experiments on synthetic matrices and real-world trace estimation problems. Code for Hutch++ and NA-Hutch++ is available at https://github.com/RaphaelArkadyMeyerNYU/HutchPlusPlus. We compare these methods to four algorithms, including both our adaptive and non-adaptive methods:

- **Hutchinson's.** The standard estimator run with $\{+1, -1\}$ random vectors.
- **Subspace Projection.** The method from [SAI17], which computes an orthogonal matrix $\boldsymbol{Q} \in \mathbb{R}^{d\times k}$ that approximately spans the top eigenvector subspace of $\boldsymbol{A} \in \mathbb{R}^{d\times d}$ and returns $\operatorname{tr}(\boldsymbol{Q}^T\boldsymbol{A}\boldsymbol{Q})$ as an approximation to $\operatorname{tr}(\boldsymbol{A})$. A similar approach is employed in [HL20]. [SAI17] computes $\boldsymbol{Q}$ using subspace iteration, which requires $k(q+1)$ matrix-vector multiplications when run for q iterations. A larger q results in a more accurate $\boldsymbol{Q}$, but requires more multiplications. As in [SAI17], we found that setting $q = 1$ gave the best performance, so we did so in our experiments. With $q = 1$, this method is similar to Hutch++, except that is does not approximate the remainder of the trace outside the top eigenspace.
- **Hutch++.** The adaptive method of Algorithm 1 with $\{+1, -1\}$ random vectors.
- **NA-Hutch++.** The non-adaptive method of Algorithm 2 with $c_1 = c_3 = 1/4$ and $c_2 = 1/2$ and $\{+1, -1\}$ random vectors.

5.1 Synthetic Matrices

We first test the methods above on random matrices with power law spectra. For varying constant c, we let $\boldsymbol{\Lambda}$ be diagonal with $\boldsymbol{\Lambda}_{ii} = i^{-c}$. We generate a random orthogonal matrix $\boldsymbol{Q} \in \mathbb{R}^{5000\times 5000}$ by orthogonalizing a random Gaussian matrix and set $\boldsymbol{A} = \boldsymbol{Q}^T\boldsymbol{\Lambda}\boldsymbol{Q}$. $\boldsymbol{A}$'s eigenvalues are the values in $\boldsymbol{\Lambda}$. A larger c results in a more quickly decaying spectrum, so we expect Subspace Projection to perform well. A smaller c results in a slowly decaying spectrum, which will mean that $\|\boldsymbol{A}\|_F \ll \operatorname{tr}(\boldsymbol{A})$. In this case, we expect Hutchinson's to outperform its worst case multiplicative error bound: instead of error $\pm\varepsilon\operatorname{tr}(\boldsymbol{A})$ after $O(1/\varepsilon^2)$ matrix-multiplication queries, Lemma 2.1 predicts error on the order of $\pm\varepsilon\|\boldsymbol{A}\|_F$. Concretely, for dimension $d = 5000$ and $c = 2$, we have $\|\boldsymbol{A}\|_F = .63 \cdot \operatorname{tr}(\boldsymbol{A})$ and for $c = .5$ we have $\|\boldsymbol{A}\|_F = .02 \cdot \operatorname{tr}(\boldsymbol{A})$. In general, unlike the Subspace Projection method and Hutchinson's estimator, we expect Hutch++ and NA-Hutch++ to be less sensitive to $\boldsymbol{A}$'s spectrum.

In Figure 1 we plot results for various c. Relative error should scale roughly as $\varepsilon = O(m^{-\gamma})$, where $\gamma = 1/2$ for Hutchinson's and $\gamma = 1$ for Hutch++ and NA-Hutch++. We thus use log-log plots, where we expect a linear relationship between the error ε and number of iterations m.



(a) Fast Eigenvalue Decay ($c = 2$)

(b) Medium Eigenvalue Decay ($c = 1.5$)

(c) Slow Eigenvalue Decay ($c = 1$)

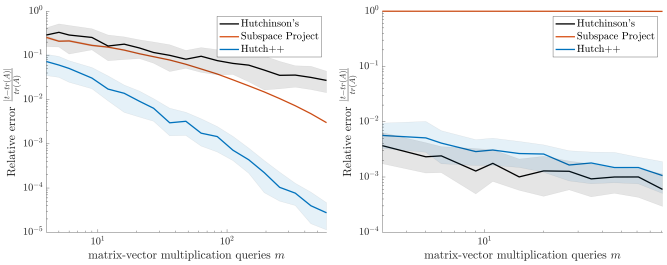
(d) Very Slow Eigenvalue Decay ($c = .5$)

Figure 1: Relative error versus number of matrix-vector multiplication queries for trace approximation algorithms run on random matrices with power law spectra. We report the median relative error of the approximation t after 200 trials. The upper and lower bounds of the shaded region around each curve are the 25^{th} and 75^{th} percentile errors. Subspace Projection has consistently low variance, but as expected, only performs better than Hutchinson's when $c = 2$ and there is very fast eigenvalue decay. Hutch++ and NA-Hutch++ typically outperform both methods.

The superior performance of Hutch++ and NA-Hutch++ shown in Figure 1 is not surprising. These methods are designed to achieve the "best of both worlds": when $\boldsymbol{A}$'s spectrum decays quickly, our methods approximate $\text{tr}(\boldsymbol{A})$ well by projecting off the top eigenvalues. When it decays slowly, they perform essentially no worse than Hutchinson's. We note that the adaptivity of Hutch++ leads to consistently better performance over NA-Hutch++, and the method is simpler to implement as we do not need to set the constants c_1, c_2, c_3. Accordingly, this is the method we move forward with in our real data experiments.

5.2 Real Matrices To evaluate the real-world performance of Hutch++ we test it in the common setting where $\boldsymbol{A} = f(\boldsymbol{B})$. In most applications, $\boldsymbol{B}$ is symmetric with eigendecomposition $\boldsymbol{V}^T\boldsymbol{\Lambda}\boldsymbol{V}$, and $f : \mathbb{R} \to \mathbb{R}$ is a function on real valued inputs. Then we have $f(\boldsymbol{B}) = \boldsymbol{V}^T f(\boldsymbol{\Lambda})\boldsymbol{V}$ where $f(\boldsymbol{\Lambda})$ is simply f applied to the real-valued eigenvalues on the diagonal of $\boldsymbol{\Lambda}$. When f returns negative values, $\boldsymbol{A}$ may not be postive semidefinite. Generally, computing $f(\boldsymbol{B})$ explicitly requires a full eigendecomposition and thus $\Omega(n^3)$ time. However, many iterative methods can more quickly approximate matrix-vector queries of the form $\boldsymbol{A}\mathbf{r} = f(\boldsymbol{B})\mathbf{r}$. The most popular and general is the Lanczos method, which we employ in our experiments [UCS17, MMS18].[2]

We consider trace estimation in three example applications, involving both PSD and non-PSD matrices. We test on relatively small inputs, for which we can explicitly compute $\text{tr}(f(\boldsymbol{B}))$ to use as a baseline for the approximation error. However, our methods can scale to much larger matrices.



(a) $\boldsymbol{A} = \exp(\boldsymbol{B})$, where $\boldsymbol{B}$ is the Roget's Thesaurus semantic graph adjacency matrix. For use in Estrada index computation. $\boldsymbol{A}$ is PSD.

(b) $\boldsymbol{A} = \log(\boldsymbol{B} + \lambda\boldsymbol{I})$, where $\boldsymbol{B}$ is a 2D Gaussian process kernel covariance matrix. For use in log-likelihood computation. $\boldsymbol{A}$ is not PSD.

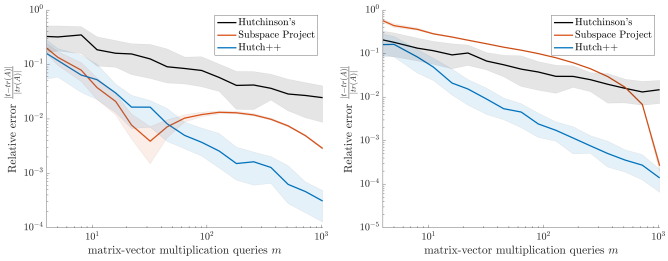
Figure 2: Relative error versus number of matrix-vector multiplication queries for trace approximations of transformed matrices, which were multiplied by vectors using the Lanczos method. We report median relative error of the approximation t after 100 trials. The upper and lower bounds of the shaded region around each curve are the 25^{th} and 75^{th} percentile errors. As expected, Subspace Project and Hutch++ outperform Hutchinson's when $\boldsymbol{A} = \exp(\boldsymbol{B})$, as exponentiating leads to a quickly decaying spectrum. On the other hand, Hutchinson's performs well for $\boldsymbol{A} = \log(\boldsymbol{B} + \lambda\boldsymbol{I})$, which has a very flat spectrum. Hutch++ is still essentially as fast, even though this matrix is not PSD. Subspace Project fails in this case because the top eigenvalues of $\boldsymbol{A}$ do not dominate its trace.

Graph Estrada Index. Given the binary adjacency matrix $\boldsymbol{B} \in \{0,1\}^{d\times d}$ of a graph G, the Estrada index is defined as $\text{tr}(\exp(\boldsymbol{B}))$ [Est00, dlPGR07], where $\exp(x) = e^x$. This index measures the strength of connectivity within G. A simple transformation of the

[2] We use our implementation of Lanczos available at https://github.com/cpmusco/fast-pcr, but modified to block matrix-vector multiplies when run on multiple query vectors $\mathbf{r}$.

Estrada index yields the *natural connectivity* metric, defined as $\log\left(\frac{1}{d}\operatorname{tr}(\exp(\boldsymbol{B}))\right)$ [JBYJHZ10, EHB12].

In our experiments, we approximated the Estrada index of the Roget's Thesaurus semantic graph, available from [BM06]. The Estrada index of this 1022 node graph was originally studied in [EH08]. We use the Lanczos method to approximate matrix multiplication with $\exp(\boldsymbol{B})$, running it for 40 iterations, after which the error of application was negligible compared to the approximation error of trace estimation. Results are shown in Figure 2.



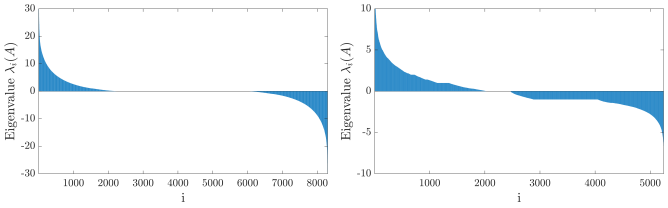
(a) $\boldsymbol{A} = \boldsymbol{B}^3$ where $\boldsymbol{B}$ is a Wikipedia voting network adjacency matrix. For use in triangle counting. $\boldsymbol{A}$ is not PSD.

(b) $\boldsymbol{A} = \boldsymbol{B}^3$ where $\boldsymbol{B}$ is an arXiv.org citation network adjacency matrix. For use in triangle counting. $\boldsymbol{A}$ is not PSD.

Figure 3: Relative error versus number of matrix-vector multiplication queries for trace approximations of transformed matrices. We report the median relative error of the approximation t after 100 trials. The upper and lower bounds of the shaded region around each curve are the 25^{th} and 75^{th} percentile errors. Hutch++ still outperforms the baseline methods even though $\boldsymbol{A}$ is not PSD. We note that Subspace Project has somewhat uneven performance: increasing m will take into account a larger number of top eigenvalues when approximating the trace. However, since these may be positive or negative, approximation error does not monotonically decrease. Hutch++ is not sensitive to this issue since it does not use *just* the top eigenvalues: see Figure 4 for more discussion.

Gaussian Process Log Likelihood. Let $\boldsymbol{B} \in \mathbb{R}^{d\times d}$ be a PSD kernel covariance matrix and let $\lambda \geq 0$ be a regularization parameter. In Gaussian process regression, the model log likelihood computation requires computing $\log\det(\boldsymbol{B}+\lambda\boldsymbol{I}) = \operatorname{tr}(f(\boldsymbol{B}))$ where $f(x) = \log(x+\lambda)$ [WR96, Ras04]. This quantity must be computed repeatedly for different choices of $\boldsymbol{B}$ and λ during hyper-parameter optimization, and it is often approximated using Hutchinson's method [BDKZ15, UCS17, HMAS17, DEN+17]. We note that, while $\boldsymbol{B}$ is positive semidefinite, $\log(\boldsymbol{B}+\lambda\boldsymbol{I})$ typically will not be. So our theoretical bounds do not apply in this case, but Hutch++ can be applied unmodified, and as we see in Figure 2, still gives good performance.

In our experiments we consider a benchmark 2D Gaussian process regression problem from the GIS literature, involving precipitation data from Slovakia [NM13]. $\boldsymbol{B}$ is the kernel covariance matrix on 6400 randomly selected training points out of 196,104 total points. Following the setup of [EMM20], we let $\boldsymbol{B}$ be a Gaussian kernel matrix with width parameter $\gamma = 64$ and regularization parameter $\lambda = .008$, both determined via cross-validation on ℓ_2 regression loss.



(a) Eigenvalues for Wikipedia voting network adjacency matrix $\boldsymbol{B}$.

(b) Eigenvalues for arXiv.org citation network adjacency matrix $\boldsymbol{B}$.

Figure 4: Even when estimating the trace of a non-PSD matrix like $\boldsymbol{A} = \boldsymbol{B}^3$, which for the triangle counting examples above will have both positive and negative eigenvalues, Hutch++ can far outperform Hutchinson's method. It will approximately project off the largest *magnitude* eigenvalues from $\boldsymbol{A}$ (whether postive or negative), which will reduce the variance in estimating the trace $\operatorname{tr}(\boldsymbol{A}) = \sum_{i=1}^{d}\lambda_i(\boldsymbol{B})^3$.

Graph Triangle Counting. Given the binary adjacency matrix $\boldsymbol{B} \in \{0,1\}^{d\times d}$ of an undirected graph G, the number of triangles in G is equal to $\frac{1}{6}\operatorname{tr}(\boldsymbol{B}^3)$. The triangle count is an important measure of local connectivity and extensive research studies its efficient approximation [SW05, BBCG08, PT12]. Popular approaches include applying Hutchinson's method to $\boldsymbol{A} = \boldsymbol{B}^3$ [Avr10], or using the EigenTriangle estimator, which is similar to the Subspace Projection method [Tso08].

In our experiments, we study approximate triangle counting on two common benchmark graphs: an arXiv.org collaboration network[3] with 5,243 nodes and 48,260 triangles, and a Wikipedia administrator voting network[4] with 7,115 nodes and 608,389 triangles.

We again note that the adjacency matrix $\boldsymbol{B}$ is not positive semidefinite, and neither is $\boldsymbol{A} = \boldsymbol{B}^3$. Nevertheless, we can apply Hutch++ and see very strong performance. In this setting we do not need

[3]Link: https://snap.stanford.edu/data/ca-GrQc.html.
[4]Link: https://snap.stanford.edu/data/wiki-Vote.html.

to apply Lanczos for matrix-vector query computation: $\boldsymbol{Ar}$ can be computed exactly using three matrix-vector multiplications with $\boldsymbol{B}$. Results are shown Figure 3 with graph spectral visualized in Figure 4

Acknowledgments: D. Woodruff would like to thank support from the National Institute of Health (NIH) grant 5R01 HG 10798-2 and a Simons Investigator Award.

References

[Ach03] Dimitris Achlioptas. Database-friendly random projections: Johnson-Lindenstrauss with binary coins. *Journal of Computer and System Sciences*, 66(4):671–687, 2003. Preliminary version in the 20th Symposium on Principles of Database Systems (PODS).

[APJ+18] Ryan P. Adams, Jeffrey Pennington, Matthew J. Johnson, Jamie Smith, Yaniv Ovadia, Brian Patton, and James Saunderson. Estimating the spectral density of large implicit matrices. *arXiv:1802.03451*, 2018.

[AT11] Haim Avron and Sivan Toledo. Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. *Journal of the ACM*, 58(2), 2011.

[Avr10] Haim Avron. Counting triangles in large graphs using randomized matrix trace estimation. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2010.

[BBCG08] Luca Becchetti, Paolo Boldi, Carlos Castillo, and Aristides Gionis. Efficient semi-streaming algorithms for local triangle counting in massive graphs. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 16–24, 2008.

[BDKZ15] Christos Boutsidis, Petros Drineas, Prabhanjan Kambadur, and Anastasios Zouzias. A randomized algorithm for approximating the log determinant of a symmetric positive definite matrix. *Linear Algebra and its Applications*, 533, 03 2015.

[BHSW20] Mark Braverman, Elad Hazan, Max Simchowitz, and Blake Woodworth. The gradient complexity of linear regression. In *Proceedings of the 33rd Annual Conference on Computational Learning Theory (COLT)*, volume 125, pages 627–647, 2020.

[BKKS20] Vladimir Braverman, Robert Krauthgamer, Aditya Krishnan, and Roi Sinoff. Schatten norms in matrix streams: Hello sparsity, goodbye dimension. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

[BM06] Vladimir Batagelj and Andrej Mrvar. Pajek datasets. http://vlado.fmf.uni-lj.si/pub/networks/data/, 2006.

[CEM+15] Michael Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Madalina Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC)*, pages 163–172, 2015.

[Che16] Jie Chen. How accurately should I compute implicit matrix-vector products when applying the Hutchinson trace estimator? *SIAM Journal on Scientific Computing*, 38(6):A3515–A3539, 2016.

[CK20] Alice Cortinovis and Daniel Kressner. On randomized trace estimates for indefinite matrices with an application to determinants. *arXiv:2005.10009*, 2020.

[CKSV18] David Cohen-Steiner, Weihao Kong, Christian Sohler, and Gregory Valiant. Approximating the spectrum of a graph. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1263–1271, 2018.

[CMW15] Tony Cai, Zongming Ma, and Yihong Wu. Optimal estimation and rank detection for sparse spiked covariance matrices. *Probability theory and related fields*, 161:781–815, 2015.

[CR12] Amit Chakrabarti and Oded Regev. An optimal lower bound on the communication complexity of gap-hamming-distance. *SIAM Journal on Computing*, 41(5):1299–1317, 2012.

[CW09] Kenneth L. Clarkson and David P. Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 205–214, 2009.

[DEN+17] Kun Dong, David Eriksson, Hannes Nickisch, David Bindel, and Andrew Gordon Wilson. Scalable log determinants for Gaussian process kernel learning. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pages 6327–6337, 2017.

[DG03] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003.

[dlPGR07] José Antonio de la Peña, Ivan Gutman, and Juan Rada. Estimating the Estrada index. *Linear Algebra and its Applications*, 427(1):70–76, 2007.

[DNPS16] Edoardo Di Napoli, Eric Polizzi, and Yousef Saad. Efficient estimation of eigenvalue counts in an interval. *Numerical Linear Algebra with Applications*, 2016.

[EH08] Ernesto Estrada and Naomichi Hatano. Communicability in complex networks. *Phys. Rev. E*, 77:036111, Mar 2008.

[EHB12] Ernesto Estrada, Naomichi Hatano, and Michele Benzi. The physics of communicability in complex networks. *Physics Reports*, 514(3):89 – 119, 2012.

[EMM20] Tamás Erdélyi, Cameron Musco, and Christopher Musco. Fourier sparse leverage scores and approximate kernel learning. *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020.

[Est00] Ernesto Estrada. Characterization of 3d molecular structure. *Chemical Physics Letters*, 319(5-6):713–718, 2000.

[FKN90] Kaitai Fang, Samuel Kotz, and Kai Wang Ng. *Symmetric Multivariate and Related Distributions.*

London: Chapman and Hall, 1990.

[Gir87] Didier Girard. Un algorithme simple et rapide pour la validation croisee géénéralisée sur des problémes de grande taille. Technical report, 1987.

[GSO17] Arjun Singh Gambhir, Andreas Stathopoulos, and Kostas Orginos. Deflation as a method of variance reduction for estimating the trace of a matrix inverse. *SIAM Journal on Scientific Computing*, 39(2):A532–A558, 2017.

[Hig08] Nicholas J. Higham. *Functions of Matrices: Theory and Computation*. Society for Industrial and Applied Mathematics, 2008.

[HL20] Yuanyang Zhu Hanyu Li. Randomized block Krylov space methods for trace and log-determinant estimators. *arXiv:2003.00212*, 2020.

[HMAS17] Insu Han, Dmitry Malioutov, Haim Avron, and Jinwoo Shin. Approximating the spectral sums of large-scale matrices using stochastic Chebyshev approximations. *SIAM Journal on Scientific Computing*, 2017.

[HMS15] Insu Han, Dmitry Malioutov, and Jinwoo Shin. Large-scale log-determinant computation through stochastic Chebyshev expansions. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 908–917, 2015.

[Hut90] Michael F. Hutchinson. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 19(2):433–450, 1990.

[IW03] Piotr Indyk and David Woodruff. Tight lower bounds for the distinct elements problem. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2003.

[JBYJHZ10] WU Jun, Mauricio Barahona, Tan Yue-Jin, and Deng Hong-Zhong. Natural connectivity of complex networks. *Chinese Physics Letters*, 27(7):078902, 2010.

[Lin17] Lin Lin. Randomized estimation of spectral densities of large matrices made accurate. *Numerische Mathematik*, 136(1):183–213, 2017.

[LSTZ20] Jerry Li, Aaron Sidford, Kevin Tian, and Huishuai Zhang. Well-conditioned methods for ill-conditioned systems: Linear regression with semi-random noise. *arXiv:2008.01722*, 2020.

[LSY16] Lin Lin, Yousef Saad, and Chao Yang. Approximating spectral densities of large matrices. *SIAM Review*, 58(1):34–65, 2016.

[MM20] Cameron Musco and Christopher Musco. Projection-cost-preserving sketches: Proof strategies and constructions. *arXiv:2004.08434*, 2020.

[MMS18] Cameron Musco, Christopher Musco, and Aaron Sidford. Stability of the Lanczos method for matrix function approximation. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1605–1624, 2018.

[MNS+18] Cameron Musco, Praneeth Netrapalli, Aaron Sidford, Shashanka Ubaru, and David P. Woodruff. Spectrum approximation beyond fast matrix multiplication: Algorithms and hardness. *Proceedings of the 9th Conference on Innovations in Theoretical Computer Science (ITCS)*, 2018.

[MS77] Florence Jessie MacWilliams and Neil James Alexander Sloane. *The theory of error correcting codes*, volume 16. Elsevier, 1977.

[NM13] Markus Neteler and Helena Mitasova. *Open source GIS: a GRASS GIS approach*, volume 689. Springer Science & Business Media, 2013.

[PT12] Rasmus Pagh and Charalampos E Tsourakakis. Colorful triangle counting and a mapreduce implementation. *Information Processing Letters*, 112(7):277–281, 2012.

[PWBM18] Amelia Perry, Alexander Wein, Afonso Bandeira, and Ankur Moitra. Optimality and sub-optimality of PCA I: Spiked random matrix models. *Annals of Statistics*, 46:2416–2451, 10 2018.

[RA15] Farbod Roosta-Khorasani and Uri M. Ascher. Improved bounds on sample size for implicit matrix trace estimators. *Foundations of Computational Mathematics*, 15(5):1187–1212, 2015.

[Ras04] Carl Edward Rasmussen. Gaussian Processes in Machine Learning. In *Advanced Lectures on Machine Learning*, pages 63–71. Springer, 2004.

[RV+13] Mark Rudelson, Roman Vershynin, et al. Hanson-Wright inequality and sub-Gaussian concentration. *Electronic Communications in Probability*, 18, 2013.

[SAI17] Arvind K. Saibaba, Alen Alexanderian, and Ilse C. F. Ipsen. Randomized matrix-free trace and log-determinant estimators. *Numerische Mathematik*, 137(2):353–395, 2017.

[Sar06] Tamas Sarlos. Improved approximation algorithms for large matrices via random projections. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 143–152, 2006.

[SEAR18] Max Simchowitz, Ahmed El Alaoui, and Benjamin Recht. Tight query complexity lower bounds for PCA via finite sample deformed Wigner law. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1249–1259, 2018.

[She12] Alexander A. Sherstov. The communication complexity of gap hamming distance. *Theory of Computing*, 8(8):197–208, 2012.

[SLO13] Andreas Stathopoulos, Jesse Laeuchli, and Kostas Orginos. Hierarchical probing for estimating the trace of the matrix inverse on toroidal lattices. *SIAM Journal on Scientific Computing*, 35(5):S299–S322, 2013.

[SW05] Thomas Schank and Dorothea Wagner. Finding, counting and listing all triangles in large graphs, an experimental study. In *International Workshop on Experimental and Efficient Algorithms*, pages 606–609. Springer, 2005.

[SWYZ19] Xiaoming Sun, David P. Woodruff, Guang Yang, and Jialin Zhang. Querying a matrix through matrix-vector products. In *Proceedings of the 46th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 132, pages 94:1–94:16,

2019.

[TS11] Jok M. Tang and Yousef Saad. Domain-decomposition-type methods for computing the diagonal of a matrix inverse. *SIAM Journal on Scientific Computing*, 33(5):2823–2847, 2011.

[Tso08] Charalampos E Tsourakakis. Fast counting of triangles in large real networks without counting: Algorithms and laws. In *2008 Eighth IEEE International Conference on Data Mining*, pages 608–617, 2008.

[UCS17] 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.

[US18] Shashanka Ubaru and Yousef Saad. Applications of trace estimation techniques. In *High Performance Computing in Science and Engineering*, pages 19–33, 2018.

[Vid12] Thomas Vidick. A concentration inequality for the overlap of a vector on a large set, with application to the communication complexity of the gap-hamming-distance problem. *Chicago Journal of Theoretical Computer Science*, 2012.

[Woo14] David P. Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science*, 10(1–2):1–157, 2014.

[WR96] Christopher K. I. Williams and Carl Edward Rasmussen. Gaussian Processes for Regression. In *Advances in Neural Information Processing Systems 9 (NeurIPS)*, pages 514–520, 1996.

[WSMB20] Sheng Wang, Yuan Sun, Christopher Musco, and Zhifeng Bao. Route planning for robust transit networks: When connectivity matters. *Preprint*, 2020.

[WWZ14] Karl Wimmer, Yi Wu, and Peng Zhang. Optimal query complexity for estimating the trace of a matrix. In *Proceedings of the 41st International Colloquium on Automata, Languages and Programming (ICALP)*, pages 1051–1062, 2014.

A Proof of Lemma 2.1

We start by stating the Hanson-Wright inequality for i.i.d sub-Gaussian random variables:

Imported Theorem A.1. ([RV+13]) *Let $\mathbf{x} \in \mathbb{R}^n$ be a vector of mean 0, i.i.d. sub-Gaussian random variables with constant sub-Gaussian parameter C. Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ be a matrix. Then, there exists a constant c only depending on C such that for every $t \geq 0$,*

$$\Pr\left\{|\mathbf{x}^T\boldsymbol{A}\mathbf{x} - \mathbb{E}[\mathbf{x}^T\boldsymbol{A}\mathbf{x}]| > t\right\} \leq 2\exp\left(-c\cdot\min\left\{\frac{t^2}{\|\boldsymbol{A}\|_F^2}, \frac{t}{\|\boldsymbol{A}\|_2}\right\}\right).$$

Above, $\|\boldsymbol{A}\|_2 = \max_x \|\boldsymbol{A}\mathbf{x}\|_2/\|\mathbf{x}\|_2$ denotes the spectral norm. We refer the reader to [RV+13] for a formal definition of sub-Gaussian random variables: both normal $\mathcal{N}(0,1)$ random variables and ± 1 random variables are sub-Gaussian with constant C.

Lemma 2.1 Restated 1. *Let $\boldsymbol{A} \in \mathbb{R}^{d\times d}$, $\delta \in (0, 1/2]$, $\ell \in \mathbb{N}$. Let $\mathrm{H}_\ell(\boldsymbol{A})$ be the ℓ-query Hutchinson estimator defined in (1.1), implemented with mean 0, i.i.d. sub-Gaussian random variables with constant sub-Gaussian parameter. For fixed constants c, C, if $\ell > c\log(1/\delta)$, then with prob. $1-\delta$,*

$$|\mathrm{H}_\ell(\boldsymbol{A}) - \mathrm{tr}(\boldsymbol{A})| \leq C\sqrt{\frac{\log(1/\delta)}{\ell}}\|\boldsymbol{A}\|_F.$$

Proof. Let $\bar{\boldsymbol{A}} \in \mathbb{R}^{\ell d\times \ell d}$ be a block-diagonal matrix formed from ℓ repetitions of $\boldsymbol{A}$:

$$\bar{\boldsymbol{A}} := \begin{bmatrix} \boldsymbol{A} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{A} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \boldsymbol{A} \end{bmatrix}.$$

Let $\boldsymbol{G} \in \mathbb{R}^{d\times\ell}$ be as in (1.1). Let $\mathbf{g}_i$ be $\boldsymbol{G}$'s i^{th} column and let $\mathbf{g} = [\mathbf{g}_1, \dots, \mathbf{g}_\ell] \in \mathbb{R}^{d\ell}$ be a vectorization of $\boldsymbol{G}$. We have that $\ell\cdot\mathrm{H}_\ell(\boldsymbol{A}) = \mathrm{tr}(\boldsymbol{G}^T\boldsymbol{A}\boldsymbol{G}) = \mathbf{g}^T\bar{\boldsymbol{A}}\mathbf{g}$. So, by Imported Theorem A.1,

$$\Pr\left\{|\mathbf{g}^T\bar{\boldsymbol{A}}\mathbf{g} - \mathbb{E}[\mathbf{g}^T\bar{\boldsymbol{A}}\mathbf{g}]| > t\right\} \leq 2\exp\left(-c\cdot\min\left\{\frac{t^2}{\|\bar{\boldsymbol{A}}\|_F^2}, \frac{t}{\|\bar{\boldsymbol{A}}\|_2}\right\}\right). \tag{A.1}$$

We let $t' = t/\ell$, and substitute $\mathbb{E}[\mathbf{g}^T\bar{\boldsymbol{A}}\mathbf{g}] = \mathrm{tr}(\bar{\boldsymbol{A}}) = \ell\,\mathrm{tr}(\boldsymbol{A})$, $\|\bar{\boldsymbol{A}}\|_F^2 = \ell\|\boldsymbol{A}\|_F^2$, and $\|\bar{\boldsymbol{A}}\|_2 = \|\boldsymbol{A}\|_2$ into (A.1) to get:

$$\Pr\left\{|\mathrm{H}_\ell(\boldsymbol{A}) - \mathrm{tr}(\boldsymbol{A})| > t'\right\} \leq 2\exp\left(-c\min\left\{\frac{\ell t'^2}{\|\boldsymbol{A}\|_F^2}, \frac{\ell t'}{\|\boldsymbol{A}\|_2}\right\}\right).$$

Now, taking $t' = \sqrt{\frac{\ln(2/\delta)}{c\ell}}\|\boldsymbol{A}\|_F$, we have:

$$\Pr\left\{|\mathrm{H}_\ell(\boldsymbol{A}) - \mathrm{tr}(\boldsymbol{A})| > \sqrt{\frac{\ln(2/\delta)}{c\ell}}\|\boldsymbol{A}\|_F\right\} \leq 2\exp\left(-\min\left\{\log(2/\delta), \sqrt{c\ell\log(2/\delta)}\frac{\|\boldsymbol{A}\|_F}{\|\boldsymbol{A}\|_2}\right\}\right)$$

Since $\frac{\|\boldsymbol{A}\|_F}{\|\boldsymbol{A}\|_2} \geq 1$, if we take $\ell \geq \ln(2/\delta)/c$, we have that the minimum takes value $\log(2/\delta)$, so

$$\Pr\left\{|\mathrm{H}_\ell(\boldsymbol{A}) - \mathrm{tr}(\boldsymbol{A})| > \sqrt{\frac{\ln(2/\delta)}{c\ell}}\|\boldsymbol{A}\|_F\right\} \leq \delta.$$

The final result follows from noting that $\ln(2/\delta) \leq 2\ln(1/\delta)$ for $\delta \leq 1/2$. ∎

B Proof of Theorem 4.2

To prove our non-adaptive lower bound for the real RAM moodel we first introduce a simple testing problem which we reduce to estimating the trace of a PSD matrix $\boldsymbol{A}$ to $(1 \pm \varepsilon)$ relative error:

PROBLEM B.1. *Fix $d, n \in \mathbb{N}$ such that $d \geq n$ and $n := \frac{1}{\varepsilon}$ for $\varepsilon \in (0, 1]$. Let $\boldsymbol{D}_1 = \boldsymbol{I}_n$ and $\boldsymbol{D}_2 = \begin{pmatrix} \boldsymbol{I}_{n-1} \\ 0 \end{pmatrix}$.[5] Consider $\boldsymbol{A} = \boldsymbol{G}^T \boldsymbol{D} \boldsymbol{G}$ generated by selecting $\boldsymbol{G} \in \mathbb{R}^{n \times d}$ with i.i.d. random Guassian $\mathcal{N}(0, 1)$ entries and $\boldsymbol{D} = \boldsymbol{D}_1$ or $\boldsymbol{D} = \boldsymbol{D}_2$ with equal probability. Then consider any algorithm which fixes a query matrix $\boldsymbol{U} \in \mathbb{R}^{d \times m}$, observes $\boldsymbol{A}\boldsymbol{U} \in \mathbb{R}^{d \times m}$, and guesses if $\boldsymbol{D} = \boldsymbol{D}_1$ or $\boldsymbol{D} = \boldsymbol{D}_2$.*

The reduction from Problem B.1 to relative error trace estimation is as follows:

LEMMA B.1. *For any $\varepsilon \in (0, 1]$ and sufficient large d, if a randomized algorithm $\mathcal{A}$ can estimate the trace of any $d \times d$ PSD matrix to relative error $1 \pm \frac{\varepsilon}{4}$ with success probability $\geq \frac{3}{4}$ using m queries, then $\mathcal{A}$ can be used to solve Problem B.1 with success probability $\geq \frac{2}{3}$ using m queries.*

Proof. To solve Problem B.1 we simply apply $\mathcal{A}$ to the matrix $\boldsymbol{A} = \boldsymbol{G}^T \boldsymbol{D} \boldsymbol{G}$ and guess $\boldsymbol{D}_1$ if the trace is closer to $\frac{d}{\varepsilon}$ and $\boldsymbol{D}_2$ if it's closer to $\frac{d}{\varepsilon} - d$. To see that this succeeds with probability $2/3$, we first need to understand the trace of $\boldsymbol{A}$. To do so, note that $\operatorname{tr}(\boldsymbol{A}) = \operatorname{tr}(\boldsymbol{G}^T \boldsymbol{D} \boldsymbol{G})$ is simply a scaled Hutchinson estimate for $\operatorname{tr}(\boldsymbol{D})$, i.e. $\operatorname{tr}(\boldsymbol{G}^T \boldsymbol{D} \boldsymbol{G}) = d \cdot \mathrm{H}_d(\boldsymbol{D})$. So, via Lemma 2.1, for large enough d we have that with probability $\geq \frac{11}{12}$ both of the following hold:

$$\frac{1}{d} \operatorname{tr}(\boldsymbol{G}^T \boldsymbol{D}_1 \boldsymbol{G}) \geq \left(1 - \frac{\varepsilon}{4}\right) \operatorname{tr}(\boldsymbol{D}_1) \text{ and}$$
$$\frac{1}{d} \operatorname{tr}(\boldsymbol{G}^T \boldsymbol{D}_2 \boldsymbol{G}) \leq \left(1 + \frac{\varepsilon}{4}\right) \operatorname{tr}(\boldsymbol{D}_2).$$

Additionally, with probability $\frac{3}{4}$, $\mathcal{A}$ computes an approximation Z with $(1 - \frac{\varepsilon}{4}) \operatorname{tr}(\boldsymbol{A}) \leq Z \leq (1 + \frac{\varepsilon}{4}) \operatorname{tr}(\boldsymbol{A})$. By a union bound, all of the above events happen with probability $\geq \frac{2}{3}$. If $\boldsymbol{D} = \boldsymbol{D}_1$:

$$Z \geq (1 - \tfrac{\varepsilon}{4}) \operatorname{tr}(\boldsymbol{A}) \geq (1 - \tfrac{\varepsilon}{4})^2 \cdot d \cdot \operatorname{tr}(\boldsymbol{D}_1) > (1 - \tfrac{\varepsilon}{2}) \cdot \frac{d}{\varepsilon}.$$

On the other hand, if $\boldsymbol{D} = \boldsymbol{D}_2$,

$$Z \leq (1 + \tfrac{\varepsilon}{4}) \operatorname{tr}(\boldsymbol{A}) \leq (1 + \tfrac{\varepsilon}{4})^2 \cdot d \cdot \operatorname{tr}(\boldsymbol{D}_2)$$
$$= (1 + \tfrac{\varepsilon}{4})^2 \cdot (1 - \varepsilon) \cdot \frac{d}{\varepsilon} < (1 - \tfrac{\varepsilon}{2}) \cdot \frac{d}{\varepsilon}.$$

Thus, with probability $2/3$, Z is closer to $\frac{d}{\varepsilon}$ when $\boldsymbol{D} = \boldsymbol{D}_1$ and closer to $\frac{d}{\varepsilon} - d$ when $\boldsymbol{D} = \boldsymbol{D}_2$, so the proposed scheme guesses correctly. ∎

In the remainder of the section we show that Problem B.1 requires $\Omega(1/\varepsilon)$ queries, which combined with Lemma B.1 proves our main lower bound, Theorem 4.2. Throughout, we let $\boldsymbol{X} \overset{\text{dist}}{=} \boldsymbol{Y}$ denote that $\boldsymbol{X}$ and $\boldsymbol{Y}$ are identically distributed. We first argue that for Problem B.1, the non-adaptive query matrix $\boldsymbol{U}$ might as well be chosen to be the first m standard basis vectors.

LEMMA B.2. *For Problem B.1, without loss of generality, we may assume that the query matrix $\boldsymbol{U}$ equals $\boldsymbol{U} = \boldsymbol{E}_m = \begin{bmatrix} \mathbf{e}_1 & \dots & \mathbf{e}_m \end{bmatrix}$, the first m standard basis vectors.*

Proof. First, we may assume without loss of generality that $\boldsymbol{U}$ is orthonormal, since if it were not, we could simply reconstruct the queries $\boldsymbol{A}\boldsymbol{U}$ by querying $\boldsymbol{A}$ with an orthonormal basis for the columns of $\boldsymbol{U}$. Next, by rotational invariance of the Gaussian distribution, if $\boldsymbol{G} \in \mathbb{R}^{n \times d}$ is an i.i.d. $\mathcal{N}(0, 1)$ matrix, and $\boldsymbol{Q} \in \mathbb{R}^{d \times d}$ is any orthogonal matrix, then $\boldsymbol{G}\boldsymbol{Q}$ is distributed identically to $\boldsymbol{G}$. Let $\bar{\boldsymbol{U}} \in \mathbb{R}^{d \times d - m}$ be any orthonormal span for the nullspace of $\boldsymbol{U}$, so that $\boldsymbol{Q} := \begin{bmatrix} \boldsymbol{U} & \bar{\boldsymbol{U}} \end{bmatrix}$ is orthogonal. We have that $\boldsymbol{Q}\boldsymbol{G}^T \boldsymbol{D} \boldsymbol{G} \boldsymbol{E}_m \overset{\text{dist}}{=} \boldsymbol{Q}\boldsymbol{Q}^T \boldsymbol{G}^T \boldsymbol{D} \boldsymbol{G} \boldsymbol{Q} \boldsymbol{E}_m = \boldsymbol{G}^T \boldsymbol{D} \boldsymbol{G} \boldsymbol{U}$. So, using the result $\boldsymbol{G}^T \boldsymbol{D} \boldsymbol{G} \boldsymbol{E}_m$ of querying with matrix $\boldsymbol{E}_m$, we can just multiply by $\boldsymbol{Q}$ on the left to obtain a set of vectors that has the same distribution as if $\boldsymbol{U}$ had been used as a query matrix. ∎

With Lemma B.2 in place, we are able to reduce Problem B.1 to a simpler testing problem on distinguishing m random vectors drawn from normal distributions with different covariance matrices:

PROBLEM B.2. *Let $n = \frac{1}{\varepsilon}$ and let $\mathbf{z} \in \mathbb{R}^n$ be a uniformly random unit vector. Let $\boldsymbol{N} \in \mathbb{R}^{n \times m}$ contain m i.i.d. random Gaussian vectors drawn from an n-dimensional Gaussian distribution, $\mathcal{N}(\mathbf{0}, \boldsymbol{C})$, where the covariance matrix $\boldsymbol{C}$ either equals $\boldsymbol{I}$ or $\boldsymbol{I} - \mathbf{z}\mathbf{z}^T$, with equality probability. The goal is to use $\boldsymbol{N}$ to distinguish, with probability $> \frac{1}{2}$ what the true identity of $\boldsymbol{C}$ is.*

LEMMA B.3. *Let $\mathcal{A}$ be an algorithm that solves Problem B.1 with m queries and success probability p. Then $\mathcal{A}$ can be used to solve Problem B.2 with m Gaussian samples and the same success probability.*

Proof. By Lemma B.2, it suffices to show how to use the observed matrix $\boldsymbol{N}$ in Problem B.2 to create a sample from the distribution $\boldsymbol{G}^T \boldsymbol{D} \boldsymbol{G} \boldsymbol{E}_m$ where $\boldsymbol{G} \in \mathbb{R}^{n \times d}$ has i.i.d. $\mathcal{N}(0, 1)$ entries. Specifically, we claim that, if we

[5] Here $\boldsymbol{I}_r$ denotes an $r \times r$ identity matrix.

sample $\boldsymbol{L} \in \mathbb{R}^{n\times(d-m)}$ with i.i.d. $\mathcal{N}(0,1)$ entries, and compute

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{N}^T\boldsymbol{N} \\ \boldsymbol{L}^T\boldsymbol{N} \end{bmatrix},$$

then $\boldsymbol{M}$ is identically distributed to $\boldsymbol{G}^T\boldsymbol{D}\boldsymbol{G}\boldsymbol{E}_m$. I.e, if we let $\boldsymbol{G}_m \in \mathbb{R}^{n\times m}$ contain the first m columns of $\boldsymbol{G}$ and let $\boldsymbol{G}_{d-m}$ contain the remaining $d-m$ columns, our goal is the show that $\boldsymbol{M} \overset{\text{dist}}{=} \begin{bmatrix}\boldsymbol{G}_m\ \boldsymbol{G}_{d-m}\end{bmatrix}^T \boldsymbol{D}\boldsymbol{G}_m$.

To see this is the case, let $\boldsymbol{Z} \in \mathbb{R}^{n\times n}$ be a uniformly random orthogonal matrix and let $\boldsymbol{D}_1, \boldsymbol{D}_2$ be as in Problem B.1. The first observation is that $\boldsymbol{N} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{C})$ is identically distributed to $\boldsymbol{Z}\boldsymbol{D}\boldsymbol{S}$ where $\boldsymbol{S} \in \mathbb{R}^{d\times m}$ has standard normal entries $\sim \mathcal{N}(0,1)$ and $\boldsymbol{D} = \boldsymbol{D}_1$ or $\boldsymbol{D} = \boldsymbol{D}_2$ with equal probability. This follows simply from that fact that $\boldsymbol{Z}\boldsymbol{D}_1\boldsymbol{D}_1\boldsymbol{Z}^T = \boldsymbol{I}$ and $\boldsymbol{Z}\boldsymbol{D}_2\boldsymbol{D}_2\boldsymbol{Z}^T = \boldsymbol{I} - \mathbf{z}_n\mathbf{z}_n^T$, where $\mathbf{z}_n$ is the last row of $\boldsymbol{Z}$, which is a uniformly random unit vector. It follows that $\boldsymbol{N}^T\boldsymbol{N} \overset{\text{dist}}{=} \boldsymbol{S}^T\boldsymbol{D}^T\boldsymbol{Z}^T\boldsymbol{Z}\boldsymbol{D}\boldsymbol{S} = \boldsymbol{S}^T\boldsymbol{D}\boldsymbol{S} \overset{\text{dist}}{=} \boldsymbol{G}_m^T\boldsymbol{D}\boldsymbol{G}_m$. Next, observe that $\boldsymbol{L}^T\boldsymbol{Z}$ is independent of $\boldsymbol{G}$ and has i.i.d. $\mathcal{N}(0,1)$ entries since $\boldsymbol{Z}$ is orthogonal (and Gaussians are rotationally invariant). So, $\boldsymbol{L}^T\boldsymbol{N} \overset{\text{dist}}{=} \boldsymbol{L}^T\boldsymbol{Z}\boldsymbol{D}\boldsymbol{S} \overset{\text{dist}}{=} \boldsymbol{G}_{d-m}^T\boldsymbol{D}\boldsymbol{G}$ and overall:

$$\begin{aligned}\boldsymbol{M} = \begin{bmatrix}\boldsymbol{N}^T\boldsymbol{N} \\ \boldsymbol{L}^T\boldsymbol{N}\end{bmatrix} &\overset{\text{dist}}{=} \begin{bmatrix}\boldsymbol{G}_m^T\boldsymbol{D}\boldsymbol{G}_m \\ \boldsymbol{G}_{d-m}^T\boldsymbol{D}\boldsymbol{G}_m\end{bmatrix} \\ &= \begin{bmatrix}\boldsymbol{G}_m\ \boldsymbol{G}_{d-m}\end{bmatrix}^T\boldsymbol{D}\boldsymbol{G}_m.\end{aligned}$$

□

Finally, we directly prove a lower bound on the number of samples m required to solve Problem B.2, and thus, via Lemma B.3, Problem B.1. Combined with Lemma B.1, this immediately yields our main lower bound on non-adaptive trace estimation, Theorem 4.2.

LEMMA B.4. *If $m < \frac{c}{\varepsilon}$ for a fixed constant c, then Problem B.2 cannot be solved with probability $\geq \frac{2}{3}$.*

Proof. The proof follows from existing work on lower bounds for learning "negatively spiked" covariance matrices [CMW15, PWBM18]. Let $\mathcal{P}$ be the distribution of $\boldsymbol{N}$ in Problem B.2, conditioned on $\boldsymbol{C} = \boldsymbol{I}$, and let $\mathcal{Q}$ be the distribution conditioned on $\boldsymbol{C} = \boldsymbol{I} - \mathbf{z}\mathbf{z}^T$. These distributions fall into the spiked covariance model of [PWBM18], specifically the negatively spiked Wishart model (see Defn. 5.1 in [PWBM18]) with spike size $\beta = -1$, and spike distribution $\mathcal{X}$ the uniform distribution over unit vectors in $\mathbb{R}^n$. Let $D_{\chi^2}(\mathcal{P}\|\mathcal{Q})$ denote the χ^2 divergence between $\mathcal{P}$ and $\mathcal{Q}$. Specifically,

$$D_{\chi^2}(\mathcal{Q}\|\mathcal{P}) = \int_{\boldsymbol{X}\in\mathbb{R}_{d\times m}} \left(\frac{\mathcal{Q}(\boldsymbol{X})}{\mathcal{P}(\boldsymbol{X})}\right)^2 \mathcal{P}(\boldsymbol{X})d\boldsymbol{X} - 1.$$

We have $D_{KL}(\mathcal{Q}\|\mathcal{P}) \leq D_{\chi^2}(\mathcal{Q}\|\mathcal{P})$, so to prove that $\mathcal{P}, \mathcal{Q}$ cannot be distinguished with good probability, it suffices to prove an upper bound on $D_{\chi^2}(\mathcal{Q}\|\mathcal{P})$. In [CMW15] (Lemma 7) it is proven that, letting $\mathbf{v}$ and $\mathbf{v}'$ be independent random unit vectors in $\mathbb{R}^n$,

$$D_{\chi^2}(\mathcal{Q}\|\mathcal{P}) = \mathop{\mathbb{E}}_{\mathbf{v},\mathbf{v}'}\left[\left(1-\langle\mathbf{v},\mathbf{v}'\rangle^2\right)^{-m/2}\right] - 1. \tag{B.2}$$

Equation (B.2) uses the notation of Prop. 5.11 in [PWBM18], which restates and proves a slightly less general form of the equality from [CMW15]. Our goal is to prove that the expectation term in (B.2) is $\leq 1+C$ for some small constant C when $m = \frac{c}{\varepsilon} = cn$ for a sufficiently small constant c.

We first note that $\langle\mathbf{v},\mathbf{v}'\rangle$ is identically distributed to $x \in [-1,1]$ where x is the first entry in a random unit vector in $\mathbb{R}^n$. It is well known that $\frac{x+1}{2}$ is distributed according to a beta distribution with parameters $\alpha = \beta = \frac{n-1}{2}$ [FKN90]. Specifically, this gives that x has density:

$$p(x) = \frac{\Gamma(2\alpha)}{2\Gamma(\alpha)^2}\cdot\left(\frac{1-x^2}{4}\right)^{\alpha-1}.$$

Plugging this density back in to the expectation term in (B.2) we obtain:

$$\begin{aligned}&\mathop{\mathbb{E}}_{\mathbf{v},\mathbf{v}'}\left[\left(1-\langle\mathbf{v},\mathbf{v}'\rangle^2\right)^{-m/2}\right] = \\ &= \int_{-1}^{1}\frac{\Gamma(2\alpha)}{2\Gamma(\alpha)^2}\cdot\left(\frac{1-x^2}{4}\right)^{\alpha-1}(1-x^2)^{-m/2}dx \\ &= \int_{-1}^{1}\frac{\Gamma(2\alpha)}{2\Gamma(\alpha)^2}\cdot\left(\frac{1}{4}\right)^{\alpha-1}(1-x^2)^{\alpha-1-m/2}dx\end{aligned}$$

Assume without loss of generality that n is an odd integer, and thus $\alpha = \frac{n-1}{2}$ is an integer. Let $m/2 = c\alpha$ for some constant $c \ll 1$ such that $c\alpha$ is an integer and thus $(1-c)\alpha$ is an integer. Then:

$$\begin{aligned}&\mathop{\mathbb{E}}_{\mathbf{v},\mathbf{v}'}\left[\left(1-\langle\mathbf{v},\mathbf{v}'\rangle^2\right)^{-m/2}\right] = \\ &= \frac{\frac{\Gamma(2\alpha)}{\Gamma(\alpha)^2}}{\frac{\Gamma(2(1-c)\alpha)}{\Gamma((1-c)\alpha)^2}}\cdot\left(\frac{1}{4}\right)^{c\alpha}\cdot \\ &\int_{-1}^{1}\frac{\Gamma(2(1-c)\alpha)}{2\Gamma((1-c)\alpha)^2}\left(\frac{1}{4}\right)^{(1-c)\alpha-1}(1-x^2)^{(1-c)\alpha-1}dx \\ &= \frac{\frac{\Gamma(2\alpha)}{\Gamma(\alpha)^2}}{\frac{\Gamma(2(1-c)\alpha)}{\Gamma((1-c)\alpha)^2}}\cdot\left(\frac{1}{4}\right)^{c\alpha},\end{aligned} \tag{B.3}$$

where the equality follows because the term being integrated is the density of x where $\frac{x+1}{2}$ is distributed according to a beta distribution with parameters $(1-c)\alpha, (1-c)\alpha$. Since we have chosen parameters such that α is a positive integer, we have:

$$\frac{\Gamma(2\alpha)}{\Gamma(\alpha)^2} = \frac{(2\alpha-1)!}{(\alpha-1)!(\alpha-1)!} = \frac{\alpha}{2} \cdot \binom{2\alpha}{\alpha}.$$

Similarly, $\frac{\Gamma(2(1-c)\alpha)}{\Gamma((1-c)\alpha)^2} = \frac{(1-c)\alpha}{2} \cdot \binom{2(1-c)\alpha}{(1-c)\alpha}$. Each of the binomial coefficients in these expressions is a central binomial coefficient (i.e., proportional to a Catalan number), and we can use well known methods like Stirling's approximation to bound them. In particular, we employ a bound given in Lemma 7 of [MS77], which gives $\frac{1}{2}\frac{4^z}{\sqrt{z}} \leq \binom{2z}{z} \leq \frac{1}{\sqrt{\pi}}\frac{4^z}{\sqrt{z}}$. for any integer z. Accordingly, we have

$$\begin{aligned}
\frac{\frac{\Gamma(2\alpha)}{\Gamma(\alpha)^2}}{\frac{\Gamma(2(1-c)\alpha)}{\Gamma((1-c)\alpha)^2}} &= \frac{1}{1-c} \cdot \frac{\binom{2\alpha}{\alpha}}{\binom{2(1-c)\alpha}{(1-c)\alpha}} \\
&\leq \frac{1}{1-c} \cdot \frac{1/\sqrt{\pi}}{1/2} \frac{4^\alpha}{4^{(1-c)\alpha}} \cdot \sqrt{\frac{(1-c)\alpha}{\alpha}} \\
&= \frac{2}{\sqrt{\pi(1-c)}} \cdot 4^{c\alpha}.
\end{aligned}$$

Plugging into (B.3) and requiring $c \leq .1$ we have:

$$\mathop{\mathbb{E}}_{\mathbf{v},\mathbf{v}'}\left[\left(1-\langle \mathbf{v}, \mathbf{v}'\rangle^2\right)^{-m/2}\right] \leq \frac{2}{\sqrt{\pi \cdot .9}} < \frac{6}{5}.$$

It follows that $D_{KL}(\mathcal{Q}\|\mathcal{P}) \leq D_{\chi^2}(\mathcal{Q}\|\mathcal{P}) \leq \frac{6}{5} - 1 = \frac{1}{5}$, and thus by Pinsker's inequality that

$$D_{TV}(\mathcal{Q}, \mathcal{P}) \leq \frac{1}{\sqrt{10}} < \frac{1}{3}.$$

Thus, no algorithm can solve Problem B.2 with probability $\geq \frac{1}{2} + \frac{1/3}{2} = \frac{2}{3}$, completing the lemma. ∎