Randomized Linear Algebra Approaches to Estimate the von Neumann Entropy of Density Matrices

Eugenia-Maria Kontopoulou, Gregory-Paul Dexter, Wojciech Szpankowski, *Fellow, IEEE*, Ananth Grama, and Petros Drineas

***Abstract*— The *von Neumann entropy*, named after John von Neumann, is an extension of the classical concept of entropy to the field of quantum mechanics. From a numerical perspective, von Neumann entropy can be computed simply by computing all eigenvalues of a density matrix, an operation that could be prohibitively expensive for large-scale density matrices. We present and analyze three randomized algorithms to approximate von Neumann entropy of real density matrices: our algorithms leverage recent developments in the Randomized Numerical Linear Algebra (RandNLA) literature, such as randomized trace estimators, provable bounds for the power method, and the use of random projections to approximate the eigenvalues of a matrix. All three algorithms come with provable accuracy guarantees and our experimental evaluations support our theoretical findings showing considerable speedup with small loss in accuracy.**

***Index Terms*— von Neumann entropy, randomized algorithms, randNLA, Taylor polynomials, Chebyshev polynomials, random projections**

I. Introduction

ENTROPY is a fundamental quantity in many areas of science and engineering. *von Neumann entropy*, named after John von Neumann, is an extension of classical entropy concepts to the field of quantum mechanics. Its foundations can be traced to von Neumann's work on *Mathematische Grundlagen der Quantenmechanik*.[1] In his work, Von Neumann introduced the notion of a *density matrix*, which facilitated extension of the tools of classical statistical mechanics to the quantum domain in order to develop a theory of quantum mechanics.

From a mathematical perspective (see Section I-A for details) the real density matrix $\mathbf{R}$ is a symmetric positive semidefinite matrix in $\mathbb{R}^{n\times n}$ with unit trace. Let p_i, $i = 1 \dots n$ be the eigenvalues of $\mathbf{R}$ in decreasing order; then, the entropy of $\mathbf{R}$ is defined as[2]

$$\mathcal{H}(\mathbf{R}) = -\sum_{i=1}^{n} p_i \ln p_i. \tag{1}$$

The above definition is a proper extension of both the Gibbs entropy and the Shannon entropy to the quantum case. It implies an obvious algorithm to compute $\mathcal{H}(\mathbf{R})$ by computing the eigendecomposition of $\mathbf{R}$; known algorithms for this task can be prohibitively expensive for large values of n, particularly when the matrix becomes dense [1]. For example, [2] describes an entangled two-photon state generated by spontaneous parametric down-conversion, which can result in a sparse and banded density matrix with $n \approx 10^8$.

Motivated by the high computational cost, we seek numerical algorithms that approximate the von Neumann entropy of large density matrices, e.g., symmetric positive definite matrices with unit trace, faster than the trivial $\mathcal{O}(n^3)$ approach. Our algorithms build upon recent developments in the field of Randomized Numerical Linear Algebra (RandNLA), an interdisciplinary research area that exploits randomization as a computational resource to develop improved algorithms for large-scale linear algebra problems. Indeed, our work here focuses at the intersection of RandNLA and information theory, delivering novel randomized linear algebra algorithms and related quality-of-approximation results for a fundamental information-theoretic metric.

A. Background

We focus on finite-dimensional function (state) spaces. In this setting, the density matrix $\mathbf{R}$ represents the statistical mixture of $k \leq n$ pure states, and has the form

$$\mathbf{R} = \sum_{i=1}^{k} p_i \boldsymbol{\psi}_i \boldsymbol{\psi}_i^T \in \mathbb{R}^{n\times n}. \tag{2}$$

The vectors $\boldsymbol{\psi}_i \in \mathbb{R}^n$ for $i = 1 \dots k$ represent the $k \leq n$ pure states and can be assumed to be pairwise orthogonal and normal, while p_i's correspond to the probability of each state and satisfy $p_i > 0$ and $\sum_{i=1}^{k} p_i = 1$. From a linear algebraic perspective, eqn. (2) can be rewritten as

$$\mathbf{R} = \boldsymbol{\Psi}\boldsymbol{\Sigma}_p\boldsymbol{\Psi}^T \in \mathbb{R}^{n\times n}, \tag{3}$$

Manuscript received May 20, 2018; revised January 12, 2020; accepted January 21, 2020. Date of publication February 6, 2020; date of current version July 14, 2020. The work of Eugenia-Maria Kontopoulou and Petros Drineas was partially supported by NSF 10001390, NSF FRG 1760353 and NSF CCF-BSF 1814041. The work of Wojciech Szpankowski and Ananth Grama was supported in part by the NSF Center for Science of Information (CSoI) under Grant CCF-0939370 and in part by the NSF under Grant CCF-1524312 and Grant NIH 1U01CA198941-01. This article was presented in part at the 2018 IEEE International Symposium on Information Theory. *(Corresponding author: Eugenia-Maria Kontopoulou.)*

The authors are with the Department of Computer Science, Purdue University, West Lafayette, IN 47906 USA (e-mail: ekontopo@purdue.edu; gdexter@purdue.edu; szpan@purdue.edu; ayg@purdue.edu; pdrineas@purdue.edu).

Communicated by M. M. Wilde, Associate Editor for Quantum Information Theory.

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIT.2020.2971991

[1] Originally published in German in 1932; published in English under the title *Mathematical Foundations of Quantum Mechanics* in 1955.

[2] $\mathbf{R}$ is symmetric positive semidefinite and thus all its eigenvalues are non-negative. If p_i is equal to zero we set $p_i \ln p_i$ to zero as well.

where $\mathbf{\Psi} \in \mathbb{R}^{n\times k}$ is the matrix whose columns are the vectors $\boldsymbol{\psi}_i$ and $\mathbf{\Sigma}_p \in \mathbb{R}^{k\times k}$ is a diagonal matrix whose entries are the (positive) p_i's. Given our assumptions for $\boldsymbol{\psi}_i$, $\mathbf{\Psi}^T\mathbf{\Psi} = \mathbf{I}$; also $\mathbf{R}$ is symmetric positive semidefinite with its eigenvalues equal to p_i and corresponding left/ right singular vectors equal to $\boldsymbol{\psi}_i$'s; and $\mathbf{tr}(\mathbf{R}) = \sum_{i=1}^{k} p_i = 1$. Notice that eqn. (3) essentially reveals the (thin) Singular Value Decomposition (SVD) [1] of $\mathbf{R}$. The Von Neumann entropy of $\mathbf{R}$, denoted by $\mathcal{H}(\mathbf{R})$ is equal to (see also eqn. (1))

$$\mathcal{H}(\mathbf{R}) = -\sum_{i:p_i>0} p_i \ln p_i = -\mathbf{tr}(\mathbf{R}\ln\mathbf{R}). \tag{4}$$

The second equality follows from the definition of matrix functions [3]. More precisely, we overload notation and consider the full SVD of $\mathbf{R}$, namely $\mathbf{R} = \mathbf{\Psi}\mathbf{\Sigma}_p\mathbf{\Psi}^T$, where $\mathbf{\Psi} \in \mathbb{R}^{n\times n}$ is an orthogonal matrix whose top k columns correspond to the k pure states and the bottom $n-k$ columns are chosen so that $\mathbf{\Psi}\mathbf{\Psi}^T = \mathbf{\Psi}^T\mathbf{\Psi} = \mathbf{I}_n$. Here $\mathbf{\Sigma}_p$ is a diagonal matrix whose bottom $n-k$ diagonal entries are set to zero. Let $h(x) = x\ln x$ for any $x > 0$ and let $h(0) = 0$. Then, using the cyclical property of the trace and the definition of $h(x)$,

$$\begin{aligned} -\sum_{i,p_i>0} p_i \ln p_i &= -\mathbf{tr}\left(\mathbf{\Psi} h(\mathbf{\Sigma}_p)\mathbf{\Psi}^T\right) \\ &= -\mathbf{tr}\left(h(\mathbf{R})\right) \\ &= -\mathbf{tr}(\mathbf{R}\ln\mathbf{R}). \end{aligned} \tag{5}$$

B. Trace Estimators

The following lemma appeared in [4] and is immediate from Theorem 5.2 in [5]. It implies an algorithm to approximate the trace of any symmetric positive semidefinite matrix $\mathbf{A}$ by computing inner products of the matrix with Gaussian random vectors.

Lemma 1: Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be a positive semi-definite matrix, let $0 < \epsilon < 1$ be an accuracy parameter, and let $0 < \delta < 1$ be a failure probability. If $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_s \in \mathbb{R}^n$ are independent random standard Gaussian vectors, then, for $s = \lceil 20\ln(2/\delta)/\epsilon^2 \rceil$, with probability at least $1-\delta$,

$$\left| \mathbf{tr}(\mathbf{A}) - \frac{1}{s}\sum_{i=1}^{s} \mathbf{g}_i^\top \mathbf{A}\mathbf{g}_i \right| \leq \epsilon \cdot \mathbf{tr}(\mathbf{A}).$$

C. Our Contributions

We present and analyze three randomized algorithms to approximate the von Neumann entropy of density matrices. The *first two* algorithms (Sections II and III) leverage two different polynomial approximations of the matrix function $\mathcal{H}(\mathbf{R}) = -\mathbf{tr}(\mathbf{R}\ln\mathbf{R})$: the first approximation uses a Taylor series expansion, while the second approximation uses Chebyschev polynomials. Both algorithms return, with high probability, relative-error approximations to the true entropy of the input density matrix, under certain assumptions. More specifically, in both cases, we need to assume that the input density matrix has n non-zero eigenvalues, or, equivalently, that the probabilities p_i, $i = 1\ldots n$, corresponding to the underlying n pure states are non-zero. The running time of both algorithms is proportional to the sparsity of the input density matrix and depends (see Theorems 2 and 4 for precise statements) on, roughly, the ratio of the largest to the smallest probability p_1/p_n (recall that the smallest probability is assumed to be non-zero), as well as the desired accuracy.

The *third* algorithm (Section V) is fundamentally different, if not orthogonal, to the previous two approaches. It leverages the power of random projections [6], [7] to approximate numerical linear algebra quantities, such as the eigenvalues of a matrix. Assuming that the density matrix $\mathbf{R}$ has exactly $k \ll n$ non-zero eigenvalues, e.g., there are k pure states with non-zero probabilities p_i, $i = 1\ldots k$, the proposed algorithm returns, with high probability, relative error approximations to all k probabilities p_i. This, in turn, implies an additive-relative error approximation to the entropy of the density matrix, which, under a mild assumption on the true entropy of the density matrix, becomes a relative error approximation (see Theorem 10 for a precise statement). The running time of the algorithm is again proportional to the sparsity of the density matrix and depends on the target accuracy, but, unlike the previous two algorithms, does not depend on any function of the p_i.

From a technical perspective, the theoretical analysis of the first two algorithms proceeds by combining the power of polynomial approximations, either using Taylor series or Chebyschev polynomials, to matrix functions, combined with randomized trace estimators. A provably accurate variant of the power method is used to estimate the largest probability p_1. If this estimate is significantly smaller than one, it can improve the running times of the proposed algorithms (see discussion after Theorem 2). The third algorithm leverages a powerful, multiplicative matrix perturbation result that first appeared in [8]. Our work in Section V is a novel application of this inequality to derive bounds for RandNLA algorithms.

Finally, in Section VI, we present a detailed evaluation of our algorithms on synthetic density matrices of various sizes, most of which were generated using Matlab's QETLAB toolbox [9]. For some of the larger matrices that were used in our evaluations, the exact computation of the entropy takes hours, whereas our algorithms return approximations with relative errors well below 0.5% in only a few minutes.

D. Prior Work

The first non-trivial algorithm to approximate the von Neumann entropy of a density matrix appeared in [2]. Their approach is essentially the same as our approach in Section III. Indeed, our algorithm in Section III was inspired by their approach. However, our analysis is somewhat different, leveraging a provably accurate variant of the power method, as well as provably accurate trace estimators to derive a relative error approximation to the entropy of a density matrix, under appropriate assumptions. A detailed, technical comparison between our results in Section III and the work of [2] is delegated to Section III-C.

Independently and in parallel with our work, [10] presented a multipoint interpolation algorithm (building upon [11]) to compute a relative error approximation for the entropy of a real

matrix with bounded condition number. The proposed running time of Theorem 35 of [10] does not depend on the condition number of the input matrix (i.e., the ratio of the largest to the smallest probability), which is a clear advantage in the case of ill-conditioned matrices. However, the dependence of the algorithm of Theorem 35 of [10] on terms like $(\log n/\epsilon)^6$ or $n^{1/3}\mathrm{nnz}(\mathbf{A}) + n\sqrt{\mathrm{nnz}(\mathbf{A})}$ (where $\mathrm{nnz}(\mathbf{A})$ represents the number of non-zero elements of the matrix $\mathbf{A}$) could blow up the running time of the proposed algorithm for reasonably conditioned matrices.

We also note the recent work in [4], which used Taylor approximations to matrix functions to estimate the log determinant of symmetric positive definite matrices (see also Section 1.2 of [4] for an overview of prior work on approximating matrix functions via Taylor series). The work of [12] used a Chebyschev polynomial approximation to estimate the log determinant of a matrix and is reminiscent of our approach in Section III and, of course, the work of [2].

We conclude this section by noting that our algorithms use two tools (described, for the sake of completeness, in the Appendix) that appeared in prior work. The first tool is the power method, with a provable analysis that first appeared in [13]. The second tool is a provably accurate trace estimation algorithm for symmetric positive semidefinite matrices that appeared in [5].

II. An Approach via Taylor Series

Our first approach to approximate the von Neumann entropy of a density matrix uses a Taylor series expansion to approximate the logarithm of a matrix, combined with a relative-error trace estimator for symmetric positive semi-definite matrices and the power method to upper bound the largest singular value of a matrix.

A. Algorithm and Main Theorem

Our main result is an analysis of Algorithm 1 (see below) that guarantees relative error approximation to the entropy of the density matrix $\mathbf{R}$, under the assumption that $\mathbf{R} = \sum_{i=1}^{n} p_i \boldsymbol{\psi}_i \boldsymbol{\psi}_i^T \in \mathbb{R}^{n\times n}$ has n pure states with $0 < \ell \leq p_i$ for all $i = 1 \ldots n$.

Algorithm 1 A Taylor Series Approach to Estimate the Entropy

1: **INPUT**: $\mathbf{R} \in \mathbb{R}^{n\times n}$, accuracy parameter $\varepsilon > 0$, failure probability δ, and integer $m > 0$.
2: Compute $\tilde{p}_1$, the estimate of the largest eigenvalue of $\mathbf{R}$, p_1, using Algorithm 8 (see Appendix) with $t = \mathcal{O}(\ln n)$ and $q = \mathcal{O}(\ln(1/\delta))$.
3: Set $u = \min\{1, 6\tilde{p}_1\}$.
4: Set $s = \left\lceil 20\ln(2/\delta)/\varepsilon^2 \right\rceil$.
5: Let $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_s \in \mathbb{R}^n$ be i.i.d. random Gaussian vectors.
6: **OUTPUT**: return

$$\widehat{\mathcal{H}}(\mathbf{R}) = \ln u^{-1} + \frac{1}{s}\sum_{i=1}^{s}\sum_{k=1}^{m}\frac{\mathbf{g}_i^\top \mathbf{R}(\mathbf{I}_n - u^{-1}\mathbf{R})^k \mathbf{g}_i}{k}.$$

The following theorem is our main quality-of-approximation result for Algorithm 1.

Theorem 2: Let $\mathbf{R}$ be a density matrix such that all probabilities p_i, $i = 1 \ldots n$ satisfy $0 < \ell \leq p_i$. Let u be computed as in Algorithm 1 and let $\widehat{\mathcal{H}}(\mathbf{R})$ be the output of Algorithm 1 on inputs $\mathbf{R}$, m, and $\epsilon < 1$; Then, with probability at least $1 - 2\delta$,

$$\left|\widehat{\mathcal{H}}(\mathbf{R}) - \mathcal{H}(\mathbf{R})\right| \leq 2\epsilon\mathcal{H}(\mathbf{R}),$$

by setting $m = \left\lceil \frac{u}{\ell}\ln\frac{1}{\epsilon}\right\rceil$. The algorithm runs in time

$$\mathcal{O}\left(\left(\frac{u}{\ell}\cdot\frac{\ln(1/\epsilon)}{\epsilon^2} + \ln(n)\right)\ln(1/\delta)\cdot\mathrm{nnz}(\mathbf{R})\right).$$

A few remarks are necessary to better understand the above theorem. First, ℓ could be set to p_n, the smallest of the probabilities corresponding to the n pure states of the density matrix $\mathbf{R}$. Second, it should be obvious that u in Algorithm 1 could be simply set to one and thus we could avoid calling Algorithm 8 to estimate p_1 by $\tilde{p}_1$ and thus compute u. However, if p_1 is small, then u could be significantly smaller than one, thus reducing the running time of Algorithm 1, which depends on the ratio u/ℓ. Third, ideally, if both p_1 and p_n were used instead of u and ℓ, respectively, the running time of the algorithm would scale with the ratio p_1/p_n.

B. Proof of Theorem 2

We now prove Theorem 2, which analyzes the performance of Algorithm 1. Our first lemma presents a simple expression for $\mathcal{H}(\mathbf{R})$ using a Taylor series expansion.

Lemma 3: Let $\mathbf{R} \in \mathbb{R}^{n\times n}$ be a symmetric positive definite matrix with unit trace and whose eigenvalues lie in the interval $[\ell, u]$, for some $0 < \ell \leq u \leq 1$. Then,

$$\mathcal{H}(\mathbf{R}) = \ln u^{-1} + \sum_{k=1}^{\infty}\frac{\mathbf{tr}\left(\mathbf{R}(\mathbf{I}_n - u^{-1}\mathbf{R})^k\right)}{k}.$$

Proof: From the definition of the von Neumann entropy and a Taylor expansion,

$$\begin{aligned}\mathcal{H}(\mathbf{R}) &= -\mathbf{tr}\left(\mathbf{R}\ln\left(uu^{-1}\mathbf{R}\right)\right)\\ &= -\mathbf{tr}\left((\ln u)\mathbf{R}\right) - \mathbf{tr}\left(\mathbf{R}\ln(\mathbf{I}_n - (\mathbf{I}_n - u^{-1}\mathbf{R}))\right)\\ &= \ln u^{-1} - \mathbf{tr}\left(-\mathbf{R}\sum_{k=1}^{\infty}\frac{(\mathbf{I}_n - u^{-1}\mathbf{R})^k}{k}\right) \qquad (6)\\ &= \ln u^{-1} + \sum_{k=1}^{\infty}\frac{\mathbf{tr}\left(\mathbf{R}(\mathbf{I}_n - u^{-1}\mathbf{R})^k\right)}{k}.\end{aligned}$$

Eqn. (6) follows since $\mathbf{R}$ has unit trace and from a Taylor expansion: indeed, $\ln(\mathbf{I}_n - \mathbf{A}) = -\sum_{k=1}^{\infty}\mathbf{A}^k/k$ for a symmetric matrix $\mathbf{A}$ whose eigenvalues are all in the interval $(-1, 1)$. We note that the eigenvalues of $\mathbf{I}_n - u^{-1}\mathbf{R}$ are in the interval $[0, 1 - (\ell/u)]$, whose upper bound is strictly less than one since, by our assumptions, $\ell/u > 0$. □

We now proceed to prove Theorem 2. We will condition our analysis on Algorithm 8 being successful, which happens with probability at least $1 - \delta$. In this case, $u = \min\{1, 6\tilde{p}_1\}$ is an upper bound for all probabilities p_i.

For notational convenience, set $\mathbf{C} = \mathbf{I}_n - u^{-1}\mathbf{R}$. We start by manipulating $\Delta = \left|\widehat{\mathcal{H}}(\mathbf{R}) - \mathcal{H}(\mathbf{R})\right|$ as follows:

$$\begin{aligned}\Delta &= \left|\sum_{k=1}^{m}\frac{1}{k}\cdot\frac{1}{s}\sum_{i=1}^{s}\mathbf{g}_i^\top\mathbf{R}\mathbf{C}^k\mathbf{g}_i - \sum_{k=1}^{\infty}\frac{1}{k}\mathbf{tr}\left(\mathbf{R}\mathbf{C}^k\right)\right| \\ &\le \left|\sum_{k=1}^{m}\frac{1}{k}\cdot\frac{1}{s}\sum_{i=1}^{s}\mathbf{g}_i^\top\mathbf{R}\mathbf{C}^k\mathbf{g}_i - \sum_{k=1}^{m}\frac{1}{k}\mathbf{tr}\left(\mathbf{R}\mathbf{C}^k\right)\right| + \left|\sum_{k=m+1}^{\infty}\frac{1}{k}\mathbf{tr}\left(\mathbf{R}\mathbf{C}^k\right)\right| \\ &= \underbrace{\left|\frac{1}{s}\sum_{i=1}^{s}\mathbf{g}_i^\top\left(\sum_{k=1}^{m}\mathbf{R}\mathbf{C}^k/k\right)\mathbf{g}_i - \mathbf{tr}\left(\sum_{k=1}^{m}\frac{1}{k}\mathbf{R}\mathbf{C}^k\right)\right|}_{\Delta_1} + \underbrace{\left|\sum_{k=m+1}^{\infty}\mathbf{tr}\left(\mathbf{R}\mathbf{C}^k\right)/k\right|}_{\Delta_2}.\end{aligned}$$

We now bound the two terms Δ_1 and Δ_2 separately. We start with Δ_1: the idea is to apply Lemma 1 on the matrix $\sum_{k=1}^{m}\mathbf{R}\mathbf{C}^k/k$ with $s = \lceil 20\ln(2/\delta)/\epsilon^2\rceil$. Hence, with probability at least $1-\delta$:

$$\Delta_1 \le \epsilon\cdot\mathbf{tr}\left(\sum_{k=1}^{m}\mathbf{R}\mathbf{C}^k/k\right) \le \epsilon\cdot\mathbf{tr}\left(\sum_{k=1}^{\infty}\mathbf{R}\mathbf{C}^k/k\right). \quad (7)$$

A subtle point in applying Lemma 1 is that the matrix $\sum_{k=1}^{m}\mathbf{R}\mathbf{C}^k/k$ must be symmetric positive semidefinite. To prove this, let the SVD of $\mathbf{R}$ be $\mathbf{R} = \mathbf{\Psi}\mathbf{\Sigma}_p\mathbf{\Psi}^T$, where all three matrices are in $\mathbb{R}^{n\times n}$ and the diagonal entries of $\mathbf{\Sigma}_p$ are in the interval $[\ell, u]$. Then, it is easy to see that $\mathbf{C} = \mathbf{I}_n - u^{-1}\mathbf{R} = \mathbf{\Psi}(\mathbf{I}_n - u^{-1}\mathbf{\Sigma}_p)\mathbf{\Psi}^T$ and $\mathbf{R}\mathbf{C}^k = \mathbf{\Psi}\mathbf{\Sigma}_p(\mathbf{I}_n - u^{-1}\mathbf{\Sigma}_p)^k\mathbf{\Psi}^T$, where the diagonal entries of $\mathbf{I}_n - u^{-1}\mathbf{\Sigma}_p$ are non-negative, since the largest entry in $\mathbf{\Sigma}_p$ is upper bounded by u. This proves that $\mathbf{R}\mathbf{C}^k$ is symmetric positive semidefinite for any k, a fact which will be useful throughout the proof. Now,

$$\sum_{k=1}^{m}\mathbf{R}\mathbf{C}^k/k = \mathbf{\Psi}\left(\mathbf{\Sigma}_p\sum_{k=1}^{m}(\mathbf{I}_n - u^{-1}\mathbf{\Sigma}_p)^k/k\right)\mathbf{\Psi}^T,$$

which shows that the matrix of interest is symmetric positive semidefinite. Additionally, since $\mathbf{R}\mathbf{C}^k$ is symmetric positive semidefinite, its trace is non-negative, which proves the second inequality in eqn. (7) as well.

We proceed to bound Δ_2 as follows:

$$\begin{aligned}\Delta_2 &= \left|\sum_{k=m+1}^{\infty}\mathbf{tr}\left(\mathbf{R}\mathbf{C}^k\right)/k\right| \\ &= \left|\sum_{k=m+1}^{\infty}\mathbf{tr}\left(\mathbf{R}\mathbf{C}^m\mathbf{C}^{k-m}\right)/k\right| \\ &= \left|\sum_{k=m+1}^{\infty}\mathbf{tr}\left(\mathbf{C}^m\mathbf{C}^{k-m}\mathbf{R}\right)/k\right| \\ &\le \left|\sum_{k=m+1}^{\infty}\|\mathbf{C}^m\|_2\cdot\mathbf{tr}\left(\mathbf{C}^{k-m}\mathbf{R}\right)/k\right| \quad (8) \\ &= \|\mathbf{C}^m\|_2\cdot\left|\sum_{k=m+1}^{\infty}\mathbf{tr}\left(\mathbf{R}\mathbf{C}^{k-m}\right)/k\right| \\ &\le \|\mathbf{C}^m\|_2\cdot\left|\sum_{k=1}^{\infty}\mathbf{tr}\left(\mathbf{R}\mathbf{C}^k\right)/k\right| \quad (9) \\ &\le \left(1-\frac{\ell}{u}\right)^m\sum_{k=1}^{\infty}\mathbf{tr}\left(\mathbf{R}\mathbf{C}^k\right)/k. \quad (10)\end{aligned}$$

To prove eqn. (8), we used von Neumann's trace inequality.[3] Eqn. (8) now follows since $\mathbf{C}^{k-m}\mathbf{R}$ is symmetric positive semidefinite.[4] To prove eqn. (9), we used the fact that $\mathbf{tr}\left(\mathbf{R}\mathbf{C}^k\right)/k \ge 0$ for any $k \ge 1$. Finally, to prove eqn. (10), we used the fact that $\|\mathbf{C}\|_2 = \|\mathbf{I}_n - u^{-1}\mathbf{\Sigma}_p\|_2 \le 1 - \ell/u$ since the smallest entry in Σ_p is at least ℓ by our assumptions. We also removed unnecessary absolute values since $\mathbf{tr}\left(\mathbf{R}\mathbf{C}^k\right)/k$ is non-negative for any positive integer k.

Combining the bounds for Δ_1 and Δ_2 gives

$$\left|\widehat{\mathcal{H}}(\mathbf{R}) - \mathcal{H}(\mathbf{R})\right| \le \left(\epsilon + \left(1-\frac{\ell}{u}\right)^m\right)\sum_{k=1}^{\infty}\frac{\mathbf{tr}\left(\mathbf{R}\mathbf{C}^k\right)}{k}.$$

We have already proven in Lemma 3 that

$$\sum_{k=1}^{\infty}\frac{\mathbf{tr}\left(\mathbf{R}\mathbf{C}^k\right)}{k} \le \mathcal{H}(\mathbf{R}) - \ln u^{-1} \le \mathcal{H}(\mathbf{R}),$$

where the last inequality follows since $u \le 1$. Collecting our results, we get

$$\left|\widehat{\mathcal{H}}(\mathbf{R}) - \mathcal{H}(\mathbf{R})\right| \le \left(\epsilon + \left(1-\frac{\ell}{u}\right)^m\right)\mathcal{H}(\mathbf{R}).$$

Setting

$$m = \left\lceil\frac{u}{\ell}\ln\frac{1}{\epsilon}\right\rceil$$

and using $\left(1-x^{-1}\right)^x \le e^{-1}$ $(x > 0)$, guarantees that $(1 - \ell/u)^m \le \epsilon$ and concludes the proof of the theorem. We note that the failure probability of the algorithm is at most 2δ (the sum of the failure probabilities of the power method and the trace estimation algorithm).

Finally, we discuss the running time of Algorithm 1, which is equal to $\mathcal{O}(s\cdot m\cdot\mathrm{nnz}(\mathbf{R}))$. Since $s = \mathcal{O}\left(\frac{\ln(1/\delta)}{\epsilon^2}\right)$ and $m = \mathcal{O}\left(\frac{u\ln(1/\epsilon)}{\ell}\right)$, the running time becomes (after accounting for the running time of Algorithm 8)

$$\mathcal{O}\left(\left(\frac{u}{\ell}\cdot\frac{\ln(1/\epsilon)}{\epsilon^2} + \ln(n)\right)\ln(1/\delta)\cdot\mathrm{nnz}(\mathbf{R})\right).$$

III. An Approach via Chebyschev Polynomials

Our second approach is to use a Chebyschev polynomial-based approximation scheme to estimate the entropy of a density matrix. Our approach follows the work of [2], but our analysis uses the trace estimators of [5] and Algorithm 8 and its analysis. Importantly, we present conditions under which the proposed approach is competitive with the approach of Section II.

[3] Indeed, for any two matrices $\mathbf{A}$ and $\mathbf{B}$, $\mathbf{tr}(\mathbf{AB}) \le \sum_i \sigma_i(\mathbf{A})\sigma_i(\mathbf{B})$, where $\sigma_i(\mathbf{A})$ (respectively $\sigma_i(\mathbf{B})$) denotes the i-th singular value of $\mathbf{A}$ (respectively $\mathbf{B}$). Let $\|\cdot\|_2$ to denote the induced-2 matrix or spectral norm, then $\|\mathbf{A}\|_2 = \sigma_1(\mathbf{A})$ (its largest singular value). Given that each singular value of $\mathbf{A}$ is upper bounded by $\sigma_1(\mathbf{A})$ then we can rewrite $\mathbf{tr}(\mathbf{AB}) \le \|\mathbf{A}\|_2\sum_i\sigma_i(\mathbf{B})$; if $\mathbf{B}$ is symmetric positive semidefinite, $\mathbf{tr}(\mathbf{B}) = \sum_i\sigma_i(\mathbf{B})$.

[4] This can be proven using an argument similar to the one used to prove eqn. (7).

A. Algorithm and Main Theorem

The proposed algorithm leverages the fact that the von Neumann entropy of a density matrix $\mathbf{R}$ is equal to the (negative) trace of the matrix function $\mathbf{R}\ln\mathbf{R}$ and approximates the function $\mathbf{R}\ln\mathbf{R}$ by a sum of Chebyschev polynomials; then, the trace of the resulting matrix is estimated using the trace estimator of [5].

Let $f_m(x) = \sum_{w=0}^{m} \alpha_w \mathcal{T}_w(x)$ with $\alpha_0 = \frac{u}{2}\left(\ln\frac{u}{4}+1\right)$, $\alpha_1 = \frac{u}{4}\left(2\ln\frac{u}{4}+3\right)$, and $\alpha_w = \frac{(-1)^w u}{w^3-w}$ for $w \geq 2$. Let $\mathcal{T}_w(x) = \cos(w \cdot \arccos((2/u)x-1))$ and $x \in [0,u]$ be the Chebyschev polynomials of the first kind for any integer $w > 0$. Algorithm 2 computes u (an upper bound estimate for the largest probability p_1 of the density matrix $\mathbf{R}$) and then computes $f_m(\mathbf{R})$ and estimates its trace. We note that the computation $\mathbf{g}_i^\top f_m(\mathbf{R})\mathbf{g}_i$ can be done efficiently using Clenshaw's algorithm; see Appendix C for the well-known approach.

Algorithm 2 A Chebyschev Polynomial-Based Approach to Estimate the Entropy

1: **INPUT**: $\mathbf{R} \in \mathbb{R}^{n\times n}$, accuracy parameter $\varepsilon > 0$, failure probability δ, and integer $m > 0$.
2: Compute $\tilde{p}_1$, the estimate of the largest eigenvalue of $\mathbf{R}$, p_1, using Algorithm 8 (see Appendix) with $t = \mathcal{O}(\ln n)$ and $q = \mathcal{O}(\ln(1/\delta))$.
3: Set $u = \min\{1, 6\tilde{p}_1\}$.
4: Set $s = \lceil 20\ln(2/\delta)/\varepsilon^2 \rceil$.
5: Let $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_s \in \mathbb{R}^n$ be i.i.d. random Gaussian vectors.
6: **OUTPUT:** $\widehat{\mathcal{H}}(\mathbf{R}) = -\frac{1}{s}\sum_{i=1}^{s} \mathbf{g}_i^\top f_m(\mathbf{R})\mathbf{g}_i$.

Our main result is an analysis of Algorithm 2 that guarantees a relative error approximation to the entropy of the density matrix $\mathbf{R}$, under the assumption that $\mathbf{R} = \sum_{i=1}^{n} p_i \boldsymbol{\psi}_i \boldsymbol{\psi}_i^T \in \mathbb{R}^{n\times n}$ has n pure states with $0 < \ell \leq p_i$ for all $i = 1\ldots n$. The following theorem is our main quality-of-approximation result for Algorithm 2.

Theorem 4: Let $\mathbf{R}$ be a density matrix such that all probabilities p_i, $i = 1\ldots n$ satisfy $0 < \ell \leq p_i$. Let u be computed as in Algorithm 1 and let $\widehat{\mathcal{H}}(\mathbf{R})$ be the output of Algorithm 2 on inputs $\mathbf{R}$, m, and $\epsilon < 1$; Then, with probability at least $1-2\delta$,

$$\left|\widehat{\mathcal{H}}(\mathbf{R}) - \mathcal{H}(\mathbf{R})\right| \leq 3\epsilon\mathcal{H}(\mathbf{R}),$$

by setting $m = \sqrt{\frac{u}{2e\ell\ln(1/(1-\ell))}}$. The algorithm runs in time

$$\mathcal{O}\left(\left(\sqrt{\frac{u}{\ell\ln(1/(1-\ell))}}\cdot\frac{1}{\epsilon^{2.5}} + \ln(n)\right)\ln(1/\delta)\cdot \mathrm{nnz}(\mathbf{R})\right).$$

The similarities between Theorems 2 and 4 are obvious: same assumptions and directly comparable accuracy guarantees. The only difference is in the running times: the Taylor series approach has a milder dependency on ϵ, while the Chebyschev-based approximation has a milder dependency on the ratio u/ℓ, which controls the behavior of the probabilities p_i. However, for small values of ℓ ($\ell \to 0$),

$$\ln\frac{1}{1-\ell} = \ln\left(1+\frac{\ell}{1-\ell}\right) \approx \frac{\ell}{1-\ell} \approx \ell.$$

Thus, the Chebyschev-based approximation has a milder dependency on u but not necessarily ℓ when compared to the Taylor-series approach. We also note that the discussion following Theorem 2 is again applicable here.

B. Proof of Theorem 4

We will condition our analysis on Algorithm 8 being successful, which happens with probability at least $1-\delta$. In this case, $u = \min\{1, 6\tilde{p}_1\}$ is an upper bound for all probabilities p_i. We now recall (from Section I-A) the definition of the function $h(x) = x\ln x$ for any real $x \in (0,1]$, with $h(0) = 0$. Let $\mathbf{R} = \boldsymbol{\Psi}\boldsymbol{\Sigma}_p\boldsymbol{\Psi}^T \in \mathbb{R}^{n\times n}$ be the density matrix, where both $\boldsymbol{\Sigma}_p$ and $\boldsymbol{\Psi}$ are matrices in $\mathbb{R}^{n\times n}$. Notice that the diagonal entries of $\boldsymbol{\Sigma}_p$ are the p_is and they satisfy $0 < \ell \leq p_i \leq u \leq 1$ for all $i = 1\ldots n$.

Using the definitions of matrix functions from [3], we can now define $h(\mathbf{R}) = \boldsymbol{\Psi}h(\boldsymbol{\Sigma}_p)\boldsymbol{\Psi}^T$, where $h(\boldsymbol{\Sigma}_p)$ is a diagonal matrix in $\mathbb{R}^{n\times n}$ with entries equal to $h(p_i)$ for all $i = 1\ldots n$. We now restate Proposition 3.1 from [2] in the context of our work, using our notation.

Lemma 5: The function $h(x)$ in the interval $[0,u]$ can be approximated by

$$f_m(x) = \sum_{w=0}^{m} \alpha_w \mathcal{T}_w(x),$$

where $\alpha_0 = \frac{u}{2}\left(\ln\frac{u}{4}+1\right)$, $\alpha_1 = \frac{u}{4}\left(2\ln\frac{u}{4}+3\right)$, and $\alpha_w = \frac{(-1)^w u}{w^3-w}$ for $w \geq 2$. For any $m \geq 1$,

$$|h(x) - f_m(x)| \leq \frac{u}{2m(m+1)} \leq \frac{u}{2m^2},$$

for $x \in [0,u]$.

In the above, $\mathcal{T}_w(x) = \cos(w\cdot\arccos((2/u)x-1))$ for any integer $w \geq 0$ and $x \in [0,u]$. Notice that the function $(2/u)x - 1$ essentially maps the interval $[0,u]$, which is the interval of interest for the function $h(x)$, to $[-1,1]$, which is the interval over which Chebyschev polynomials are commonly defined. The above theorem exploits the fact that the Chebyschev polynomials form an orthonormal basis for the space of functions over the interval $[-1,1]$.

We now move on to approximate the entropy $\mathcal{H}(\mathbf{R})$ using the function $f_m(x)$. First,

$$\begin{aligned}
-\mathbf{tr}\left(f_m(\mathbf{R})\right) &= -\mathbf{tr}\left(\sum_{w=0}^{m}\alpha_w\mathcal{T}_w(\mathbf{R})\right)\\
&= -\mathbf{tr}\left(\sum_{w=0}^{m}\alpha_w\boldsymbol{\Psi}\mathcal{T}_w(\boldsymbol{\Sigma}_p)\boldsymbol{\Psi}^T\right)\\
&= -\sum_{w=0}^{m}\alpha_w\mathbf{tr}\left(\mathcal{T}_w(\boldsymbol{\Sigma}_p)\right)\\
&= -\sum_{w=0}^{m}\alpha_w\sum_{i=1}^{n}\mathcal{T}_w(p_i)\\
&= -\sum_{i=1}^{n}\sum_{w=0}^{m}\alpha_w\mathcal{T}_w(p_i). \qquad (11)
\end{aligned}$$

Recall from Section I-A that $\mathcal{H}(\mathbf{R}) = -\sum_{i=1}^{n} h(p_i)$. We can now bound the difference between $\mathbf{tr}\left(-f_m(\mathbf{R})\right)$ and $\mathcal{H}(\mathbf{R})$. Indeed,

$$\begin{aligned}\left|\mathcal{H}(\mathbf{R}) - \mathbf{tr}\left(-f_m(\mathbf{R})\right)\right| &= \left|-\sum_{i=1}^{n} h(p_i) + \sum_{i=1}^{n}\sum_{w=0}^{m} \alpha_w \mathcal{T}_w(p_i)\right| \\ &\leq \sum_{i=1}^{n}\left|h(p_i) - \sum_{w=0}^{m} \alpha_w \mathcal{T}_w(p_i)\right| \\ &\leq \frac{nu}{2m^2}. \end{aligned} \tag{12}$$

The last inequality follows by the final bound in Lemma 5, since all p_i's are in the interval $[0, u]$.

Recall that we also assumed that all p_is are lower-bounded by $\ell > 0$ and thus

$$\mathcal{H}(\mathbf{R}) = \sum_{i=1}^{n} p_i \ln\frac{1}{p_i} \geq n\ell \ln\frac{1}{1-\ell}. \tag{13}$$

We note that the upper bound on the p_is follows since the smallest p_i is at least $\ell > 0$ and thus the largest p_i cannot exceed $1 - \ell < 1$. We note that we cannot use the upper bound u in the above formula, since u could be equal to one; $1 - \ell$ is always strictly less than one but it cannot be a priori computed (and thus cannot be used in Algorithm 2), since ℓ is not a priori known.

We can now restate the bound of eqn. (12) as follows:

$$\begin{aligned}\left|\mathcal{H}(\mathbf{R}) - \mathbf{tr}\left(-f_m(\mathbf{R})\right)\right| &\leq \frac{u}{2m^2\ell\ln(1/(1-\ell))}\mathcal{H}(\mathbf{R}) \\ &\leq \epsilon\mathcal{H}(\mathbf{R}), \end{aligned} \tag{14}$$

where the last inequality follows by setting

$$m = \sqrt{\frac{u}{2\epsilon\ell\ln(1/(1-\ell))}}. \tag{15}$$

Next, we argue that the matrix $-f_m(\mathbf{R})$ is symmetric positive semidefinite (under our assumptions) and thus one can apply Lemma 1 to estimate its trace. We note that

$$-f_m(\mathbf{R}) = \mathbf{\Psi}\left(-f_m(\mathbf{\Sigma}_p)\right)\mathbf{\Psi}^T,$$

which trivially proves the symmetry of $-f_m(\mathbf{R})$ and also shows that its eigenvalues are equal to $-f_m(p_i)$ for all $i = 1 \ldots n$. We now bound

$$\begin{aligned}\left|\left(-f_m(p_i)\right) - p_i \ln\frac{1}{p_i}\right| &= \left|-f_m(p_i) + p_i \ln p_i\right| \\ &= \left|p_i \ln p_i - f_m(p_i)\right| \\ &\leq \frac{u}{2m^2} \leq \epsilon\ell\ln\frac{1}{1-\ell}, \end{aligned}$$

where the inequalities follow from Lemma 5 and our choice for m from eqn. (15). This inequality holds for all $i = 1 \ldots n$ and implies that

$$-f_m(p_i) \geq p_i \ln\frac{1}{p_i} - \epsilon\ell\ln\frac{1}{1-\ell} \geq (1-\epsilon)\ell\ln\frac{1}{1-\ell},$$

using our upper ($1 - \ell < 1$) and lower ($\ell > 0$) bounds on the p_is. Now $\epsilon \leq 1$ proves that $-f_m(p_i)$ are non-negative for all $i = 1 \ldots n$ and thus $-f_m(\mathbf{R})$ is a symmetric positive semidefinite matrix; it follows that its trace is also non-negative.

We can now apply the trace estimator of Lemma 1 to get

$$\left|\mathbf{tr}\left(-f_m(\mathbf{R})\right) - \left(-\frac{1}{s}\sum_{i=1}^{s} \mathbf{g}_i^\top f_m(\mathbf{R})\mathbf{g}_i\right)\right| \leq \epsilon \cdot \mathbf{tr}\left(-f_m(\mathbf{R})\right). \tag{16}$$

For the above bound to hold, we need to set

$$s = \left\lceil 20\ln(2/\delta)/\epsilon^2 \right\rceil. \tag{17}$$

We now conclude as follows:

$$\begin{aligned}\left|\mathcal{H}(\mathbf{R}) - \widehat{\mathcal{H}}(\mathbf{R})\right| \leq\ & \left|\mathcal{H}(\mathbf{R}) - \mathbf{tr}\left(-f_m(\mathbf{R})\right)\right| \\ &+ \left|\mathbf{tr}\left(-f_m(\mathbf{R})\right) - \left(-\frac{1}{s}\sum_{i=1}^{s} \mathbf{g}_i^\top f_m(\mathbf{R})\mathbf{g}_i\right)\right| \\ \leq\ & \epsilon\mathcal{H}(\mathbf{R}) + \epsilon\mathbf{tr}\left(-f_m(\mathbf{R})\right) \\ \leq\ & \epsilon\mathcal{H}(\mathbf{R}) + \epsilon(1+\epsilon)\mathcal{H}(\mathbf{R}) \\ \leq\ & 3\epsilon\mathcal{H}(\mathbf{R}). \end{aligned}$$

The first inequality follows by adding and subtracting $-\mathbf{tr}\left(f_m(\mathbf{R})\right)$ and using sub-additivity of the absolute value; the second inequality follows by eqns. (14) and (16); the third inequality follows again by eqn. (14); and the last inequality follows by using $\epsilon \leq 1$.

We note that the failure probability of the algorithm is at most 2δ (the sum of the failure probabilities of the power method and the trace estimation algorithm). Finally, we discuss the running time of Algorithm 2, which is equal to $\mathcal{O}(s \cdot m \cdot \mathrm{nnz}(\mathbf{R}))$. Using the values for m and s from eqns. (15) and (17), the running time becomes (after accounting for the running time of Algorithm 8)

$$\mathcal{O}\left(\left(\sqrt{\frac{u}{\ell\ln(1/(1-\ell))}} \cdot \frac{1}{\epsilon^{2.5}} + \ln(n)\right)\ln(1/\delta) \cdot \mathrm{nnz}(\mathbf{R})\right).$$

C. A Comparison With the Results of [2]

The work of [2] culminates in the error bounds described in Theorem 4.3 (and the ensuing discussion). In our parlance, [2] first derives the error bound of eqn. (12). It is worth emphasizing that the bound of eqn. (12) holds even if the p_is are not necessarily strictly positive, as assumed by Theorem 4: the bound holds even if some of the p_is are equal to zero.

Unfortunately, without imposing a lower bound assumption on the p_is it is difficult to get a meaningful error bound and an efficient algorithm. Indeed, the error implied by eqn. (12) (without any assumption on the p_is) necessitates setting m to at least $\Omega(\sqrt{n})$ (perhaps up to a logarithmic factor, as we will discuss shortly). To understand this, note that the entropy of the density matrix $\mathbf{R}$ ranges between zero and $\ln k$, where k is the rank of the matrix $\mathbf{R}$, i.e., the number of non-zero p_i's. Clearly, $k \leq n$ and thus $\ln n$ is an upper bound for $\mathcal{H}(\mathbf{R})$. Notice that if $\mathcal{H}(\mathbf{R})$ is smaller than $n/(2m^2)$, the error bound of eqn. (12) does not even guarantee that the resulting approximation will be positive, which is, of course, meaningless as an approximation to the entropy.

In order to guarantee a relative error bound of the form $\epsilon\mathcal{H}(\mathbf{R})$ via eqn. (12), we need to set m to be at least

$$m \geq \sqrt{\frac{n}{2\epsilon\mathcal{H}(\mathbf{R})}}, \tag{18}$$

which even for "large" values of $\mathcal{H}(\mathbf{R})$ (i.e., values close to the upper bound $\ln n$) still implies that m is $\mathcal{O}(\epsilon^{-1/2}\sqrt{n/\ln n})$. Even with such a large value for m, we are still not done: we need an efficient trace estimation procedure for the matrix $-f_m(\mathbf{R})$. While this matrix is always symmetric, it is not necessarily positive or negative semi-definite (unless additional assumptions are imposed on the p_is, like we did in Theorem 4).

IV. Approaches for Hermitian Density Matrices

Hermitian, instead of symmetric, positive definite matrices, frequently arise in quantum mechanics. The analyses of Sections II and III focus on real density matrices; we now briefly discuss how they can be extended to Hermitian density matrices. Recall that both approaches follow the same algorithmic scheme. First, the dominant eigenvalue of the density matrix is estimated via the power method; a trace estimation follows using Gaussian trace estimators on either the truncated Taylor expansion of a suitable matrix function or on a Chebyshev polynomial approximation of the same matrix function. Interestingly, the Taylor expansions, as well as the Chebyshev polynomial approximations, both work when the input matrix is complex. However, the estimation of the dominant eigenvalue of $\mathbf{R}$ poses a theoretical difficulty: to the best of our knowledge, there is no known bound for the accuracy of the power method in the case where $\mathbf{R}$ is complex. Lemma 14 guarantees relative error approximations to the dominant eigenvalue of real matrices, but we are not aware of any provable relative error bound for the complex case. To avoid this issue we will be using one as a (loose) upper bound for the dominant eigenvalue.

The crucial step in order to guarantee relative error approximations to the entropy of a Hermitian positive definite matrix is to guarantee relative error approximations for the trace of a Hermitian positive definite matrix. Lemma 1 assumes symmetric positive semi-definite matrices; we now prove that the same lemma can be applied on Hermitian positive definite matrices to achieve the same guarantees.

Theorem 6: Every Hermitian matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$ can be expressed as

$$\mathbf{A} = \mathbf{B} + i\mathbf{C}, \tag{19}$$

where $\mathbf{B} \in \mathbb{R}^{n\times n}$ is symmetric and $\mathbf{C} \in \mathbb{R}^{n\times n}$ is anti-symmetric (or skew-symmetric). If $\mathbf{A} \in \mathbb{C}^{n\times n}$ is positive semi-definite, then $\mathbf{B}$ is also positive semi-definite.

Proof: The proof is trivial and uses the fact that for any Hermitian (symmetric) positive semi-definite matrix all eigenvalues are real and greater than zero. □

Theorem 7: The trace of a Hermitian matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$ expressed as in eqn. (19) is equal to the trace of its real part:

$$\mathbf{tr}(\mathbf{A}) = \mathbf{tr}(\mathbf{B}).$$

Proof: Using $\mathbf{tr}(\mathbf{A}) = \mathbf{tr}\left(\mathbf{A}^\top\right)$, it is easy to see that

$$\begin{aligned}\mathbf{tr}(\mathbf{A}) &= \mathbf{tr}(\mathbf{B}+i\mathbf{C}) = \mathbf{tr}(\mathbf{B}) + i\mathbf{tr}(\mathbf{C}) \\ &= \mathbf{tr}\left(\mathbf{B}^\top\right) + i\mathbf{tr}\left(\mathbf{C}^\top\right) = \mathbf{tr}(\mathbf{B}).\end{aligned}$$

The last equality follows by noticing that the only way for the equality to hold for a skew-symmetric matrix $\mathbf{C}$ is if $\mathbf{tr}\left(\mathbf{C}^\top\right) = -\mathbf{tr}\left(\mathbf{C}^\top\right)$. This is true only if $\mathbf{C}$ is the all-zeros matrix. □

In words, Theorem 7 states that the trace of a Hermitian matrix equals the trace of its real part. Similarly, Theorem 6 states that the real part of a Hermitian positive semi-definite matrix is symmetric positive semi-definite. Combining both theorems we conclude that we can estimate the trace of a Hermitian positive definite matrix up to relative error, using the Gaussian trace estimator of Lemma 1 on its real part. Therefore, both approaches generalize to Hermitian positive definite matrices using one as an upper bound instead of u for the dominant eigenvalue. Algorithms 3 and 4 are modified versions of Algorithms 1 and 2 respectively that work on Hermitian inputs (the function $\mathrm{Re}(\cdot)$ returns the real part of its argument in an entry-wise manner).

Algorithm 3 A Taylor Series Approach to Estimate the Entropy

1: **INPUT**: $\mathbf{R} \in \mathbb{C}^{n\times n}$, accuracy parameter $\varepsilon > 0$, failure probability δ, and integer $m > 0$.
2: Set $s = \left\lceil 20\ln(2/\delta)/\varepsilon^2 \right\rceil$.
3: Let $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_s \in \mathbb{R}^n$ be i.i.d. random Gaussian vectors.
4: **OUTPUT**: return

$$\widehat{\mathcal{H}}(\mathbf{R}) = \frac{1}{s}\sum_{i=1}^{s}\sum_{k=1}^{m}\frac{\mathbf{g}_i^\top\left(\mathrm{Re}\left[\mathbf{R}(\mathbf{I}_n - \mathbf{R})^k\right]\right)\mathbf{g}_i}{k}.$$

Algorithm 4 A Chebyschev Polynomial-Based Approach to Estimate the Entropy

1: **INPUT**: $\mathbf{R} \in \mathbb{C}^{n\times n}$, accuracy parameter $\varepsilon > 0$, failure probability δ, and integer $m > 0$.
2: Set $s = \left\lceil 20\ln(2/\delta)/\varepsilon^2 \right\rceil$.
3: Let $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_s \in \mathbb{R}^n$ be i.i.d. random Gaussian vectors.
4: **OUTPUT:** $\widehat{\mathcal{H}}(\mathbf{R}) = -\frac{1}{s}\sum_{i=1}^{s}\mathbf{g}_i^\top\left(\mathrm{Re}\left[f_m(\mathbf{R})\right]\right)\mathbf{g}_i$.

Theorems 8 and 9 are our main quality-of-approximation results for Algorithm 3 and 4.

Theorem 8: Let $\mathbf{R}$ be a complex density matrix such that all probabilities p_i, $i = 1\ldots n$ satisfy $0 < \ell \le p_i$. Let $\widehat{\mathcal{H}}(\mathbf{R})$ be the output of Algorithm 3 on inputs $\mathbf{R}$, m, and $\epsilon < 1$. Then, with probability at least $1-\delta$,

$$\left|\widehat{\mathcal{H}}(\mathbf{R}) - \mathcal{H}(\mathbf{R})\right| \le 2\epsilon\mathcal{H}(\mathbf{R}),$$

by setting $m = \left\lceil \frac{1}{\ell}\ln\frac{1}{\epsilon}\right\rceil$. The algorithm runs in time

$$\mathcal{O}\left(\frac{\ln(1/\epsilon)}{\ell\cdot\epsilon^2}\cdot\ln(1/\delta)\cdot nnz(\mathbf{R})\right).$$

Theorem 9: Let $\mathbf{R}$ be a density matrix such that all probabilities p_i, $i = 1\ldots n$ satisfy $0 < \ell \le p_i$. Let $\widehat{\mathcal{H}}(\mathbf{R})$ be the output of Algorithm 4 on inputs $\mathbf{R}$, m, and $\epsilon < 1$. Then, with probability at least $1-\delta$,

$$\left|\widehat{\mathcal{H}}(\mathbf{R}) - \mathcal{H}(\mathbf{R})\right| \le 3\epsilon\mathcal{H}(\mathbf{R}),$$

by setting $m = \sqrt{\frac{1}{2\epsilon\ell \ln(1/(1-\ell))}}$. The algorithm runs in time

$$\mathcal{O}\left(\sqrt{\frac{1}{\ell \ln(1/(1-\ell))}} \cdot \frac{1}{\epsilon^{2.5}} \ln(1/\delta) \cdot \text{nnz}(\mathbf{R})\right).$$

V. An Approach via Random Projection Matrices

Finally, we focus on perhaps the most interesting special case: the setting where at most k (out of n, with $k \ll n$) of the probabilities p_i of the density matrix $\mathbf{R}$ of eqn. (2) are non-zero. In this setting, we prove that elegant random-projection-based techniques achieve relative error approximations to all probabilities p_i, $i = 1 \dots k$. The running time of the proposed approach depends on the particular random projection that is used and can be made to depend on the sparsity of the input matrix.

A. Algorithm and Main Theorem

The proposed algorithm uses a random projection matrix $\mathbf{\Pi}$ to create a "sketch" of $\mathbf{R}$ in order to approximate the p_is.

Algorithm 5 Approximating the Entropy via Random Projection Matrices

1: **INPUT**: Integer n (dimensions of matrix $\mathbf{R}$) and integer k (with rank of $\mathbf{R}$ at most $k \ll n$, see eqn. (2)).
2: Construct the random projection matrix $\mathbf{\Pi} \in \mathbb{R}^{n \times s}$ (see Section V-B for details on $\mathbf{\Pi}$ and s).
3: Compute $\tilde{\mathbf{R}} = \mathbf{R}\mathbf{\Pi} \in \mathbb{R}^{n \times s}$.
4: Compute and return the (at most) k non-zero singular values of $\tilde{\mathbf{R}}$, denoted by $\tilde{p}_i$, $i = 1 \dots k$.
5: **OUTPUT**: $\tilde{p}_i$, $i = 1 \dots k$ and $\widehat{\mathcal{H}}(\mathbf{R}) = \sum_{i=1}^{k} \tilde{p}_i \ln \frac{1}{\tilde{p}_i}$.

In words, Algorithm 5 creates a sketch of the input matrix $\mathbf{R}$ by post-multiplying $\mathbf{R}$ by a random projection matrix; this is a well-known approach from the RandNLA literature (see [6] for details). Assuming that $\mathbf{R}$ has rank at most k, which is equivalent to assuming that at most k of the probabilities p_i in eqn. (2) are non-zero (e.g., the system underlying the density matrix $\mathbf{R}$ has at most k pure states), then the rank of $\mathbf{R}\mathbf{\Pi}$ is also at most k. In this setting, Algorithm 5 returns the non-zero singular values of $\mathbf{R}\mathbf{\Pi}$ as approximations to the p_i, $i = 1 \dots k$.

The following theorem is our main quality-of-approximation result for Algorithm 5.

Theorem 10: Let $\mathbf{R}$ be a density matrix with at most $k \ll n$ non-zero probabilities and let $\epsilon < 1/2$ be an accuracy parameter. Then, with probability at least 0.9, the output of Algorithm 5 satisfies

$$\left|p_i^2 - \tilde{p}_i^2\right| \leq \epsilon p_i^2$$

for all $i = 1 \dots k$. Additionally,

$$\left|\mathcal{H}(\mathbf{R}) - \widehat{\mathcal{H}}(\mathbf{R})\right| \leq \sqrt{\epsilon}\mathcal{H}(\mathbf{R}) + \sqrt{\frac{3}{2}}\epsilon.$$

Algorithm 5 (combined with Algorithm 7 below) runs in time

$$\mathcal{O}\left(\text{nnz}(\mathbf{R}) + nk^4/\epsilon^4\right).$$

Comparing the above result with Theorems 2 and 4, we note that the above theorem does not necessitate imposing any constraints on the probabilities p_i, $i = 1 \dots k$. Instead, it suffices to have k non-zero probabilities. The final result is an additive-relative error approximation to the entropy of $\mathbf{R}$ (as opposed to the relative error approximations of Theorems 2 and 4); under the mild assumption $\mathcal{H}(\mathbf{R}) \geq \sqrt{\epsilon}$, the above bound becomes a true relative error approximation.[5]

B. Two Constructions for the Random Projection Matrix

We now discuss two constructions for the matrix $\mathbf{\Pi}$ and we cite two bounds regarding these constructions from prior work that will be useful in our analysis. The first construction is the subsampled Hadamard Transform, a simplification of the Fast Johnson-Lindenstrauss Transform of [14]; see [15], [16] for details. We do note that even though it appears that Algorithm 7 is always better than Algorithm 6 (at least in terms of their respective *theoretical* running times), both algorithms are worth evaluating experimentally: in particular, prior work [17] has reported that Algorithm 6 often outperforms Algorithm 7 in terms of empirical accuracy and running time when the input matrix is dense, as is often the case in our setting. Therefore, we choose to present results (theoretical and empirical) for both well-known constructions of $\mathbf{\Pi}$ (Algorithms 6 and 7).

Algorithm 6 The Subsampled Randomized Hadamard Transform

1: **INPUT:** integers $n, s > 0$ with $s \ll n$.
2: Let $\mathbf{S}$ be an empty matrix.
3: **For** $t = 1, \dots, s$ (i.i.d. trials with replacement) **select uniformly at random** an integer from $\{1, 2, \dots, n\}$.
4: **If** i is selected, **then** append the column vector $\mathbf{e}_i$ to $\mathbf{S}$, where $\mathbf{e}_i \in \mathbb{R}^n$ is the i-th canonical vector.
5: Let $\mathbf{H} \in \mathbb{R}^{n \times n}$ be the normalized Hadamard transform matrix.
6: Let $\mathbf{D} \in \mathbb{R}^{n \times n}$ be a diagonal matrix with

$$\mathbf{D}_{ii} = \begin{cases} +1 & \text{, with probability } 1/2 \\ -1 & \text{, with probability } 1/2 \end{cases}$$

7: **OUTPUT:** $\mathbf{\Pi} = \mathbf{DHS} \in \mathbb{R}^{n \times s}$.

The following result has appeared in [7], [15], [16].

Lemma 11: Let $\mathbf{U} \in \mathbb{R}^{n \times k}$ such that $\mathbf{U}^T\mathbf{U} = \mathbf{I}_k$ and let $\mathbf{\Pi} \in \mathbb{R}^{n \times s}$ be constructed by Algorithm 6. Then, with probability at least 0.9,

$$\left\|\frac{n}{k}\mathbf{U}^T\mathbf{\Pi}\mathbf{\Pi}^T\mathbf{U} - \mathbf{I}_k\right\|_2 \leq \epsilon,$$

by setting $s = \mathcal{O}\left((k + \log n) \cdot \frac{\log k}{\epsilon^2}\right)$.

Our second construction is the input sparsity transform of [18]. This major breakthrough was further analyzed in [19], [20] and we present the following result from [19, Appendix A1].

Lemma 12: Let $\mathbf{U} \in \mathbb{R}^{n \times k}$ such that $\mathbf{U}^T\mathbf{U} = \mathbf{I}_k$ and let $\mathbf{\Pi} \in \mathbb{R}^{n \times s}$ be constructed by Algorithm 7. Then, with

[5] Recall that $\mathcal{H}(\mathbf{R})$ ranges between zero and $\ln k$.

Algorithm 7 An Input-Sparsity Transform

1: **INPUT:** integers $n, s > 0$ with $s \ll n$.
2: Let $\mathbf{S}$ be an empty matrix.
3: **For** $t = 1, \ldots, n$ (i.i.d. trials with replacement) **select uniformly at random** an integer from $\{1, 2, \ldots, s\}$.
4: **If** i is selected, **then** append the row vector $\mathbf{e}_i^T$ to $\mathbf{S}$, where $\mathbf{e}_i \in \mathbb{R}^s$ is the i-th canonical vector.
5: Let $\mathbf{D} \in \mathbb{R}^{n \times n}$ be a diagonal matrix with

$$\mathbf{D}_{ii} = \begin{cases} +1 & \text{, with probability } 1/2 \\ -1 & \text{, with probability } 1/2 \end{cases}$$

6: **OUTPUT:** $\mathbf{\Pi} = \mathbf{DS} \in \mathbb{R}^{n \times s}$.

probability at least 0.9,

$$\|\mathbf{U}^T \mathbf{\Pi}\mathbf{\Pi}^T \mathbf{U} - \mathbf{I}_k\|_2 \leq \epsilon,$$

by setting $s = \mathcal{O}(k^2/\epsilon^2)$.
We refer the interested reader to [20] for improved analyses of Algorithm 7 and its variants.

C. Proof of Theorem 10

At the heart of the proof of Theorem 10 lies the following perturbation bound from [8] (Theorem 2.3).

Theorem 13: Let $\mathbf{DAD}$ be a symmetric positive definite matrix such that $\mathbf{D}$ is a diagonal matrix and $\mathbf{A}_{ii} = 1$ for all i. Let $\mathbf{DED}$ be a perturbation matrix such that $\|\mathbf{E}\|_2 < \lambda_{\min}(\mathbf{A})$. Let λ_i be the i-the eigenvalue of $\mathbf{DAD}$ and let λ_i' be the i-th eigenvalue of $\mathbf{D}(\mathbf{A} + \mathbf{E})\mathbf{D}$. Then, for all i,

$$|\lambda_i - \lambda_i'| \leq \frac{\|\mathbf{E}\|_2}{\lambda_{\min}(\mathbf{A})}.$$

We note that $\lambda_{\min}(\mathbf{A})$ in the above theorem is a real, strictly positive number.[6] Now consider the matrix $\mathbf{R}\mathbf{\Pi}\mathbf{\Pi}^T\mathbf{R}^T$; we will use the above theorem to argue that its singular values are good approximations to the singular values of the matrix $\mathbf{R}\mathbf{R}^T$. Recall that $\mathbf{R} = \mathbf{\Psi}\mathbf{\Sigma}_p\mathbf{\Psi}^T$ where $\mathbf{\Psi}$ has orthonormal columns. Note that the eigenvalues of $\mathbf{R}\mathbf{R}^T = \mathbf{\Psi}\mathbf{\Sigma}_p^2\mathbf{\Psi}^T$ are equal to the eigenvalues of the matrix $\mathbf{\Sigma}_p^2$; similarly, the eigenvalues of $\mathbf{\Psi}\mathbf{\Sigma}_p\mathbf{\Psi}^T\mathbf{\Pi}\mathbf{\Pi}^T\mathbf{\Psi}\mathbf{\Sigma}_p\mathbf{\Psi}^T$ are equal to the eigenvalues of $\mathbf{\Sigma}_p\mathbf{\Psi}^T\mathbf{\Pi}\mathbf{\Pi}^T\mathbf{\Psi}\mathbf{\Sigma}_p$. Thus, we can compare the matrices

$$\mathbf{\Sigma}_p\mathbf{I}_k\mathbf{\Sigma}_p \quad \text{and} \quad \mathbf{\Sigma}_p\mathbf{\Psi}^T\mathbf{\Pi}\mathbf{\Pi}^T\mathbf{\Psi}\mathbf{\Sigma}_p.$$

In the parlance of Theorem 13, $\mathbf{E} = \mathbf{\Psi}^T\mathbf{\Pi}\mathbf{\Pi}^T\mathbf{\Psi} - \mathbf{I}_k$. Applying either Lemma 11 (after rescaling the matrix $\mathbf{\Pi}$) or Lemma 12, we immediately get that $\|\mathbf{E}_A\|_2 \leq \epsilon < 1$ with probability at least 0.9. Since $\lambda_{\min}(\mathbf{I}_k) = 1$, the assumption of Theorem 13 is satisfied. We note that the eigenvalues of $\mathbf{\Sigma}_p\mathbf{I}_k\mathbf{\Sigma}_p$ are equal to p_i^2 for $i = 1 \ldots k$ (all positive, which guarantees that the matrix $\mathbf{\Sigma}_p\mathbf{I}_k\mathbf{\Sigma}_p$ is symmetric positive definite, as mandated by Theorem 13) and the eigenvalues of $\mathbf{\Sigma}_p\mathbf{\Psi}^T\mathbf{\Pi}\mathbf{\Pi}^T\mathbf{\Psi}\mathbf{\Sigma}_p$ are equal to $\tilde{p}_i^2$, where $\tilde{p}_i$ are the singular values of $\mathbf{\Sigma}_p\mathbf{\Psi}^T\mathbf{\Pi}$. (Note that these are exactly equal to the outputs returned by Algorithm 5, since the singular values of $\mathbf{\Sigma}_p\mathbf{\Psi}^T\mathbf{\Pi}$ are equal to the singular values of $\mathbf{\Psi}\mathbf{\Sigma}_p\mathbf{\Psi}^T\mathbf{\Pi} = \mathbf{R}\mathbf{\Pi}$). Thus, we can conclude:

$$\left|p_i^2 - \tilde{p}_i^2\right| \leq \epsilon p_i^2. \tag{20}$$

The above result guarantees that all p_is can be approximated up to *relative error* using Algorithm 5. We now investigate the implication of the above bound to approximating the von Neumann entropy of $\mathbf{R}$. Indeed,

$$\begin{aligned}
\sum_{i=1}^k \tilde{p}_i \ln \frac{1}{\tilde{p}_i} &\leq \sum_{i=1}^k (1+\epsilon)^{1/2} p_i \ln \frac{1}{(1-\epsilon)^{1/2} p_i} \\
&\leq (1+\epsilon)^{1/2} \left(\sum_{i=1}^k p_i \ln \frac{1}{p_i} + \sum_{i=1}^k p_i \ln \frac{1}{(1-\epsilon)^{1/2}} \right) \\
&= (1+\epsilon)^{1/2} \mathcal{H}(\mathbf{R}) + \frac{\sqrt{1+\epsilon}}{2} \ln \frac{1}{1-\epsilon} \\
&\leq (1+\epsilon)^{1/2} \mathcal{H}(\mathbf{R}) + \frac{\sqrt{1+\epsilon}}{2} \ln(1+2\epsilon) \\
&\leq (1+\sqrt{\epsilon}) \mathcal{H}(\mathbf{R}) + \sqrt{\frac{3}{2}}\epsilon.
\end{aligned}$$

In the second to last inequality we used $1/(1-\epsilon) \leq 1+2\epsilon$ for any $\epsilon \leq 1/2$ and in the last inequality we used $\ln(1+2\epsilon) \leq 2\epsilon$ for $\epsilon \in (0, 1/2)$. Similarly, we can prove that:

$$\sum_{i=1}^k \tilde{p}_i \ln \frac{1}{\tilde{p}_i} \geq (1-\sqrt{\epsilon}) \mathcal{H}(\mathbf{R}) - \frac{1}{2}\epsilon.$$

Combining, we get

$$\left| \sum_{i=1}^k \tilde{p}_i \ln \frac{1}{\tilde{p}_i} - \mathcal{H}(\mathbf{R}) \right| \leq \sqrt{\epsilon} \mathcal{H}(\mathbf{R}) + \sqrt{\frac{3}{2}}\epsilon.$$

We conclude by discussing the running time of Algorithm 5. Theoretically, the best choice is to combine the matrix $\mathbf{\Pi}$ from Algorithm 7 with Algorithm 5, which results in a running time

$$\mathcal{O}\left(\text{nnz}(\mathbf{R}) + nk^4/\epsilon^4\right).$$

D. The Hermitian Case

The above approach via random projections critically depends on Lemmas 11 and 12, which, to the best of our knowledge, have only been proven for the real case. These results are typically proven using matrix concentration inequalities, which are well-explored for sums of random real matrices but less explored for sums of real complex matrices. We leave it as an open problem to extend the theoretical analysis of our approach to the Hermitian case.

VI. Experiments

In this section we report experimental results in order to demonstrate the practical efficiency of our algorithms. We show that our algorithms are *both* numerically accurate *and* computationally efficient. Our algorithms were implemented in Matlab R2016a on a compute node with two 10-Core Intel Xeon-E5 processors (2.60GHz) and 512 GBs of RAM.

[6] This follows from the fact that $\mathbf{A}$ is a symmetric positive definite matrix and the inequality $0 \leq \|\mathbf{E}\|_2 < \lambda_{\min}(\mathbf{A})$.

We generated random density matrices for most of which we used the QETLAB Matlab toolbox [9] to derive (real-valued) density matrices of size $5,000 \times 5,000$, on which most of our extensive evaluations were run. We also tested our methods on a much larger $30,000 \times 30,000$ density matrix, which was close to the largest matrix that Matlab would allow us to load. We used the function `RandomDensityMatrix` of QETLAB and the Haar measure; we also experimented with the Bures measure to generate random matrices, but we did not observe any qualitative differences worth reporting. Recall that exactly computing the Von-Neumann entropy using eqn. (1) presumes knowledge of the entire spectrum of the matrix; to compute all singular values of a matrix we used the `svd` function of Matlab. The accuracy of our proposed approximation algorithms was evaluated by measuring the relative error; wall-clock times were reported in order to quantify the speedup that our approximation algorithms were able to achieve.

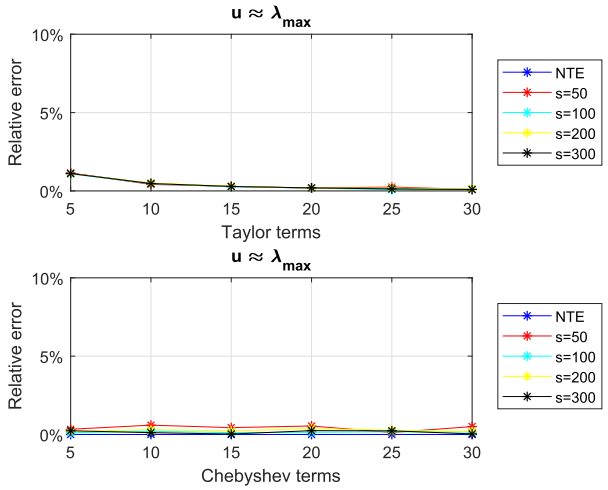

Fig. 1. Relative error for $5,000 \times 5,000$ density matrix using the Taylor and the Chebyshev approximation algorithms with $u = \tilde{\lambda}_{\max}$.

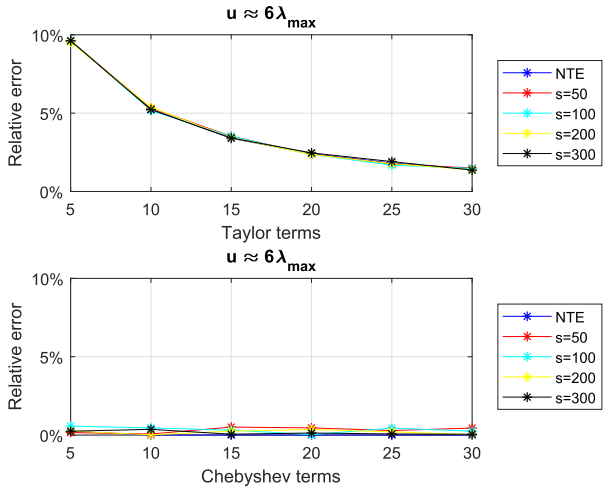

Fig. 2. Relative error for $5,000 \times 5,000$ density matrix using the Taylor and the Chebyshev approximation algorithms with $u = 6\tilde{\lambda}_{\max}$.

A. Empirical Results for the Taylor and Chebyshev Approximation Algorithms

We start by reporting results on the Taylor and Chebyshev approximation algorithms, which have two sources of error: the number of terms that are retained in either the Taylor series expansion or the Chebyshev polynomial approximation *and* the trace estimation that is used in both approximation algorithms. We will separately evaluate the accuracy loss that is contributed by each source of error in order to understand the behavior of the proposed approximation algorithms.

Consider a $5,000 \times 5,000$ random density matrix and let m (the number of terms retained in the Taylor series approximation or the degree of the polynomial used in the Chebyshev polynomial approximation) range between five and 30 in increments of five. Let s, the number of random Gaussian vectors used to estimate the trace, be set to $\{50, 100, 200, 300\}$. Recall that our error bounds for Algorithms 1 and 2 depend on u, an estimate for the largest eigenvalue of the density matrix. We used the power method to estimate the largest eigenvalue (let $\tilde{\lambda}_{\max}$ be the estimate) and we set u to $\tilde{\lambda}_{\max}$ and $6\tilde{\lambda}_{\max}$. Figures 1 and 2 show the relative error (out of 100%) for all combinations of m, s, and u for the Taylor and Chebyshev approximation algorithms. It is worth noting that we also report the error when no trace estimation (NTE) is used in order to highlight that most of the accuracy loss is due to the Taylor/Chebyshev approximation and not the trace estimation.

We observe that the relative error is always small, typically close to 1-2%, for any choice of the parameters s, m, and u. The Chebyshev algorithm returns better approximations when u is an overestimate for $\lambda_{\max}$ while the two algorithms are comparable (in terms of accuracy) where u is very close to $\lambda_{\max}$, which agrees with our theoretical results. We also note that estimating the largest eigenvalue incurs minimal computational cost (less than one second). The NTE line (no trace estimation) in the plots serves as a lower bound for the relative error. Finally, we note that computing the exact Von-Neumann entropy took approximately 1.5 minutes for matrices of this size.

The second dataset that we experimented with was a much larger density matrix of size $30,000 \times 30,000$. This matrix was the largest matrix for which the memory was sufficient to perform operations like the full SVD. Notice that since the increase in the matrix size is six-fold compared to the previous one and SVD's running time grows cubically with the input size, we expect the running time to compute the exact SVD to be roughly $6^3 \cdot 90$ seconds, which is approximately 5.4 hours; indeed, the exact computation of the Von-Neumann entropy took approximately 5.6 hours. We evaluated both the Taylor and the Chebyshev approximation schemes by setting the parameters m and s to take values in the sets $\{5, 10, 15, 20\}$ and $\{50, 100, 200\}$, respectively. The parameter u was set to $\tilde{\lambda}_{\max}$, where the latter value was computed using the power method, which took approximately 3.6 minutes. We report the wall-clock running times and relative error (out of 100%) in Figures 5 and 4.

We observe that the relative error is always less than 1% for both methods, with the Chebyshev approximation yielding

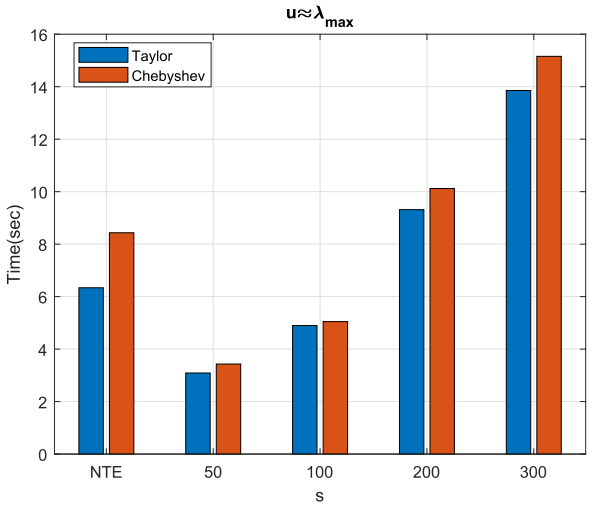

Fig. 3. Time (in seconds) to run the approximate algorithms for the $5,000 \times 5,000$ density matrix for $m = 5$. *Exactly* computing the Von-Neumann entropy took approximately 90 seconds.

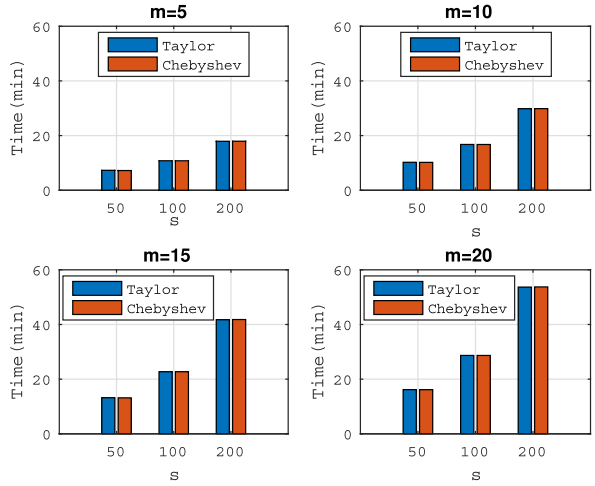

Fig. 5. Wall-clock times: Taylor approximation (blue) and Chebyshev approximation (red) for $u = \tilde{\lambda}_{\max}$. Exact computation needed approximately 5.6 hours.

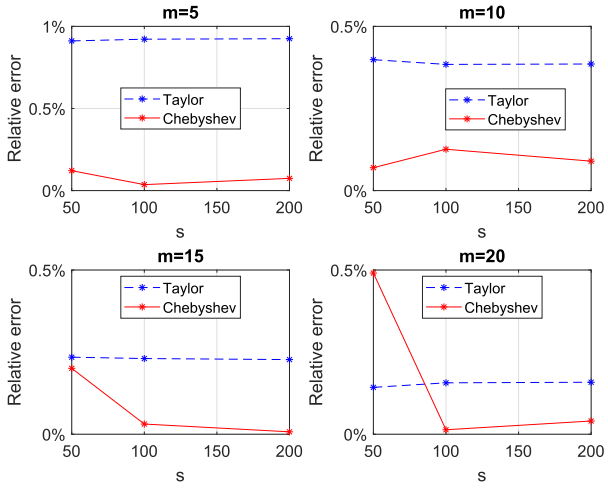

Fig. 4. Relative error for $30,000 \times 30,000$ density matrix using the Taylor and the Chebyshev approximation algorithms with $u = \tilde{\lambda}_{\max}$.

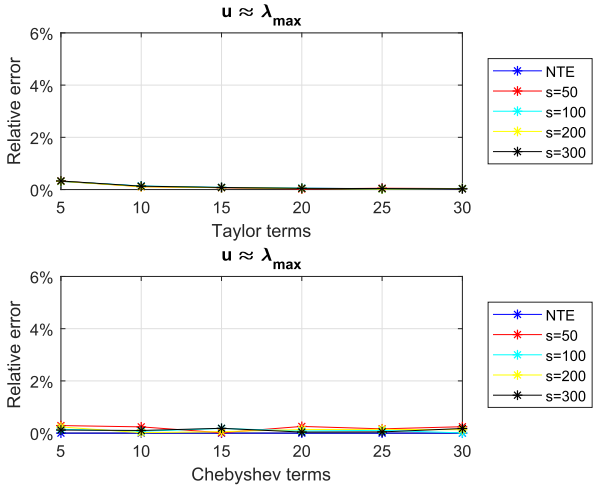

Fig. 6. Relative error for $5,000 \times 5,000$ tridiagonal density matrix using the Taylor and the Chebyshev approximation algorithms with $u = \lambda_{\max}$.

almost always slightly better results. Note that our Chebyshev-polynomial-based approximation algorithm significantly outperformed the exact computation: e.g., for $m = 5$ and $s = 50$, our estimate was computed in less than ten minutes and achieved less than .2% relative error.

The third dataset we experimented with was the tridiagonal matrix from [12, Section 5.1]:

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 & \dots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \dots & 0 & -1 & 2 \end{bmatrix} \tag{21}$$

This matrix is the coefficient matrix of the discretized one-dimensional Poisson equation:

$$f(x) = -\frac{d^2 v_x}{dx}$$

defined in the interval $[0, 1]$ with Dirichlet boundary conditions $v(0) = v(1) = 0$. We normalize $\mathbf{A}$ by dividing it with its trace in order to make it a density matrix. Consider the $5,000 \times 5,000$ normalized matrix $\mathbf{A}$ and let m (the number of terms retained in the Taylor series approximation or the degree of the polynomial used in the Chebyshev polynomial approximation) range between five and 30 in increments of five. Let s, the number of random Gaussian vectors used for estimating the trace be set to 50, 100, 200, or 300. We used the formula

$$\lambda_i = \frac{4}{2n} \sin^2\left(\frac{i\pi}{2n+2}\right), \quad i = 1, \dots, n \tag{22}$$

to compute the eigenvalues of $\mathbf{A}$ (after normalization) and we set u to $\lambda_{\max}$ and $6\lambda_{\max}$. Figures 6 and 7 show the relative error (out of 100%) for all combinations of m, s, and u for the Taylor and Chebyshev approximation algorithms. We also report the error when no trace estimation (NTE) is used.

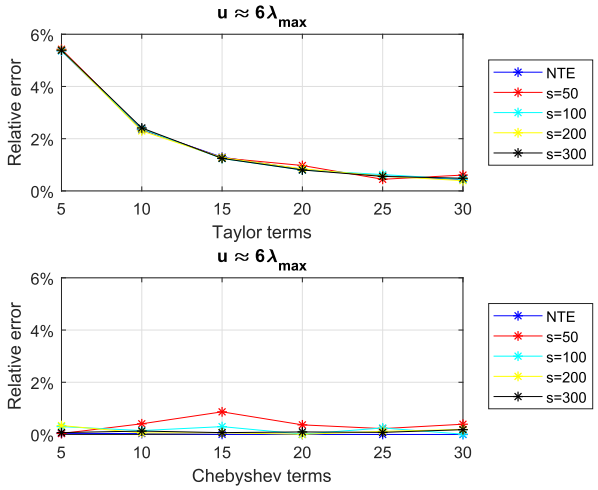

Fig. 7. Relative error for $5,000 \times 5,000$ tridiagonal density matrix using the Taylor and the Chebyshev approximation algorithms with $u = 6\lambda_{\max}$.

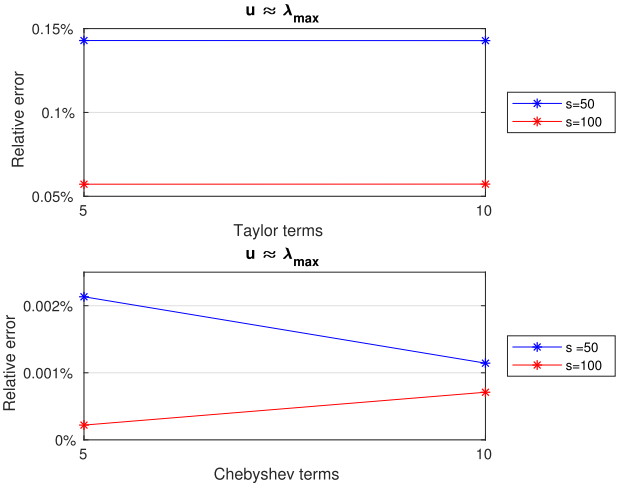

Fig. 9. Relative error for the $10^8 \times 10^8$ tridiagonal density matrix using the Taylor and the Chebyshev approximation algorithms with $u = \lambda_{\max}$.

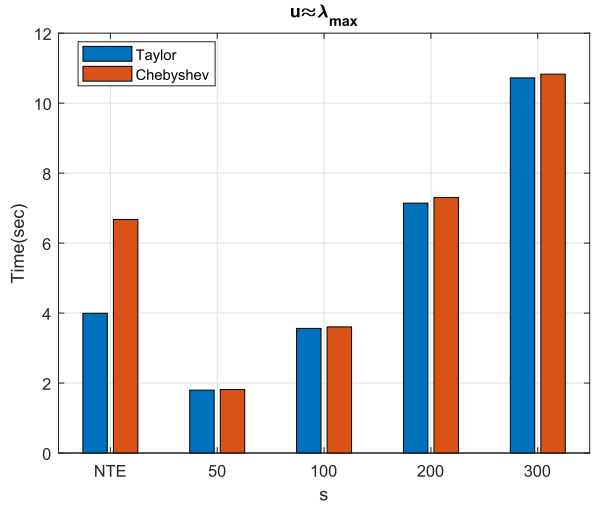

Fig. 8. Wall-clock times: Taylor approximation (blue) and Chebyshev approximation (red) for $m = 5$. Exact computation needed approximately 30 seconds.

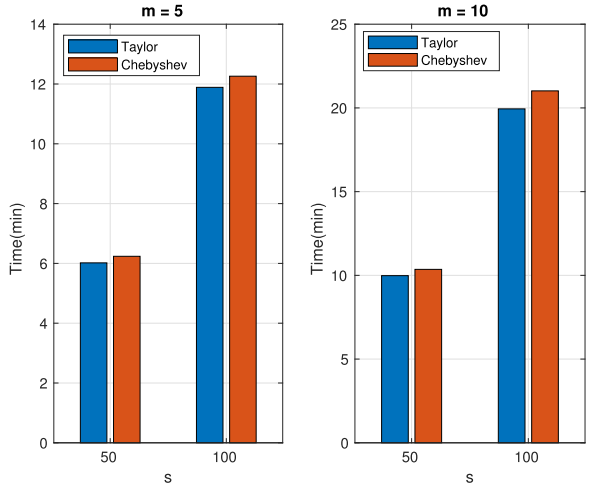

Fig. 10. Wall-clock times: Taylor approximation (blue) and Chebyshev approximation (red) for the $10^8 \times 10^8$ triadiagonal density matrix. Exact computation using the Singular Value Decomposition was infeasible using our computational resources.

We observe that the relative error is higher than the one observed for the $5,000 \times 5,000$ random density matrix. We report wall-clock running times in Figure 8. The Chebyshev-polynomial-based algorithm returns better approximations for all choices of the parameters and, in most cases, is faster than the Taylor-polynomial-based algorithm, e.g. for $m = 5$, $s = 50$ and $u = \lambda_{\max}$, our estimate was computed in about two seconds and achieved less than $.5\%$ relative error.

We further considered a $10^8 \times 10^8$ tridiagonal matrix of the form of eqn. (21). Although an exact computation of the singular values of $\mathbf{A}$ is not feasible (at least with our computational resources), such a computation is not necessary since eqn. (22) provides a closed formula for its eigenvalues and, thus, its entropy. Let m (the number of terms retained in the Taylor series approximation or the degree of the polynomial used in the Chebyshev polynomial approximation) be equal to five or ten and let s, the number of random Gaussian vectors used to estimate the trace be equal to 50 or 100. Figures 9 and 10 show the relative error (out of 100%) and the runtime, respectively, for all combinations of m and s for both the Taylor and Chebyshev approximation algorithms. We observe that in both cases we estimated the entropy in less than ten minutes with a relative error below 0.15%.

The fourth dataset we experimented with includes $5,000 \times 5,000$ density matrices whose first top-k eigenvalues follow a linear decay and the remaining $5,000 - k$ a uniform distribution. Let k, the number of eigenvalues that follow the linear decay, take values in the set $\{50,\ 1000,\ 3500,\ 5000\}$. Let m, the number of terms retained in the Taylor series approximation or the degree of the polynomial used in the Chebyshev polynomial approximation, range between five and 30 in increments of five. Let s, the number of random Gaussian vectors used to estimate the trace, be set to $\{50, 100, 200, 300\}$. The estimate of the largest eigenvalue u is set to $\tilde{\lambda}_{\max}$. Figures 11 to 14 show the relative error (out of 100%) for all combinations of k, m, s, and u for the Taylor and Chebyshev approximation algorithms.

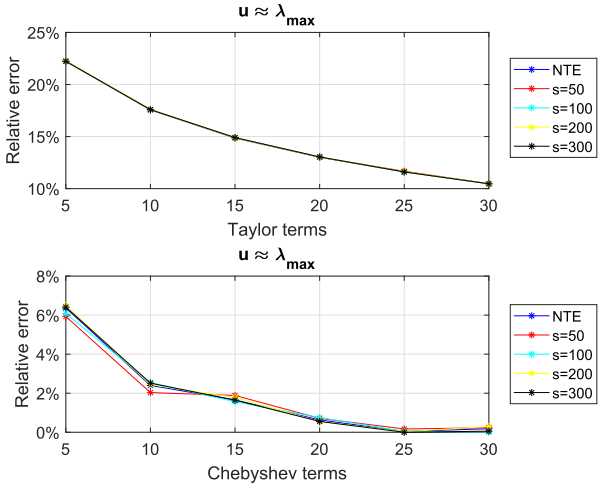

Fig. 11. Relative error for $5,000 \times 5,000$ density matrix with the top-50 eigenvalues decaying linearly using the Taylor and the Chebyshev approximation algorithms with $u = \lambda_{\max}$.

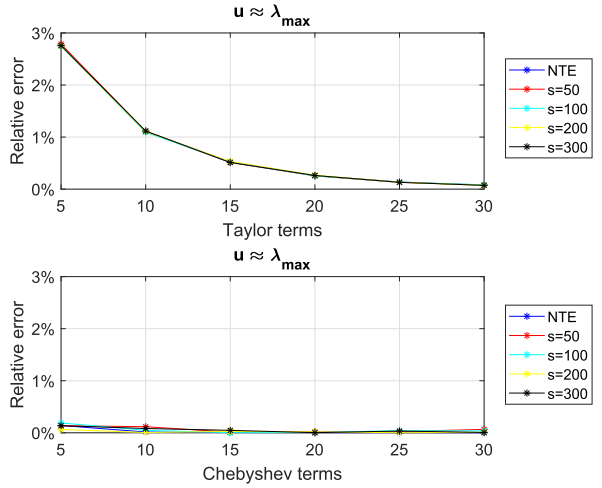

Fig. 12. Relative error for $5,000 \times 5,000$ density matrix with the top-1000 eigenvalues decaying linearly using the Taylor and the Chebyshev approximation algorithms with $u = \lambda_{\max}$.

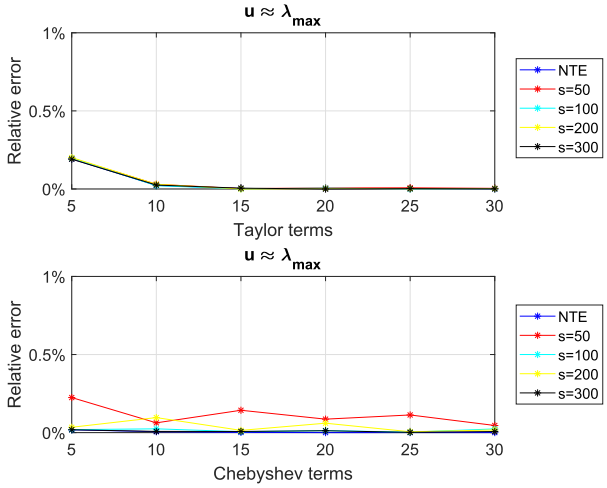

Fig. 13. Relative error for $5,000 \times 5,000$ density matrix with the top-3500 eigenvalues decaying linearly using the Taylor and the Chebyshev approximation algorithms with $u = \lambda_{\max}$.

We observe that the relative error is decreasing as k increases. It is worth noting that when $k = 3,500$ and

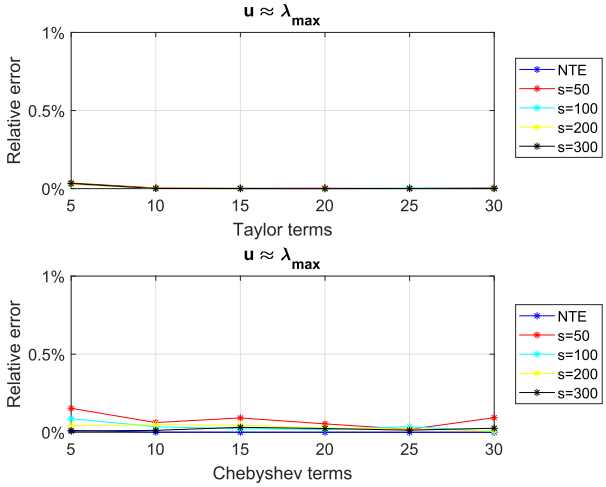

Fig. 14. Relative error for $5,000 \times 5,000$ density matrix with the top-5000 eigenvalues decaying linearly using the Taylor and the Chebyshev approximation algorithms with $u = \lambda_{\max}$.

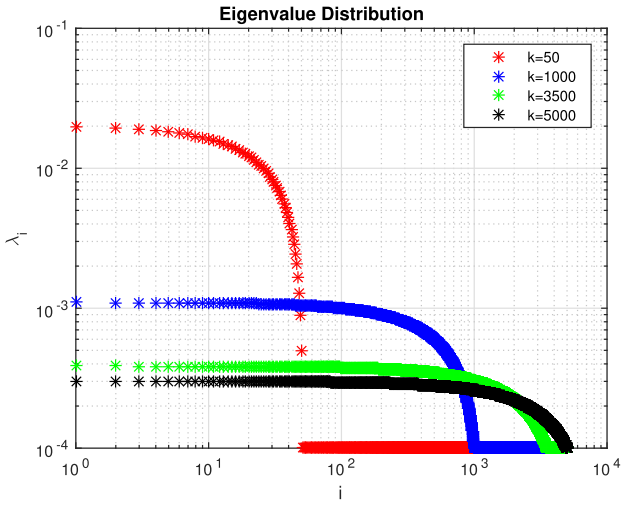

Fig. 15. Eigenvalue distribution of $5,000 \times 5,000$ density matrices with the top-$k = \{50,\ 1000,\ 3500,\ 5000\}$ eigenvalues decaying linearly and the remaining ones $(5,000 - k)$ following a uniform distribution.

$k = 5,000$ the Taylor-polynomial-based algorithm returns better relative error approximation than the Chebyshev-polynomial-based algorithm. In the latter case we observe that the relative error of the Taylor-based algorithm is almost zero. This observation has a simple explanation. Figure 15 shows the distribution of the eigenvalues in the four cases we examine. We observe that for $k = 50$ the eigenvalues are spread in the interval $(10^{-2}, 10^{-4})$; for $k = 1,000$ the eigenvalues are spread in the interval $(10^{-3}, 10^{-4})$; while for $k = 3,500$ or $k = 5,000$ the eigenvalues are of order 10^{-4}. It is well known that the Taylor polynomial returns highly accurate approximations when it is computed on values lying inside the open disc centered at a specific value u, which, in our case, is the approximation to the dominant eigenvalue. The radius of the disk is roughly $r = \lambda_{m+1}/\lambda_m$, where m is the degree of the Taylor polynomial. If $r \leq 1$ then the Taylor polynomial converges; otherwise it diverges. Figure 16 shows the convergence rate for various values of k. We observe that

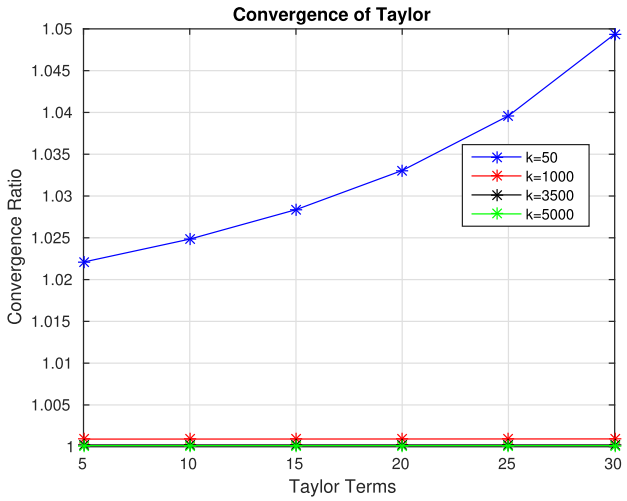

Fig. 16. Convergence radius of the Taylor polynomial for the $5,000 \times 5,000$ density matrices with the top-k $= \{50,\ 1000,\ 3500,\ 5000\}$ eigenvalues decaying linearly and the remaining ones $(5,000 - k)$ following a uniform distribution.

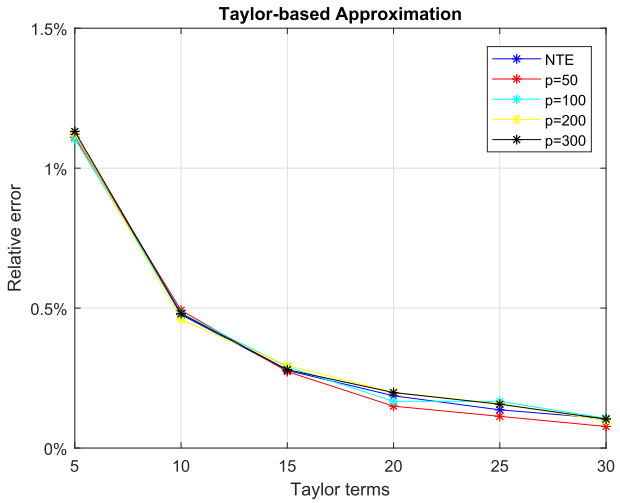

Fig. 17. Relative error for $5,000 \times 5,000$ density matrix using the Taylor approximation algorithm.

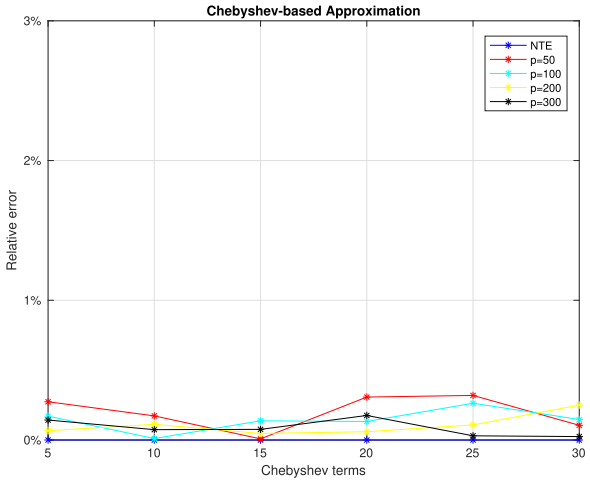

Fig. 18. Relative error for $5,000 \times 5,000$ density matrix using the Chebyshev approximation algorithm.

for $k = 50$ the polynomial diverges, which leads to increased errors for the Taylor-based approximation algorithm (reported error close to 23%). In all other cases, the convergence rate is close to one, resulting in negligible impact to the overall error.

In all four cases, the Chebyshev-polynomial based algorithm behaves better or similar to the Taylor-polynomial based algorithm. It is worth noting that when the majority of the eigenvalues are clustered around the smallest eigenvalue, then to achieve relative error similar to the one observed for the QETLAB random density matrices, more than 30 polynomial terms need to be retained, which increases the computational time of our algorithms. The increase of the computational time as well as the increased relative error can be justified by the large condition number that these matrices have (remember that for both approximation algorithms the running time depends on the approximate condition number u/l). As an example, for $k = 50$, the condition number is in the order of hundreds which is significant larger than the roughly constant condition number when $k = 5,000$.

B. Empirical Results for the Hermitian Case

Our last dataset is a random $5,000 \times 5,000$ complex density matrix generated using the QETLAB Matlab toolbox. We used the function `RandomDensityMatrix` of QETLAB and the Haar measure. Let m (the number of terms retained in the Taylor series approximation or the degree of the polynomial used in the Chebyshev polynomial approximation) range between five and 30 in increments of five. Let s, the number of random Gaussian vectors used to estimate the trace, be set to $\{50, 100, 200, 300\}$. Figures 17 and 18 show the relative error (out of 100%) for all combinations of m, s, and u for the Taylor-based and Chebyshev-based approximation algorithms respectively.

We observe that the relative error is always small, typically below 1%, for any choice of the parameters s and m. The NTE line (no trace estimation) in the plots serves as a lower bound for the relative error. We note that computing the exact Von-Neumann entropy took approximately 52 seconds for matrices of this size. Finally, our algorithm seems to outperform exact computation of the von-Neumann entropy by approximating it in about ten seconds (for the Taylor-based approach) with a relative error of 0.5% using 100 random Gaussian vectors and retaining ten Taylor terms (see Fig. 19) or in about 18 seconds (for the Chebyshev-based approach) with a relative error of 0.2% using 50 random Gaussian vectors and five Chebyshev polynomials (see Fig.20).

C. Empirical Results for the Random Projection Approximation Algorithms

In order to evaluate our third algorithm, we generated low-rank random density matrices (recall that the algorithm of Section V works only for random density matrices of rank k with $k \ll n$). Additionally, in order to evaluate the subsampled randomized Hadamard transform and avoid padding with all-zero rows, we focused on values of n (the number of rows and columns of the density matrix) that are

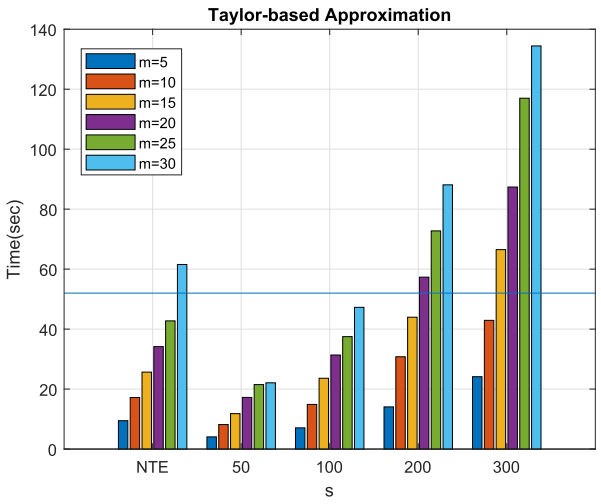

Fig. 19. Time (in seconds) to run the Taylor-based algorithm for the $5,000 \times 5,000$ density matrix for all combinations of m and s. *Exactly* computing the Von-Neumann entropy took approximately 52 seconds, designated by the straight horizontal line in the figure.

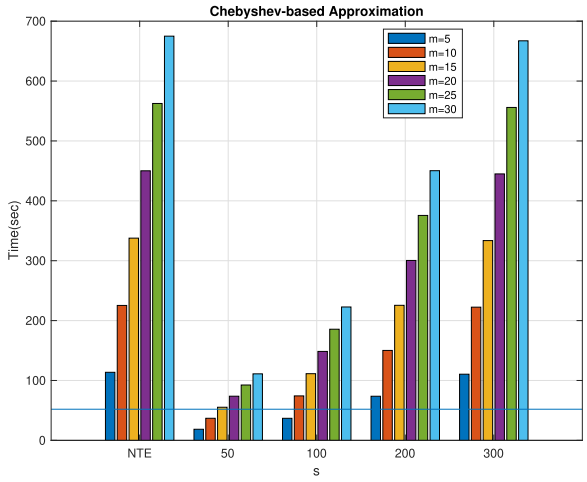

Fig. 20. Time (in seconds) to run the Chebyshev-based algorithm for the $5,000 \times 5,000$ density matrix for all combinations of m and s. *Exactly* computing the Von-Neumann entropy took approximately 52 seconds, designated by the straight horizontal line in the figure.

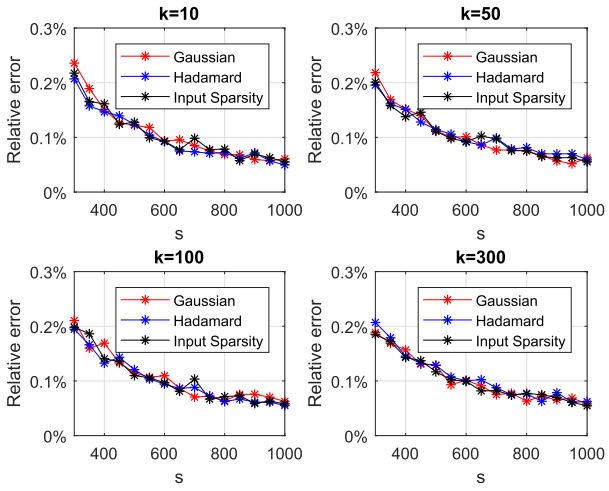

Fig. 21. Relative error for the $4,096 \times 4,096$ rank-k density matrix with exponentially decaying eigenvalues using Algorithm 5 with the Gaussian (red), the subsampled randomized Hadamard transform (blue), and the input sparsity transform (black) random projection matrices.

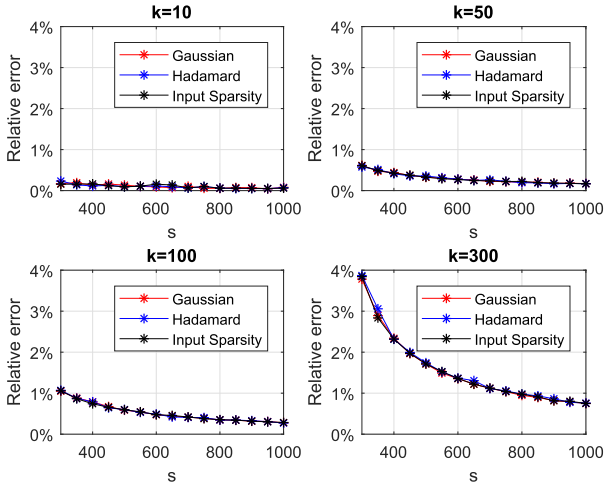

Fig. 22. Relative error for the $4,096 \times 4,096$ rank-k density matrix with linearly decaying eigenvalues using Algorithm 5 with the Gaussian (red), the subsampled randomized Hadamard transform (blue), and the input sparsity transform (black) random projection matrices.

powers of two. Finally, we also evaluated a simpler random projection matrix, namely the Gaussian random matrix, whose entries are all Gaussian random variables with zero mean and unit variance.

We generated *low rank* random density matrices with exponentially (using the QETLAB Matlab toolbox) and linearly decaying eigenvalues. The sizes of the density matrices we tested were $4,096 \times 4,096$ and $16,384 \times 16,384$. We also generated much larger $30,000 \times 30,000$ random matrices on which we only experimented with the Gaussian random projection matrix.

We computed all the non-zero singular values of a matrix using the `svds` function of Matlab in order to take advantage of the fact that the target density matrix has low rank. The accuracy of our proposed approximation algorithms was evaluated by measuring the relative error; wall-clock times were reported in order to quantify the speedup that our approximation algorithms were able to achieve.

We start by reporting results for Algorithm 5 using the Gaussian, the subsampled randomized Hadamard transform (Algorithm 6), and the input-sparsity transform (Algorithm 7) random projection matrices. Consider the $4,096 \times 4,096$ low rank density matrices and let k, the rank of the matrix, be 10, 50, 100, and 300. Let s, the number of columns of the random projection matrix, range from 50 to $1,000$ in increments of 50. Figures 21 and 22 depict the relative error (out of 100%) for all combinations of k and s. We also report the wall-clock running times for values of s between 300 and 450 at Figure 23.

We observe that in the case of the random matrix with exponentially decaying eigenvalues and for all algorithms the relative error is under 0.3% for any choice of the parameters k and s and, as expected, decreases as the dimension of the projection space s grows larger. Interestingly, all three random projection matrices returned essentially identical accuracies

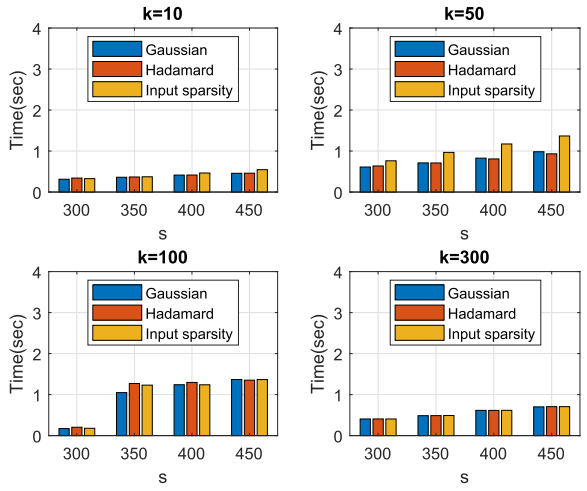

Fig. 23. Wall-clock times: Algorithm 5 on $4,096 \times 4,096$ random matrices, with the Gaussian (blue), the subsampled randomized Hadamard transform (red) and the input sparsity transform (orange) projection matrices. The exact entropy was computed in 1.5 seconds for the rank-10 approximation, in eight seconds for the rank-50 approximation, in 15 seconds for the rank-100 approximation, and in one minute for the rank-300 approximation.

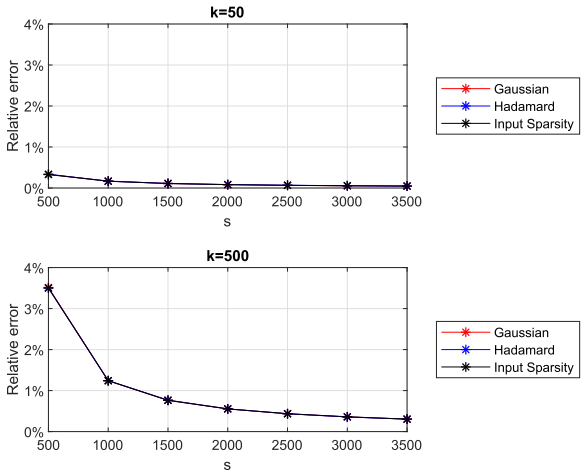

Fig. 24. Relative error for the $16,384 \times 16,384$ rank-k density matrix with exponentially decaying eigenvalues using Algorithm 5 with the Gaussian (red), the subsampled randomized Hadamard transform (blue), and the input sparsity transform (black) random projection matrices.

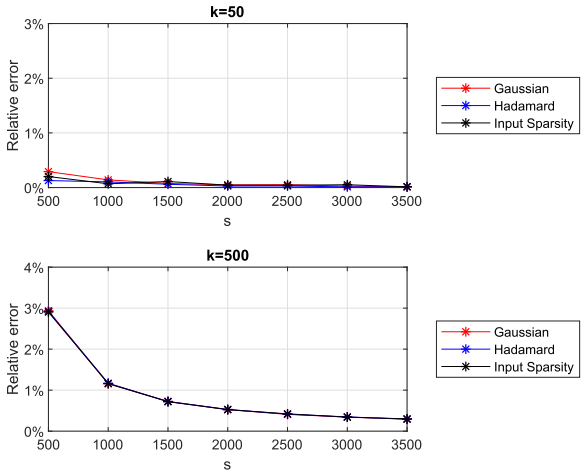

Fig. 25. Relative error for the $16,384 \times 16,384$ rank-k density matrix with linearly decaying eigenvalues using Algorithm 5 with the Gaussian (red), the subsampled randomized Hadamard transform (blue), and the input sparsity transform (black) random projection matrices.

and very comparable wall-clock running time results. This observation is due to the fact that for all choices of k, after scaling the matrix to unit trace, the only eigenvalues that were numerically non-zero were the 10 dominant ones.

In the case of the random matrix with linearly decaying eigenvalues (and for all algorithms) the relative error increases as the rank of the matrix increases and decreases as the size of the random projection matrix increases. This is expected: as the rank of the matrix increases, a larger random projection space is needed to capture the "energy" of the matrix. Indeed, we observe that for all values of k, setting $s = 1,000$ guarantees a relative error under 1%. Similarly, for $k = 10$, the relative error is under 0.3% for any choice of s.

The running time depends not only on the size of the matrix, but also on its rank, e.g. for $k = 100$ and $s = 450$, our approximation was computed in about 2.5 seconds, whereas for $k = 300$ and $s = 450$, it was computed in less than one second. Considering, for example, the case of $k = 300$ exponentially decaying eigenvalues, we observe that for $s = 400$ we achieve relative error below 0.15% and a speedup of over 60 times compared to the exact computation. Finally, it is observed that all three algorithms returned very comparable wall-clock running time results. This observation could be due to the fact that matrix multiplication is heavily optimized in Matlab and therefore the theoretical advantages of the Hadamard transform did not manifest themselves in practice.

The second dataset we experimented with was a $16,384 \times 16,384$ low rank density matrix. We set $k = 50$ and $k = 500$ and we let s take values in the set $\{500,\ 1000,\ 1500, \ldots, 3000,\ 3500\}$. We report the relative error (out of 100%) for all combinations of k and s in Figure 24 for the matrix with exponentially decaying eigenvalues and in Figure 25 for the matrix with linearly decaying eigenvalues. We also report the wall-clock running times for s between 500 and $2,000$ in Figure 26. We observe that the relative error is typically around 1% for both types of matrices, with running times ranging between ten seconds and four minutes, significantly outperforming the exact entropy computation which took approximately 1.6 minutes for the rank 50 approximation and 20 minutes for the rank 500 approximation.

The last dataset we experimented with was a $30,000 \times 30,000$ low rank density matrix on which we ran Algorithm 5 using a Gaussian random projection matrix. We set $k = 50$ and $k = 500$ and we let s take values in the set $\{500,\ 1000,\ 1500, \ldots, 3000,\ 3500\}$. We report the relative error (out of 100%) for all combinations of k and s in Figure 27 for the matrix with exponentially decaying eigenvalues and in Figure 28 for the matrix with the linearly decaying eigenvalues. We also report the wall-clock running times for s ranging between 500 and $2,000$ in Figure 29. We observe that the relative error is typically around 1% for both types of matrices, with the running times ranging between 30 seconds and two minutes, outperforming the exact entropy which was

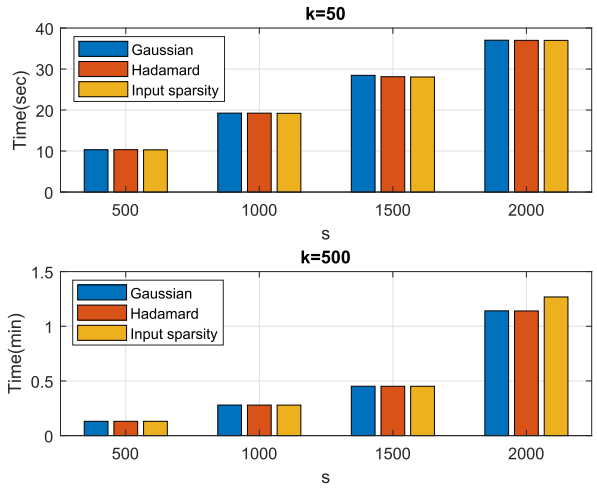

Fig. 26. Wall-clock times: Algorithm 5 with the Gaussian (blue), the sub-sampled randomized Hadamard transform (red) and the input sparsity transform (orange) projection matrices. The exact entropy was computed in 1.6 minutes for the rank 50 approximation and in 20 minutes for the rank 500 approximation.

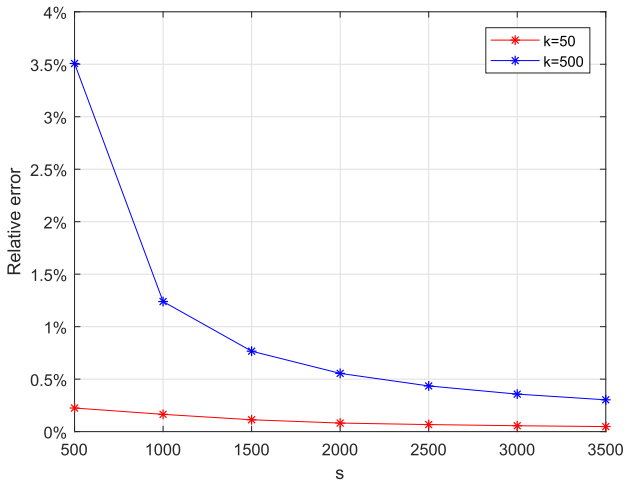

Fig. 27. Relative error for the $30,000 \times 30,000$ rank-k density matrix with exponentially decaying eigenvalues using Algorithm 5 with the Gaussian random projection matrix for $k = 50$ (red) and for $k = 500$ (blue).

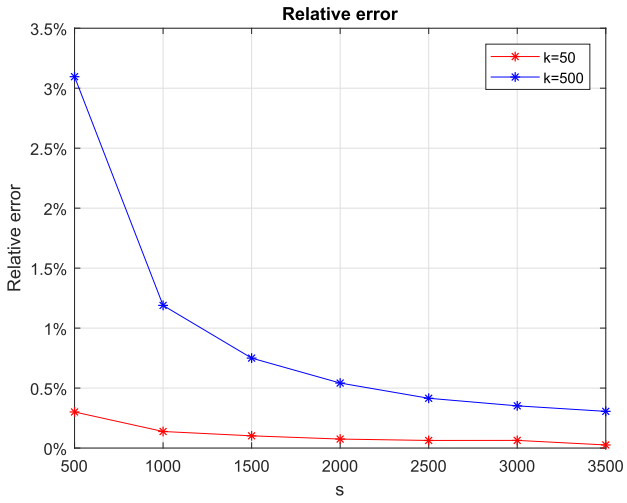

Fig. 28. Relative error for the $30,000 \times 30,000$ rank-k density matrix with linearly decaying eigenvalues using Algorithm 5 with the Gaussian random projection matrix for $k = 50$ (red) and for $k = 500$ (blue).

computed in six minutes for the rank 50 approximation and in one hour for the rank 500 approximation.

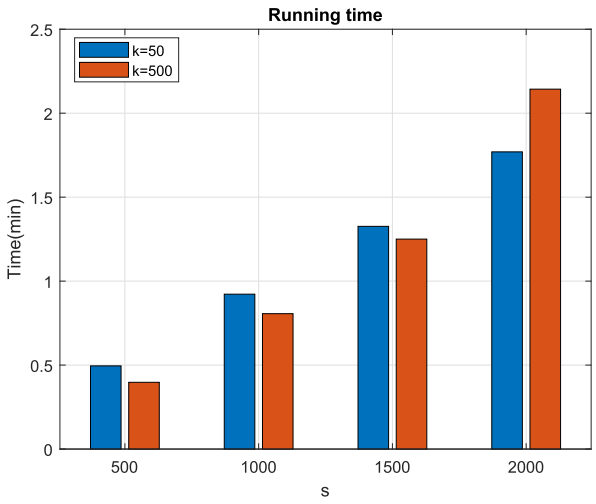

Fig. 29. Wall-clock times: rank-50 approximation (blue) and rank-500 approximation (red). Exact computation needed about six minutes and one hour respectively.

VII. Conclusions and Open Problems

We presented and analyzed three randomized algorithms to approximate the von Neumann entropy of density matrices. Our algorithms leverage recent developments in the RandNLA literature: randomized trace estimators, provable bounds for the power method, the use of random projections to approximate the singular values of a matrix, etc. All three algorithms come with provable accuracy guarantees under assumptions on the spectrum of the density matrix. Empirical evaluations on $30,000 \times 30,000$ synthetic density matrices support our theoretical findings and demonstrate that we can efficiently approximate the von Neumann entropy in a few minutes with minimal loss in accuracy, whereas an the exact computation takes over 5.5 hours.

An interesting open problem would be to consider the estimation of the cross entropy. The cross entropy is a measure between two probability distributions and is particularly important in information theory. Algebraically, it can be defined as $\mathcal{H}(\mathbf{S}, \mathbf{R}) = -\mathbf{tr}(\mathbf{S} \log \mathbf{R})$, where $\mathbf{S} \in \mathbb{C}^{n \times n}$ and $\mathbf{R} \in \mathbb{C}^{n \times n}$ are density matrices with a full set of pure states. One can further extend our polynomial-based approaches using the Taylor expansion or the Chebyshev polynomials to approximate the matrix $\Gamma = \mathbf{S} \log \mathbf{R}$. The case where both or one of the density matrices have an incomplete set of pure states is an open problem: if $\mathbf{R}$ is low-rank, then our first two approaches would not work for the reasons discussed in Section V. However, if the only low rank matrix is $\mathbf{S}$, then our first two approaches would still work: $\mathbf{S}$ is only appearing in the trace estimation part, and having eigenvalues equal to zero does not affect the positive semi-definiteness of Γ. When $\mathbf{R}$ is of low rank then one might be able to use our random projection approaches to reduce its dimensionality and/or the dimensionality of $\mathbf{S}$.

The most important open problem is to relax (or eliminate) the assumptions associated with our three key technical results without sacrificing our running time guarantees. It would be critical to understand whether our assumptions are, for example, necessary to achieve relative error approximations

and either provide algorithmic results that relax or eliminate our assumptions or provide matching lower bounds and counterexamples.

APPENDIX A
THE POWER METHOD

We consider the well-known power method to estimate the largest eigenvalue of a matrix. In our context, we will use the power method to estimate the largest probability p_i for a density matrix $\mathbf{R}$.

Algorithm 8 Power Method, Repeated q Times

- **INPUT:** SPD matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, integers $q,\ t > 0$.
- For $j = 1, \ldots, q$
  1) Pick uniformly at random a vector $\mathbf{x}_0^j \in \{+1, -1\}^n$.
  2) For $i = 1, \ldots, t$
     - $\mathbf{x}_i^j = \mathbf{A} \cdot \mathbf{x}_{i-1}^j$.
  3) Compute: $\tilde{p}_1^j = \frac{\mathbf{x}_t^{j\,T}\mathbf{A}\mathbf{x}_t^j}{\mathbf{x}_t^{j\,T}\mathbf{x}_t^j}$.
- **OUTPUT:** $\tilde{p}_1 = \max_{j=1\ldots q} \tilde{p}_1^j$.

Algorithm 8 requires $\mathcal{O}(qt(n + \mathrm{nnz}(\mathbf{A})))$ arithmetic operations to compute $\tilde{p}_1$. The following lemma appeared in [4], building upon [13].

Lemma 14: Let $\tilde{p}_1$ be the output of Algorithm 8 with $q = \lceil 4.82 \log(1/\delta) \rceil$ and $t = \lceil \log \sqrt{4n} \rceil$. Then, with probability at least $1 - \delta$,

$$\frac{1}{6}\, p_1 \le \tilde{p}_1 \le p_1.$$

The running time of Algorithm 8 is $\mathcal{O}\left((n + \mathrm{nnz}(\mathbf{A})) \log(n) \log\left(\frac{1}{\delta}\right)\right)$.

APPENDIX B
THE CLENSHAW ALGORITHM

We briefly sketch Clenshaw's algorithm to evaluate Chebyshev polynomials with matrix inputs. Clenshaw's algorithm is a recursive approach with base cases $b_{m+2}(x) = b_{m+1}(x) = 0$ and the recursive step (for $k = m, m-1, \ldots, 0$):

$$b_k(x) = \alpha_k + 2x b_{k+1}(x) - b_{k+2}(x). \tag{23}$$

(See Section III for the definition of α_k.) Then,

$$f_m(x) = \frac{1}{2}\left(\alpha_0 + b_0(x) - b_2(x)\right). \tag{24}$$

Using the mapping $x \to 2(x/u) - 1$, eqn. (23) becomes

$$b_k(x) = \alpha_k + 2\left(\frac{2}{u}x - 1\right) b_{k+1}(x) - b_{k+2}(x). \tag{25}$$

In the matrix case, we substitute x by a matrix. Therefore, the base cases are $\mathbf{B}_{m+2}(\mathbf{R}) = \mathbf{B}_{m+1}(\mathbf{R}) = \mathbf{0}$ and the recursive step is

$$\mathbf{B}_k(\mathbf{R}) = \alpha_k \mathbf{I}_n + 2\left(\frac{2}{u}\mathbf{R} - \mathbf{I}_n\right)\mathbf{B}_{k+1}(\mathbf{R}) - \mathbf{B}_{k+2}(\mathbf{R}) \tag{26}$$

for $k = m, m-1, \ldots, 0$. The final sum is

$$f_m(\mathbf{R}) = \frac{1}{2}\left(\alpha_0 \mathbf{I}_n + \mathbf{B}_0(\mathbf{R}) - \mathbf{B}_2(\mathbf{R})\right). \tag{27}$$

Using the matrix version of Clenshaw's algorithm, we can now rewrite the trace estimation $\mathbf{g}^\top f_m(\mathbf{R})\mathbf{g}$ as follows. First, we right multiply eqn. (26) by $\mathbf{g}$,

$$\mathbf{B}_k(\mathbf{R})\mathbf{g} = \alpha_k \mathbf{I}_n \mathbf{g} + 2\left(\frac{2}{u}\mathbf{R} - \mathbf{I}_n\right)\mathbf{B}_{k+1}(\mathbf{R})\mathbf{g} - \mathbf{B}_{k+2}(\mathbf{R})\mathbf{g},$$

$$\mathbf{y}_k = \alpha_k \mathbf{g} + 2\left(\frac{2}{u}\mathbf{R} - \mathbf{I}_n\right)\mathbf{y}_{k+1} - \mathbf{y}_{k+2}. \tag{28}$$

Eqn. (28) follows by substituting $\mathbf{y}_i = \mathbf{B}_i(\mathbf{R})\mathbf{g}$. Multiplying the base cases by $\mathbf{g}$, we get $\mathbf{y}_{m+2} = \mathbf{y}_{m+1} = \mathbf{0}$ and the final sum becomes

$$\mathbf{g}^\top f_m(\mathbf{R})\mathbf{g} = \frac{1}{2}\left(\alpha_0(\mathbf{g}^\top \mathbf{g}) + \mathbf{g}^\top(\mathbf{y}_0 - \mathbf{y}_2)\right). \tag{29}$$

Algorithm 9 summarizes all the above.

Algorithm 9 Clenshaw's Algorithm to Compute $\mathbf{g}^\top f_m(\mathbf{R})\mathbf{g}$

INPUT: α_i, $i = 0, \ldots, m$, $\mathbf{R} \in \mathbb{R}^{n\times n}$, $\mathbf{g} \in \mathbb{R}^n$
Set $\mathbf{y}_{m+2} = \mathbf{y}_{m+1} = \mathbf{0}$
for $k = m, m-1, \ldots, 0$ **do**
  $\mathbf{y}_k = \alpha_k \mathbf{g} + \frac{4}{u}\mathbf{R}\mathbf{y}_{k+1} - 2\mathbf{y}_{k+1} - \mathbf{y}_{k+2}$
end for
OUTPUT: $\mathbf{g}^\top f_m(\mathbf{R})\mathbf{g} = \frac{1}{2}\left(\alpha_0(\mathbf{g}^\top \mathbf{g}) + \mathbf{g}^\top(\mathbf{y}_0 - \mathbf{y}_2)\right)$

ACKNOWLEDGMENT

The authors would like to thank the editor for numerous useful suggestions that significantly improved the presentation of their work, especially in the Hermitian case.

REFERENCES

[1] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD, USA: The Johns Hopkins Univ. Press, 1996.
[2] T. P. Wihler, B. Bessire, and A. Stefanov, "Computing the entropy of a large matrix," *J. Phys. A, Math. Gen.*, vol. 47, no. 24, Jun. 2014, Art. no. 245201.
[3] N. J. Higham, *Functions of Matrices: Theory and Computation*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2008.
[4] C. Boutsidis, P. Drineas, P. Kambadur, E.-M. Kontopoulou, and A. Zouzias, "A randomized algorithm for approximating the log determinant of a symmetric positive definite matrix," *Linear Algebra Appl.*, vol. 533, pp. 95–117, Nov. 2017.
[5] H. Avron and S. Toledo, "Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix," *J. ACM*, vol. 58, no. 2, pp. 1–34, Apr. 2011.
[6] P. Drineas and M. W. Mahoney, "RandNLA: Randomized numerical linear algebra," *Commun. ACM*, vol. 59, no. 6, pp. 80–90, May 2016.
[7] D. P. Woodruff, "Sketching as a tool for numerical linear algebra," *Found. Trend Theor. Comput. Sci.*, vol. 10, no. 2, pp. 1–157, 2014.
[8] J. Demmel and K. Veselić, "Jacobi's method is more accurate than QR," *SIAM J. Matrix Anal. Appl.*, vol. 13, no. 4, pp. 1204–1245, 1992.
[9] N. Johnston. (2016). *QETLAB: A MATLAB Toolbox for Quantum Entanglement, Version 0.9*. [Online]. Available: http://qetlab.com
[10] C. Musco, P. Netrapalli, A. Sidford, S. Ubaru, and D. P. Woodruff, "Spectrum approximation beyond fast matrix multiplication: Algorithms and hardness," Apr. 2017, *arXiv:1704.04163*. [Online]. Available: http://arxiv.org/abs/1704.04163
[11] N. J. Harvey, J. Nelson, and K. Onak, "Sketching and streaming entropy via approximation theory," in *Proc. 49th Annu. IEEE Symp. Found. Comput. Sci.*, Oct. 2008, pp. 489–498.
[12] I. Han, D. Malioutov, and J. Shin, "Large-scale log-determinant computation through stochastic Chebyshev expansions," in *Proc. 32nd Int. Conf. Mach. Learn.*, vol. 37, 2015, pp. 908–917.

[13] L. Trevisan, "Graph partitioning and expanders," Lecture Notes, Stanford Univ., Stanford, CA, USA, 2011.
[14] N. Ailon and B. Chazelle, "The fast Johnson–Lindenstrauss transform and approximate nearest neighbors," *SIAM J. Comput.*, vol. 39, no. 1, pp. 302–322, May 2009.
[15] P. Drineas, M. W. Mahoney, S. Muthukrishnan, and T. Sarlós, "Faster least squares approximation," *Numer. Math.*, vol. 117, no. 2, pp. 219–249, Feb. 2011.
[16] J. A. Tropp, "Improved analysis of the subsampled randomized Hadamard transform," *Adv. Adapt. Data Anal.*, vol. 3, nos. 1–2, pp. 115–126, Apr. 2011.
[17] S. Paul, C. Boutsidis, M. Magdon-Ismail, and P. Drineas, "Random projections for support vector machines," in *Proc. 16th Int. Conf. Artif. Intell. Stat.*, 2013, pp. 498–506.
[18] K. L. Clarkson and D. P. Woodruff, "Low rank approximation and regression in input sparsity time," in *Proc. 45th Annu. ACM Symp. Theory Comput. (STOC)*. New York, NY, USA: ACM, 2013, pp. 81–90.
[19] X. Meng and M. W. Mahoney, "Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression," in *Proc. 45th Annu. ACM Symp. Theory Comput. (STOC)*, 2013, pp. 91–100.
[20] J. Nelson and H. L. Nguyen, "OSNAP: Faster numerical linear algebra algorithms via sparser subspace embeddings," in *Proc. IEEE 54th Annu. Symp. Found. Comput. Sci.*, Oct. 2013, pp. 117–126.

Eugenia-Maria Kontopoulou received the B.Eng. and M.Eng. degrees from the Computer Science and Informatics Department, University of Patras, Greece, in 2012. She is currently pursuing the Ph.D. degree with the Department of Computer Science, Purdue University. Her current research interests lie in the areas of (randomized) numerical linear algebra, with a focus on designing and implementing randomized algorithms for the solution of linear algebraic problems in large-scale data applications.

Gregory-Paul Dexter is currently pursuing the degree (Hons.) in statistics and mathematics with Purdue University. He is broadly interested in artificial intelligence and hopes to pursue a Ph.D. degree focused in this area.

Wojciech Szpankowski (Fellow, IEEE) is currently the Saul Rosen Distinguished Professor of computer science with Purdue University, where he teaches and conducts research in analysis of algorithms, information theory, analytic combinatorics, data science, random structures, and stability problems of distributed systems. He held several visiting professor/scholar positions, including McGill University; INRIA, France; Stanford; Hewlett-Packard Labs; the Universite de Versailles; the University of Canterbury, New Zealand; Ecole Polytechnique, France; the Newton Institute, Cambridge, U.K.; ETH, Zurich; and Gdansk University of Technology, Poland. He is an Erskine Fellow. He has published two books: *Average Case Analysis of Algorithms on Sequences* (John Wiley & Sons, 2001) and *Analytic Pattern Matching: From DNA to Twitter* (Cambridge, 2015). He launched the interdisciplinary Institute for Science of Information in 2008. In 2010, he became the Director of the newly established NSF Science and Technology Center for Science of Information. He received the Humboldt Research Award in 2010, the Inaugural Arden L. Bement Jr. Award in 2015, and the Flajolet Lecture Prize in 2020.

Ananth Grama received the Ph.D. degree in computer science from the University of Minnesota. He is currently the Samuel Conte Professor of computer science with Purdue University. His research interests include parallel and distributed computing, large-scale data analytics, and applications in life sciences. He is a Fellow of the American Association for the Advancement of Sciences and a Distinguished Alumnus of the University of Minnesota. He was a recipient of the National Science Foundation CAREER Award and the Purdue University Faculty Scholar Award. He chaired the Bio-Data Management and Analysis (BDMA) Study Section, National Institutes of Health, from 2012 to 2014.

Petros Drineas received the B.S. degree in computer engineering and informatics from the University of Patras, Greece, in 1997, and the Ph.D. degree in computer science from Yale University in 2003. From 2003 to 2016, he was an Assistant (until 2009) and then an Associate Professor with the Rensselaer Polytechnic Institute. He is currently a Professor with the Department of Computer Science, Purdue University. His research interests lie in the design and analysis of randomized algorithms for linear algebraic problems, and their applications to the analysis of modern, massive datasets, with a particular emphasis on the analysis of population genetics data.