Sublinear Time Spectral Density Estimation

Vladimir Braverman
Johns Hopkins University
Baltimore, MD, USA
vova@cs.jhu.edu

Aditya Krishnan
Johns Hopkins University
Baltimore, MD, USA
akrish23@jhu.edu

Christopher Musco
New York University
New York, NY, USA
cmusco@nyu.edu

ABSTRACT

We present a new sublinear time algorithm for approximating the spectral density (eigenvalue distribution) of an $n \times n$ normalized graph adjacency or Laplacian matrix. The algorithm recovers the spectrum up to ϵ accuracy in the Wasserstein-1 distance in $O(n \cdot \text{poly}(1/\epsilon))$ time given sample access to the graph. This result compliments recent work by David Cohen-Steiner, Weihao Kong, Christian Sohler, and Gregory Valiant (2018), which obtains a solution with runtime independent of n, but exponential in $1/\epsilon$. We conjecture that the trade-off between dimension dependence and accuracy is inherent.

Our method is simple and works well experimentally. It is based on a Chebyshev polynomial moment matching method that employees randomized estimators for the matrix trace. We prove that, for any Hermitian A, this moment matching method returns an ϵ approximation to the spectral density using just $O(1/\epsilon)$ matrix-vector products with A. By leveraging stability properties of the Chebyshev polynomial three-term recurrence, we then prove that the method is amenable to the use of coarse approximate matrix-vector products. Our sublinear time algorithm follows from combining this result with a novel sampling algorithm for approximating matrix-vector products with a normalized graph adjacency matrix.

Of independent interest, we show a similar result for the widely used *kernel polynomial method* (KPM), proving that this practical algorithm nearly matches the theoretical guarantees of our moment matching method. Our analysis uses tools from Jackson's seminal work on approximation with positive polynomial kernels.

CCS CONCEPTS

• **Theory of computation → Streaming, sublinear and near linear time algorithms**; **Graph algorithms analysis**.

KEYWORDS

density estimation, moment matching, approximate matrix-vector multiplication, graph approximation

ACM Reference Format:
Vladimir Braverman, Aditya Krishnan, and Christopher Musco. 2022. Sublinear Time Spectral Density Estimation. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC '22), June 20–24, 2022, Rome, Italy.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3519935.3520009

STOC '22, June 20–24, 2022, Rome, Italy

ACM ISBN 978-1-4503-9264-8/22/06…$15.00
https://doi.org/10.1145/3519935.3520009

1 INTRODUCTION

A ubiquitous task in computational science, engineering, and data science is to extract information about the eigenvalue spectrum of a matrix $A \in \mathbb{R}^{n \times n}$. A full eigendecomposition takes at least $O(n^\omega)$ time[1], which is prohibitively expensive for large matrices [3, 29]. So, we are typically interested in extracting *partial information* about the spectrum. This can be done using iterative methods like the power or Lanczos methods, which access A via a small number of matrix-vector multiplications. Each multiplication takes at most $O(n^2)$ time to compute, and can be accelerated when A is sparse or structured, leading to fast algorithms.

However, the partial spectral information computed by most iterative methods is limited. Algorithms typically only obtain accurate approximations to the outlying, or *largest magnitude* eigenvalues of A, missing information about the *interior* of A's spectrum that may be critical in applications. For example, in network science, clusters of interior eigenvalues can indicate graph structures like repeated motifs [11]. In deep learning, information on how the spectrum of a weight matrix differs from its random initialization can give hints about model convergence and generalization [23, 31], and Hessian eigenvalues are useful in optimization [13]. Coarse information about interior eigenvalues is also used to initialize parallel GPU based methods for full eigendecomposition [1, 21].

To address these needs and many other applications, there has been substantial interest in methods for estimating the full *spectral density* of a matrix A [41]. Concretely, assume that A is Hermitian with real eigenvalues $\lambda_1, \ldots, \lambda_n$. We view its spectrum as a probability density s:

$$\text{Spectral density:} \qquad s(x) = \frac{1}{n} \sum_{i=1}^{n} \delta(x - \lambda_i). \tag{1}$$

Here δ is the Dirac delta function. The goal is to find a probability density q that approximates s in some natural metric, like the Wasserstein distance. The density q can either be continuous (represented in some closed form) or discrete (represented as a list of approximate eigenvalues $\tilde{\lambda}_1, \ldots, \tilde{\lambda}_n$). See Figure 1 for an illustration. Both sorts of approximation are useful in applications.

Methods for spectral density estimation that run in $o(n^\omega)$ time were first introduced for applications in condensed matter physics and quantum chemistry [34, 36, 39]. Many are based on the combination of two important tools: 1) moment matching, and 2) stochastic trace estimation. Specifically, if we had access to moments of the distribution s, i.e. $\frac{1}{n}\sum_{i=1}^n \lambda_i$, $\frac{1}{n}\sum_{i=1}^n \lambda_i^2$, $\frac{1}{n}\sum_{i=1}^n \lambda_i^3$, etc., then we could find a good approximation q by finding a distribution that

[1] Here $\omega < 2.373$ is the fast matrix multiplication exponent.

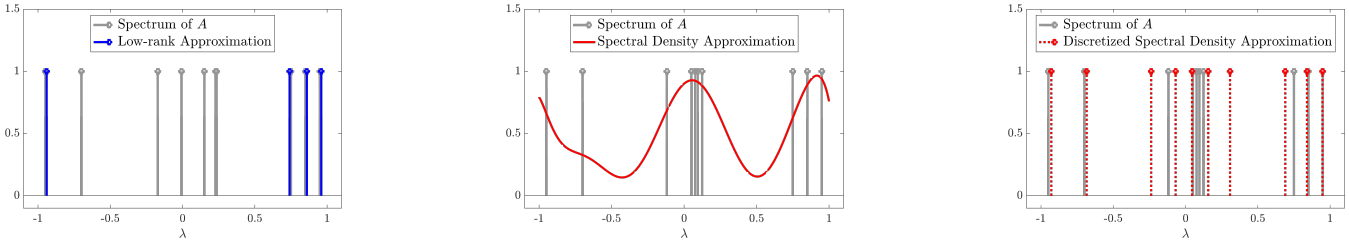

Figure 1: Different approximations for the spectrum of a matrix A with eigenvalues in $[-1, 1]$. A typical approximation computed using an iterative eigenvalue algorithm mostly preserves information about the largest magnitude eigenvalues. In contrast, the spectral density estimates in the two right figures coarsely approximate the entire distribution of A's eigenvalues, the first with a low-degree polynomial, and the second with a discrete distribution.

agrees with s on these moments. Moreover, these *spectral moments* can be computed via the matrix trace: note that $\text{tr}(A) = \sum_{i=1}^n \lambda_i$, $\text{tr}(A^2) = \sum_{i=1}^n \lambda_i^2$, $\text{tr}(A^3) = \sum_{i=1}^n \lambda_i^3$, etc. While we cannot hope to compute $\text{tr}(A^k)$ exactly in $o(n^\omega)$ time, thanks to stochastic trace estimators like Hutchinson's method, this trace can be approximated much more quickly [2, 15]. Such estimators are based on the observation that, for any matrix $B \in \mathbb{R}^{n\times n}$, $\text{tr}(B)$ can be well approximated by $\text{tr}(G^T BG)$ where $G \in \mathbb{R}^{n\times m}$ contains random sub-Gaussian entries and $m \ll n$. For any k degree polynomial g, $G^T g(A)G$ can be computed with just $O(km)$ matrix-vector multiplications, so we can quickly approximate any low-degree moment of A's spectral density.

While this high-level approach and related techniques have been applied successfully to estimating the spectra of a wide variety of matrices [22, 41], theoretical guarantees have only appeared relatively recently. Perhaps surprisingly, it can be shown that many common methods provably run in *linear time* for any Hermitian matrix A. For instance, in work concurrent to ours, Chen, Trogdan, and Ubaru [6] show that for any $n \times n$ Hermitian matrix A with spectral density s, the popular Stochastic Lanczos Quadrature (SLQ) method provably computes an approximate spectral density q satisfying:

$$W_1(s, q) \leq \epsilon \tag{2}$$

using just $\text{poly}(1/\epsilon)$ matrix-vector multiplications with A. Above W_1 denotes the Wasserstein-1 distance, aka the "earth-movers distance".[2] We defer a formal definition of W_1 to Section 2. The measure is convenient because, unlike many other measures of statistical distance, it allows a discrete distribution like the spectral density to be meaningfully compared to a possibly continuous approximation. For discrete approximations, the Wasserstein distance is related to a simple ℓ_1 metric. If we let $\Lambda = [\lambda_1, \ldots, \lambda_n]$ be a vector of A's eigenvalues and $\tilde{\Lambda} = [\tilde{\lambda}_1, \ldots, \tilde{\lambda}_n]$ be a vector of approximate eigenvalues, then $\|\Lambda - \tilde{\Lambda}\|_1 \leq n\epsilon$ if and only if $W_1(s, q) \leq \epsilon$ for the discrete spectral density q with eigenvalues in $\tilde{\Lambda}$.

As a step towards our main sublinear time result, in this work we show that similar bounds to [6] can also be proven for the popular kernel polynomial method (KPM) [41] and for a natural moment matching algorithm based on Chebyshev polynomials.

[2]We assume $\|A\|_2 \leq 1$ for simplicity of stating errror guarantees, noting that Wasserstein distance is not scale invariant. This assumption is without loss of generality since $\|A\|_2$ can always be scaled after computing the top eigenvector up to constant fact accuracy, which takes just $O(\log n)$ matrix-vector multiplications [26].

1.1 Our Contributions

With linear time spectral density estimation algorithms in hand for all Hermitian matrices, a natural question is if we can go faster for specific classes of matrices. In particular, there has been growing interest in SDE algorithms for graph structured matrices like adjacency matrices and Laplacians [11]. A remarkable recent result by Cohen et al. [8] shows that, for normalized graph adajeceny matrices, it is possible to achieve guarantee (2) in $2^{O(1/\epsilon)}$ time, given appropriate query access to the target graph. Importantly, this runtime *does not depend on n*. However, given the exponential dependence on ϵ, the algorithm is impractical even for coarse spectral approximations.

Our main contribution is a method that obtains a *polynomial* dependence on ϵ, at the cost of a linear dependence on the matrix dimension n. Since A can have n^2 non-zero entries, the runtime is still *sublinear* in the problem size, but with a much more acceptable dependence on accuracy.

Theorem 1.1 (Sublinear time spectral density estimation for graphs.). *Let $G = (V, E)$ be an unweighted, undirected n-vertex graph and let $A \in \mathbb{R}^{n\times n}$ be the normalized adjacency of G with spectral density s. Let $\epsilon, \delta \in (0, 1)$ be fixed values. Assume that we can 1) uniformly sample a random vertex in constant time, 2) uniformly sample a random neighbor of any vertex $i \in V$ in constant time, and 3) for a vertex i with degree d_i, read off all neighbors in $O(d_i)$ time.[3] Then there is a randomized algorithm with expected running time $O(n\,\text{poly}(\log(1/\delta)/\epsilon))$ which outputs a density function $q : [-1, 1] \to \mathbb{R}^+$ such that $W_1(q, s) \leq \epsilon$ with probability at least $1 - \delta$.*

Note that the normalized graph Laplacian $L = I - A$ has the same eigenvalues as A up to a shift and reflection, so Theorem 1.1 also yields a sublinear time result for normalized Laplacians, whose spectral densities are of interest in network science [11].

1.1.1 Robust spectral density estimation. Theorem 1.1 is proven in Section 5. A key component of the result is a sublinear time routine for computing coarse approximate matrix-vector products with any normalized graph adjacency matrix. To make use of such a routine, we need to develop an SDE algorithm that is *robust* to the use of an approximate matrix-vector oracle. This is one of the

[3]A standard adjacency list representation of the graph would support these operations. As discussed in Section 5, assumption (3) can be eliminated at the cost of an extra $\log n$ in the runtime as long as we know vertex degrees.

main contributions of our work, as previous methods assume exact matrix-vector products. Formally, we assume access to the oracle:

Definition 1.2. An ϵ_{MV}*-approximate matrix-vector multiplication oracle* for $A \in \mathbb{R}^{n\times n}$ and error parameter $\epsilon_{\mathsf{MV}} \in (0,1)$ is an algorithm that, given any vector $y \in \mathbb{R}^n$, outputs a vector z such that $\|z - Ay\|_2 \leq \epsilon_{\mathsf{MV}} \|A\|_2 \|y\|_2$. We will denote a call to such an oracle for by $\mathrm{AMV}(A, y, \epsilon_{\mathsf{MV}})$.

In Section 4.2 we prove the following for any Hermitian matrix A (e.g., real symmetric) under the assumption that $\|A\|_2 \leq 1$, i.e., that A's eigenvalues lie in $[-1,1]$:

Theorem 1.3 (Robust spectral density estimation). *Let $A \in \mathbb{R}^{n\times n}$ be a Hermitian matrix with spectral density s and $\|A\|_2 \leq 1$. Let C, C', C'' be fixed positive constants. For any $\epsilon, \delta \in (0,1)$ and $\epsilon_{MV} = C''\epsilon^{-3}\ln(1/\epsilon)$, there is an algorithm (Algorithm 1, with Algorithm 3 used as a subroutine to approximate moments) which makes $T = C\ell/\epsilon$ calls to an ϵ_{MV}-approximate matrix-vector oracle for A, where $\ell = \max\left(1, \frac{C'}{n}\epsilon^{-2}\log^2(\frac{1}{\epsilon\delta})\log^2(\frac{1}{\epsilon})\right)$, and in $\mathrm{poly}(1/\epsilon)$ additional runtime, outputs a probability density function $q : [-1,1] \to \mathbb{R}^{\geq 0}$ such that $W_1(s,q) \leq \epsilon$ with probability $1-\delta$.*

The requirement for the approximate matrix-vector oracle in Theorem 1.3 is relatively weak: we only need accuracy ϵ_{MV} that is polynomial in the final accuracy ϵ. Importantly, there is no dependence on $1/n$, which allows for the theorem to be combined with coarse AMV methods, including the one developed in Section 5 for normalized adjacency matrices. Based on random sampling, that method returns an ϵ-approximate matrix-vector multiply in $O(n/\epsilon^2)$ time. This immediately yields our result for graphs given by Theorem 1.1. We hope that Theorem 1.3 will find broader applications, since spectral density estimation is often applied to matrices where we only have inexact access to A. For example, A might be a Hessian matrix that we can multiply by approximately using stochastic approximation [30, 43], or the inverse of some other matrix, which we can multiply by approximately using an iterative solver.

We note that the result in Theorem 1.3 actually *improves* as n increases. Intuitively, when A is larger, each matrix-vector product returns more information about the spectral density s, so we can estimate it more easily. We also remark that the density function q returned by Algorithm 1 is in the form of an $O(1/\epsilon^3)$ dimensional vector, with the i-th entry corresponding to probability mass placed on the i-th point of an evenly spaced grid on $[-1,1]$. Alternatively, a simple rounding scheme that runs in $O(n + \mathrm{poly}(1/\epsilon))$ time can extract from q a vector of approximate eigenvalues $\tilde{\Lambda} = [\tilde{\lambda}_1, \ldots, \tilde{\lambda}_n]$ satisfying $\|\Lambda - \tilde{\Lambda}\|_1 \leq n\epsilon$, which, as discussed, is ϵ close to the spectral density s in Wasserstein distance (see Theorem B.1 [5]).

Our approach for density estimation is based on a moment matching method that approximates *Chebyshev polynomial* moments instead of the standard moments. I.e. we approximate $\mathrm{tr}(T_0(A)), \ldots, \mathrm{tr}(T_N(A))$ where $T_0, \ldots, T_N$ are the Chebyshev polynomials of the first kind and then return a distribution whose Chebyshev moments closely match our approximations. By leveraging Jackson's theorem on polynomial approximation of Lipschitz functions [17], we show how to bound the Wasserstein distance between two distributions in terms of the magnitude of the differences between their first $N = O(1/\epsilon)$ Chebyshev moments (see Lemma 3.1). Unlike results for standard moments [20], the bound shows a near-linear relationship between Wasserstein distance and difference in the Chebyshev moments. Ultimately this allows us to obtain a polynomial dependence on ϵ in the number of approximate matrix-vector multiplications needed in Theorem 1.3.

Along the way to proving that theorem, in Section 4.1 we first establish the follow result that is compatible with exact matrix-vector multiplications:

Theorem 1.4 (Linear time spectral density estimation). *Let $A \in \mathbb{R}^{n\times n}$ be a Hermitian matrix with spectral density s and $\|A\|_2 \leq 1$. Let C, C' be fixed positive constants. For any $\epsilon, \delta \in (0,1)$, there is an algorithm (Algorithm 1, with Algorithm 2 used as a subroutine to approximate moments) which computes $T = C\ell/\epsilon$ matrix-vector multiplications with A where $\ell = \max\left(1, \frac{C'}{n}\epsilon^{-2}\log^2(\frac{1}{\epsilon\delta})\log^2(\frac{1}{\epsilon})\right)$, and in $\mathrm{poly}(1/\epsilon)$ additional runtime, outputs a probability density function $q : [-1,1] \to \mathbb{R}^{\geq 0}$ such that $W_1(s,q) \leq \epsilon$ with probability $1-\delta$.*

As in Theorem 1.3, the theorem improves as n increases, requiring just $T = O(1/\epsilon)$ matrix vector multiplies when $n = \Omega(1/\epsilon^2)$. The runtime of Theorem 1.4 is dominated by the cost of the matrix-vector multiplications, which take $O(T \cdot n^2)$ time to compute for a dense matrix, and $O(T \cdot \mathrm{nnz}(A))$ time for a sparse matrix with $\mathrm{nnz}(A)$ non-zero entries, so the algorithm runs in linear time when ϵ, δ are considered constant.

Given Theorem 1.4, we prove Theorem 1.3 by showing that the error introduced by approximate matrix-vector multiplications does not hinder our ability to estimate the Chebyshev polynomial moments. We do so by drawing on stability results for the three-term recurrence relation defining these polynomials [7, 27].

Remark. The number of matrix-vector multiplies in Theorems 1.3 and 1.4, $N\ell = N \cdot \max(1, \frac{C'}{n}\epsilon^{-2}\log^2(\frac{1}{\epsilon\delta})\log^2(\frac{1}{\epsilon}))$, can be improved by up to a $\log^2(1/\epsilon)$ factor in the regime when n is small, specifically $n \leq C'\epsilon^{-2}\log^2(1/(\epsilon\delta))$. This is discussed further in Section 4.

1.1.2 *Spectral density estimation via the kernel polynomial method.*

In addition to the Chebyshev moment matching method used to give Theorem 1.4 and Theorem 1.3, we prove that a version of the popular kernel polynomial method (KPM) can be used to obtain a spectral density estimate with similar running times, albeit with slightly worse dependence on the accuracy parameter ϵ.[4] Along with the Stochastic Lanczos Quadrature method, the kernel polynomial method is one of two dominant spectrum estimation algorithms used in practice.

Given sufficiently accurate approximations to the Chebyshev polynomial moments, the KPM method outputs a density function q in the form of a $O(1/\epsilon)$ degree polynomial multiplied by a simple closed form function. This is described in Algorithm 6 in Section A.2 of the full version [5] and should be thought of as analagous to Algorithm 1. Specifically, we can obtain Theorem 1.4 and Theorem 1.3 with $\ell = \max(1, \frac{C'}{n}\epsilon^{-4}\log^2(\frac{1}{\epsilon\delta}))$ and $\epsilon_{\mathsf{MV}} = C''\epsilon^{-4}$ (in the robust setting), by using Algorithm 6 in Braverman et al. [5] instead of

[4] We believe that the extra $O(\epsilon^{-2})$ factor in the number of matrix-vector multiplications (or calls to an approximate matrix-vector oracle in the robust setting) may be an artifact of our analysis and can be further improved to match the approximate Chebyshev moment matching bounds.

Algorithm 1. Our proof in the KPM case is again based on Jackson's work on polynomial approximations for Lipschitz functions: we take advantage of the fact that Jackson constructs approximations that are both *linear* and *preserve positivity* [16].

1.2 Related Work

As mentioned, most closely related to our sublinear time result on graphs is the result of Cohen et al. [8]. They prove a result which matches the guarantee of Theorem 1.1, but with runtime of $2^{O(1/\epsilon)}$ – i.e., with *no dependence* on n. In comparison, our result depends linearly on n, but only polynomially on $1/\epsilon$. An interesting open question is if a poly$(1/\epsilon)$ time algorithm is possible but we conjecture that the trade-off between the dependence on n and the accuracy ϵ is inherent. Our bound in Lemma 3.1 on the Wasserstein-1 distance between two distributions can be seen as analagous to Proposition 1 from [20], which is the basis of the result in [8]. They bound the Wasserstein-1 distance between two distributions in terms of the differences in the standard moments of the distributions. The bound requires an exponentially small dependence on $1/\epsilon$, i.e. $2^{-O(1/\epsilon)}$, in the difference between the standard moments while the bound from Lemma 3.1 only requires an $O(\epsilon/\ln(1/\epsilon))$ difference in the Chebyshev moments.

As discussed, algorithms for spectral density estimation have been studied since the early 90s [34, 36, 39] but only analyzed recently. In addition to the work of Chen, Trogdon, and Ubaru that was discussed [6], [28] provides an algorithm for computing an approximate histogram for the spectrum of matrix. That result can be shown to yield an ϵ error approximation to the spectral density in the Wasserstein-1 distance with roughly $O(1/\epsilon^5)$ matrix-vector multiplications. This compares to the improved $O(1/\epsilon)$ matrix-vector multiplications required by our Theorem 1.4.

Matrix-vector query algorithms. Our work fits into a broader line of work on proving upper and lower bounds on the *matrix-vector query complexity* of linear algebraic problems, from top eigenvector, to matrix inversion, to rank estimation [4, 10, 24, 35, 37]. The goal in this model is to minimize the total number of matrix-vector multiplications with A, recognizing that such multiplications either 1) dominate runtime cost or 2) are the only way to access A when it is an implicit matrix. The matrix-vector query model generalizes both classical Krylov subspace methods, as well as randomized sketching methods [42]. Studying other basic linear algebra problem when matrix-vector multiplication queries are only assumed to be approximate (as in Definition 1.2) is an interesting future direction.

1.3 Paper Roadmap

We describe notation and preliminaries on polynomial approximation in Section 2. We use these tools in Section 3 to prove that a good approximation to the first $O(1/\epsilon)$ Chebyshev polynomial moments of the spectral density can be used to extract a good approximation in Wasserstein-1 distance. This result is the basis for our result on robust spectral density estimation stated in Theorem 1.4 and linear time spectral density estimation stated in Theorem 1.3, which are proven in Section 4. Finally, we give a randomized algorithm to implement an approximate matrix-vector multplication oracle for adjacency matrices in Section 5 and prove our main result, Theorem 1.1. In Section 6, we empirically investigate the potential of combining approximate matrix-vector multiplications with our moment matching method, the kernel polynomial method, and the stochastic Lanczos quadrature method studied in [6]. We show that all three can achieve accurate SDE estimates in sublinear time for a variety of graph Laplacians.

2 PRELIMINARIES

Throughout we assume that $A \in \mathbb{R}^{n\times n}$ is Hermitian with eigendecomposition $A = U\Lambda U^*$, where $UU^* = U^*U = I_{n\times n}$. We assume that A's eigenvalues satisfy $-1 \le \lambda_n \le \cdots \le \lambda_1 \le 1$. In many applications A is real symmetric. We denote A's spectral density by s, which is defined in (1). Our goal is to approximate s in the Wasserstein-1 metric with another distribution q supported on $[-1, 1]$. Specifically, as per the dual formulation given by the Kantorovich-Rubinstein theorem [18], for s, q supported on $[-1, 1]$ the metric is equal to:

$$W_1(s,q) = \sup_{\substack{f:\mathbb{R}\to\mathbb{R} \\ |f(x)-f(y)|\le|x-y|}} \left\{ \int_{-1}^{1} f(x)\left(s(x)-q(x)\right)dx \right\}. \quad (3)$$

In words, s and q are close in Wasserstein-1 distance if their difference has small inner product with all 1-Lipschitz functions f. Alternatively, $W_1(s, q)$ is equal to the cost of "changing" one distribution to another, where the cost of moving one unit of mass from x to y is $|x - y|$: this is the "earthmover's" formulation common in computer science. Note that (3) can be applied to arbitrary functions s, q, even if they are *not distributions*, and we will occasionally do so.

Functions and inner products. We introduce notation for functions used throughout the paper. Let $\mathcal{F}([-1, 1], \mathbb{R})$ denote the space of real-valued functions on $[-1, 1]$. For $g, h \in \mathcal{F}([-1, 1], \mathbb{R})$, let $\langle g, h\rangle$ denote $\langle g, h\rangle \coloneqq \int_{-1}^{1} g(x)h(x)dx$. For $f \in \mathcal{F}([-1, 1], \mathbb{R})$, we define $\|f\|_2 \coloneqq \sqrt{\langle f, f\rangle}$ and let $\|f\|_\infty$ denote the max-norm $\|f\|_\infty = \max_{x\in[-1,1]} |f(x)|$. We let $\|f\|_1$ denote $\|f\|_1 = \int_{-1}^{1} |f(x)|dx$.

Let $\mathcal{F}(\mathbb{Z}, \mathbb{R})$ be the space of real-valued functions on the integers, $\mathbb{Z}$. For $f, g \in \mathcal{F}(\mathbb{Z}, \mathbb{R})$ let $(f * g)$ denote the discrete convolution: $(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n - m]$. Let $\mathcal{F}(\mathbb{N}, \mathbb{R})$ be the space of real-valued functions on the natural numbers, $\mathbb{N}$. For functions in $\mathcal{F}(\mathbb{Z}, \mathbb{R})$ or $\mathcal{F}(\mathbb{N}, \mathbb{R})$ we typically used square brackets instead of parentheses.

For two functions f, g let $h = fg$ (or $h = f \cdot g$) and $j = f/g$ denote the pointwise product and quotient respectively. I.e. $h(x) = f(x)g(x)$ and $j(x) = f(x)/g(x)$ for all x.

Chebyshev polynomials. Our approach is based on approximating Chebyshev polynomial moments of A's spectral density, and we will use basic properties of these polynomials, the k^{th} of which we denote T_k. The Chebyshev polynomial of the first kind can be defined via the recurrence:

$$T_0(x) = 1 \qquad T_1(x) = x$$
$$T_k(x) = 2x \cdot T_{k-1}(x) - T_{k-2}(x) \qquad \text{for } k \ge 2.$$

We will use the well known fact that the Chebyshev polynomials of the first kind are bounded between $[-1, 1]$, i.e. $\max_{x\in[-1,1]} |T_k(x)| \le 1$.

Let $w(x) \coloneqq \frac{1}{\sqrt{1-x^2}}$. It is well known that $\langle T_0, w \cdot T_0 \rangle = \pi$, $\langle T_k, w \cdot T_k \rangle = \pi/2$ for $k > 0$, and

$$\langle T_i, w \cdot T_j \rangle = 0 \qquad \text{for } i \neq j.$$

In other words, the Chebyshev polynomials are orthogonal on $[-1, 1]$ under the weight function w. The first k Chebyshev polynomials form an orthogonal basis for the degree k polynomials under this weight function. We let $\bar{T}_k$ denote the *normalized* Chebyshev polynomial $\bar{T}_k \coloneqq T_k / \sqrt{\langle T_k, w \cdot T_k \rangle}$.

Definition 2.1 (Chebyshev Series)**.** The *Chebyshev expansion or series* for a function $f \in \mathcal{L}_\infty([-1, 1], \mathbb{R})$ is given by

$$\sum_{k=0} \langle f, w \cdot \bar{T}_k \rangle \cdot \bar{T}_k.$$

We call $\sum_{k=0}^{N} \langle f, w \cdot \bar{T}_k \rangle \cdot \bar{T}_k$ the truncated Chebyshev expansion or series of degree N.

Other notation. Let $[n]$ denote $1, \ldots, n$. For a scalar function $f : \mathbb{R} \to \mathbb{R}$ and $n \times n$ matrix A with eigendecomposition $U \Lambda U^*$, we let $f(A)$ denote the matrix function $U f(\Lambda) U^*$. Here $f(\Lambda)$ is understood to mean f applied entrywise to the diagonal matrix Λ containing A's eigenvalues. Note that $\mathrm{tr}(f(A)) = \sum_{i=1}^{n} f(\lambda_i)$. When $f(x)$ is a degree q polynomial, $c_0 + c_1 x + \ldots, c_q x^q$, then we can check that $f(A)$ exactly equals $c_0 I + c_1 A + \ldots, c_q A^q$, where I is then $n \times n$ identity matrix. So $f(A)y$ can be computed for any vector y using q matrix-vector multiplications with A.

3 APPROXIMATE CHEBYSHEV MOMENT MATCHING

In this section we show that the spectral density s of a Hermitian matrix A with eigenvalues in $[-1, 1]$ can be well approximated given access to *approximations* of the first $N = O(1/\epsilon)$ normalized Chebyshev polynomial moments of s, i.e., to approximations of $\mathrm{tr}(\bar{T}_1(A)), \ldots, \mathrm{tr}(\bar{T}_N(A))$. We state our result in Algorithm 1. We show later, in Section 4, a method to approximate these moments using a stochastic trace estimator, implemented with either exact or approximate matrix vector multiplications with A.

Given approximations $\tilde{\tau}_1, \ldots, \tilde{\tau}_N$ to the first N normalized Chebyshev moments of A, a natural approach is to find a probability density $q : [-1, 1] \to \mathbb{R}^+$ such that the first N normalized Chebyshev moments of q, i.e., $\langle \bar{T}_1, q \rangle, \ldots, \langle \bar{T}_N, q \rangle$, closely approximate $\tilde{\tau}_1, \ldots, \tilde{\tau}_N$. In order for this approximate moment matching approach to return a good spectral density estimate, it requires that: *for any density function q, if the first N Chebyshev moments of q closely approximate those of s, then q must be close to s in Wasserstein distance.* To that end, we prove the following lemma:

Lemma 3.1. *Let $N \in 4\mathbb{N}^+$ be a degree parameter and p, q be distributions on $[-1, 1]$.*

$$W_1(p, q) \leq \frac{36}{N} + 2 \sum_{k=1}^{N} \frac{|\langle \bar{T}_k, p \rangle - \langle \bar{T}_k, q \rangle|}{k}.$$

Lemma 3.1 shows that if the first N normalized Chebyshev moments of two distributions are identical, then the Wasserstein distance between the distributions is at most $O(1/N)$. When the moments between the distributions differ, the contribution of the difference between the k-th moments to the Wasserstein distance is scaled by $O(1/k)$. In particular, the lemma shows that deviation in the lower moments between distributions contributes more to the Wasserstein distance.

To prove Lemma 3.1, we will use two well-known results on approximating Lipschitz functions by polynomials. The first is proven in [17]. and concerns uniform approximation of Lipschitz continuous functions by a Chebyshev series:

Fact 3.2. *Let $f \in \mathcal{F}([-1, 1], \mathbb{R})$ be a Lipschitz continuous function with Lipschitz constant $\lambda > 0$. Then, for every $N \in 4\mathbb{N}^+$, there exists $N + 1$ constants $\hat{b}_N[0] > \cdots > \hat{b}_N[N] \geq 0$ such that the polynomial $\bar{f}_N = \sum_{k=0}^{N} \frac{\hat{b}_N[k]}{\hat{b}_N[0]} \langle f, w \cdot \bar{T}_k \rangle \bar{T}_k$ has the property that $\max_{x \in [-1,1]} |f(x) - \bar{f}_N(x)| \leq 18\lambda/N$.*

The coefficients of the polynomial in Fact 3.2 are not explicitly stated since we only require the existence of such a polynomial in order to prove Lemma 3.1. We defer the reader to Appendix A.1 in [5] for an explicit construction of the polynomial[5] and Appendix C.6 in the same for a proof of Fact 3.2.

Next, we state a well-known fact that the magnitude of the inner-product of a Lipschitz function f with the k-th Chebyshev polynomial (for $k \geq 1$) under the Chebyshev weight function $w = 1/\sqrt{1-x^2}$ is bounded by $O(1/k)$, i.e., $|\langle f, w \cdot \bar{T}_k \rangle| \leq O(1/k)$. Our proof is given in Appendix and is a simple adaptation of the proof of Theorem 4.2 in [38].

Fact 3.3. *Let $f \in \mathcal{F}([-1, 1], \mathbb{R})$ be a Lipschitz continuous function with Lipschitz constant $\lambda > 0$. Then, for any $k \geq 1$, we have that $|\langle f, w \cdot \bar{T}_k \rangle| = |\int_{-1}^{1} f(x) \bar{T}_k(x) w(x) dx| \leq 2\lambda/k$.*

With Fact 3.2 and 3.3 in place, we are now ready to prove Lemma 3.1

Proof of Lemma 3.1. Recall that the dual formulation of the Wasserstein-1 distance due to Kantorovich-Rubinstein gives us that $W_1(p, q) = \sup_{f \in \mathrm{lip}_1} \int_{-1}^{1} f(x)(p(x) - q(x))dx$ where lip_1 denotes the set of 1-Lipschitz functions on $[-1, 1]$. Let $f \in \mathrm{lip}_1$ be an arbitrary 1-Lipschitz function and let $\{\hat{b}_N[k]\}_{k=0}^{N}$ and $\bar{f}_N$ be the coefficients and polynomial respectively from Fact 3.2 for function f. We can then bound $W \coloneqq W_1(p, q)$ using the triangle inquality as

$$W \leq \underbrace{\int_{-1}^{1} |f(x) - \bar{f}_N(x)|(p(x) - q(x))dx}_{t_1} + \underbrace{\int_{-1}^{1} \bar{f}_N(p(x) - q(x))dx}_{t_2}$$

Using the fact that f is Lipschitz and the bound from Fact 3.2, along with the fact that p and q are distributions, we have that $t_1 \leq 36/N$.

It is left to bound t_2. We expand t_2 using the Chebyshev series expansion of $\bar{f}_N$ and note that $\langle g/w, w \cdot \bar{T}_k \rangle = \langle g, \bar{T}_k \rangle$ for any function $g \in \mathcal{F}([-1, 1], \mathbb{R})$, giving us

$$t_2 = \int_{-1}^{1} \bar{f}_N(x) w(x) \frac{p(x) - q(x)}{w(x)} dx$$
$$= \int_{-1}^{1} \bar{f}_N(x) w(x) \sum_{k=0}^{\infty} \langle p - q, \bar{T}_k \rangle \bar{T}_k(x) dx$$

[5] The construction of the polynomial $\bar{f}_N$ in Fact 3.2 and its uniform approximation to f forms the basis of our alternate approach, the Kernel Polynomial Method, which is discussed in-depth in Appendix A.1.

$$= \int_{-1}^{1} \left(w(x) \sum_{k=0}^{N} \frac{\hat{b}_N[k]}{\hat{b}_N[0]} \langle f, w\bar{T}_k \rangle \bar{T}_k(x) \right) \left(\sum_{k=0}^{\infty} \langle p - q, \bar{T}_k \rangle \bar{T}_k(x) \right) dx.$$

By the orthogonality of the Chebyshev polynomials under the weight function w and the fact that $\langle \bar{T}_k, \bar{T}_k \rangle = 1$ for all $k \in [N]$, we can bound the magnitude of t_2 as

$$|t_2| \leq \sum_{k=1}^{N} |\langle f, w \cdot \bar{T}_k \rangle| \cdot |\langle \bar{T}_k, p \rangle - \langle \bar{T}_k, q \rangle|$$

since we have that $0 \leq \hat{b}_N[k]/\hat{b}_N[0] \leq 1$ and $|\int_{-1}^{1} \bar{T}_k(p(x) - q(x))dx| = |\langle \bar{T}_k, p \rangle - \langle \bar{T}_k, q \rangle|$ for each $k \in [N]$. Additionally, since p and q are distributions we have that $\langle \bar{T}_0, s \rangle = \langle \bar{T}_0, z \rangle = 1/\sqrt{\pi}$. We then use the bound from Fact 3.3 on $|\langle f, w \cdot \bar{T}_k \rangle|$ for each $k \in [N]$. Putting this together gives us that $|t_2| \leq \sum_{k=1}^{N} 2|\langle \bar{T}_k, p \rangle - \langle \bar{T}_k, q \rangle|/k$.

Putting together the bound on t_1 and t_2 gives us the bound on $W_1(p, q)$. □

Moment matching algorithm. With Lemma 3.1 in place, our next step is develop a method to find a distribution q with Chebyshev moments closely matching a given set of target moments. In order to search for a distribution, we consider an evenly-spaced grid of the interval $[-1, 1]$. Specifically, let $d \in \mathbb{N}^+$ be a discretization parameter and let $X_d = [-1, -1 + \frac{2}{d}, \ldots, 1 - \frac{2}{d}, 1]$ be a $(d+1)$-length evenly-spaced grid of the interval $[-1, 1]$. Our goal is to output a distribution supported on X_d for an appropriately chosen value of d. Any such distribution can be described by a vector in $\mathbb{R}^d_{\geq 0}$ such that the i-th entry corresponds to the probability mass placed at point $-1 + 2i/d$ on the grid. Where it is clear from the context, we will denote the distribution and its probability mass vector interchangeably.

In order to compute the first N normalized Chebyshev moments of functions on the grid X_d, we define two matrices $\mathcal{T}_N^d, \widehat{\mathcal{T}}_N^d \in \mathbb{R}^{N \times d}$ such that for $k \in [N]$ and $i \in [d]$,

$$(\mathcal{T}_N^d)_{k,i} = \bar{T}_k(-1 + 2i/d) \quad \text{and} \quad (\widehat{\mathcal{T}}_N^d)_{k,i} = \frac{\bar{T}_k(-1 + 2id)}{k}.$$

The matrix $\mathcal{T}_N^d$ corresponds to a "discretization" of the continuous operator that computes the first N normalized Chebyshev moments of a continuous function on $[-1, 1]$. In particular, for a distribution q supported on X_d, we have that $\langle q, \bar{T}_k \rangle = \sum_{i=0}^{d} q_i \bar{T}_k(-1 + 2i/d) = (\mathcal{T}_N^d q)_k$. Notice that the matrix $\mathcal{T}_N^d$ does not contain the row for $\bar{T}_0$; since we are working with distributions we know that $\bar{T}_0(q) = 1/\sqrt{\pi} \cdot \int_{-1}^{1} q dx = 1/\sqrt{\pi}$ for any distribution q on $[-1, 1]$. The matrix $\widehat{\mathcal{T}}_N^d$ is the matrix $\mathcal{T}_N^d$ with the k-th row scaled by $1/k$. With this notation in place, we state the approximate moment matching algorithm in full in Algorithm 1.

Note that the optimization problem in Line 3 of Algorithm 1 can easily be written as a linear program in $O(d + N)$ variables and constraints and hence can be solved efficiently in $\text{poly}(N, d) = \text{poly}(1/\epsilon)$ time[6]. Since this method is independent of the matrix dimension n, it is a lower order term in the running time stated in Theorems 1.4 and 1.3, as we will discuss in Section 4.

Algorithm 1 Approximate Chebyshev Moment Matching

Input: Symmetric $A \in \mathbb{R}^{n \times n}$, degree parameter $N \in 4\mathbb{N}^+$, algorithm $\mathcal{M}(A)$ that computes moment approximations $\tilde{\tau}_1, \ldots, \tilde{\tau}_N$ with the guarantee that $|\tilde{\tau}_k - \frac{1}{n} \text{tr}(\bar{T}_k(A))| \leq (N \ln(eN))^{-1}$ for all k.

Output: A vector q corresponding to a discrete density function on $[-1, 1]$.

1: For $k = 1, \ldots, N$ use $\mathcal{M}$ to compute $\tilde{\tau}_1, \ldots, \tilde{\tau}_N$ and set $z = [\tilde{\tau}_1/1, \tilde{\tau}_2/2, \ldots, \tilde{\tau}_N/N]$.
2: Set $d = \lceil N^3/2 \rceil$ and compute matrix $\widehat{\mathcal{T}}_N^d \in \mathbb{R}^{N \times d}$. ▷ $(\widehat{\mathcal{T}}_N^d)_{k,i} = \bar{T}_k(-1 + \frac{2i}{d})/k$.
3: Minimize $\|\widehat{\mathcal{T}}_N^d q - z\|_1$ subject to $q^\top \vec{1} = 1$ and $q \geq 0$.
4: Return q.

We show that when $N = O(1/\epsilon)$, Algorithm 1 returns a distribution satisfying $W(s, q) \leq \epsilon$.

Lemma 3.4. *Let $\epsilon \in [0, 1]$ and let $N \geq 18/\epsilon$. Then the distribution $q : [-1, 1] \to \mathbb{R}^+$ returned by Algorithm 1 satisfies $W_1(q, s) \leq 3\epsilon$.*

Proof. We start by giving some notation – for a distribution $y : [-1, 1] \to \mathbb{R}^+$, we denote $\vec{\tau}_y := [\langle \bar{T}_1, y \rangle, \ldots, \langle \bar{T}_N, y \rangle]$ to be the vector of the first N normalized Chebyshev moments of y. For an integer $k \in \mathbb{N}^+$, we denote $\vec{k}$ to be the vector in $\mathbb{R}^k$ given by $\vec{k} := [1, \ldots, k]$ and for a vector $y \in \mathbb{R}^k$ write $y/\vec{k}$ to denote the vector $y/\vec{k} := [y_1/1, \ldots, y_k/k]$. Notice then that we have $\vec{\tau}_q = \mathcal{T}_N^d q$ and $\vec{\tau}_q/\vec{N} = \widehat{\mathcal{T}}_N^d q$.

We start by bounding the scaled differences in the first N normalized Chebyshev moments of q and s in order to use Lemma 3.1 on q and s: $\|\vec{\tau}_q/\vec{N} - \vec{\tau}_s/\vec{N}\|_1 \leq \|\vec{\tau}_q/\vec{N} - z\|_1 + \|z - \vec{\tau}_s/\vec{N}\|_1 \leq \|\vec{\tau}_q/\vec{N} - z\|_1 + \frac{1}{N}$. The first inequality follows by applying the triangle inequality and in the second inequality we used the fact that $\|z - \vec{\tau}_s/\vec{N}\|_1 = \sum_{k=1}^{N} |\tilde{\tau}_k - (\vec{\tau}_s)_k|/k \leq H_n \cdot (N \ln(eN))^{-1} \leq 1/N$.

Next we show that there exists a distribution q' supported on X_d such that $\|\vec{\tau}_{q'}/\vec{N} - z\| \leq 1/N$. To this end, consider the following distribution q^* on X_d:

$$q^*(x) = \frac{1}{n} \sum_{i=1}^{n} \delta(x - \underset{p \in X_d}{\text{argmin}} |p - \lambda_i|).$$

In words, q^* is the distribution corresponding to moving the mass from each λ_i to its nearest point on the grid X_d. Notice that we have $W_1(s, q^*) \leq 1/d$ due to the earthmover distance interpretation of the Wasserstein-1 distance.

Applying the triangle inequality and the guarantee from the moment approximations, we get that $\|\vec{\tau}_{q^*}/\vec{N} - z\|_1 \leq 1/N + \|\vec{\tau}_{q^*}/\vec{N} - \vec{\tau}_s/\vec{N}\|_1$. It is left then to bound $\|\vec{\tau}_{q^*}/\vec{N} - \vec{\tau}_s/\vec{N}\|_1$. To this end, we state the following well-known fact about the derivatives of Chebyshev polynomials.

Fact 3.5. *For $k \geq 1$, $\frac{dT_k(x)}{dx} = kU_{k-1}(x)$.*

[6] Additionally, note that the optimization problem has a convex objective and constraints – in particular, the set of distributions supported on X_d is a convex set. The objective function $\|\widehat{\mathcal{T}}_N^d q - z\|_1$ is not differentiable, but has subgradients. Hence, this program can be solved efficiently in $\text{poly}(1/\epsilon)$ time using a projected subgradient method. This requires an oracle that projects onto the the probability simplex supported on the grid X_d – an algorithm that runs in $O(d \log d)$ time has been given in multiple papers, see [40] for more details.

We then have using the definition of q^* that, for any $1 \le k \le N$,

$$|\langle \bar{T}_k, s\rangle - \langle \bar{T}_k, q^*\rangle| = \left|\frac{1}{n}\sum_{i=1}^{n} \bar{T}_k(\lambda_i) - \bar{T}_k(\operatorname*{argmin}_{p\in X_d}|p-\lambda_i|)\right|$$
$$\le \frac{1}{n}\sum_{i=1}^{n}\left|\bar{T}_k(\lambda_i) - \bar{T}_k(\operatorname*{argmin}_{p\in X_d}|p-\lambda_i|)\right|$$
$$\le \frac{\sqrt{2}}{n\sqrt{\pi}}\sum_{i=1}^{n}\max_{x\in[-1,1]}\left|\frac{dT_k(x)}{dx}\right|\cdot|\lambda_i - \operatorname*{argmin}_{p\in X_d}|p-\lambda_i|| \le \frac{\sqrt{2}k^2}{d\sqrt{\pi}}$$

where in the last inequality we used the fact that $\max_{x\in[-1,1]}|U_{k-1}(x)| \le k$. It follows then that

$$\|\vec{\tau}_{q^*}/\vec{N} - \vec{\tau}_s/\vec{N}\|_1 = \sum_{k=1}^{N}\frac{|(\vec{\tau}_{q^*})_k - (\vec{\tau}_s)_k|}{k} \le \frac{N(N+1)}{d\sqrt{2\pi}} \le \frac{1}{N}$$

by taking the sum over all k and noting that $d \ge N^3/2$. Putting these bounds together gives us that $\|\vec{\tau}_{q^*}/\vec{N} - z\|_1 \le 2/N$.

Since $\|\vec{\tau}_q/\vec{N} - z\|_1 \le \|\vec{\tau}_{q^*}/\vec{N} - z\|_1$ from Line 3 of Algorithm 1, we plug this into the bound on $\|\vec{\tau}_q/\vec{N} - \vec{\tau}_s/\vec{N}\|_1$ to get that $\|\vec{\tau}_q/\vec{N} - \vec{\tau}_s/\vec{N}\|_1 \le 3/N$. We can then use Lemma 3.1 with distributions q and s along with the fact that $\|\vec{\tau}_q/\vec{N} - \vec{\tau}_s/\vec{N}\|_1 = \sum_{k=1}^{N}|(\vec{\tau}_s)_k - (\vec{\tau}_q)_k|/k \le 3/N$ to give us the result since $N > 18/\epsilon$. □

Remark. Note that Algorithm 1 can easily be adapted when the minimization problem in Line 3 is solved approximately – as is the case if projected subgradient descent methods are used. In particular, a constant factor approximation to the minimal loss increases the Wasserstein distance bound in Lemma 3.4 by an $O(1)$ factor.

4 EFFICIENT CHEBYSHEV MOMENT APPROXIMATION

With Lemma 3.4 in place, we are ready to prove our main results. To do so, we need to show how to efficiently approximate the first N Chebyshev moments of a matrix A's spectral density s, as required by Algorithm 1. Recall that the k^{th} normalized Chebyshev moment of s is equal to $\langle s, \bar{T}_k\rangle = \frac{1}{n}\operatorname{tr}(\bar{T}_k(A))$. We will prove that this trace can be approximated using Hutchinson's stochastic trace estimator, implemented with either exact or approximate matrix-vector multiplications with A.

This estimator requires repeatedly computing $\bar{T}_k(A)g$ for a random vector g, which is done using the standard three-term (forward) recurrence for the Chebyshev polynomials and requires a total of k matrix-vector multiplications with A. We analyze the basic approach in Section 4.1, which yields Theorem 1.4. Then in Section 4.2, we argue that the approach is stable even when implemented with approximate matrix-vector multiplication, which yields Theorem 1.3.

4.1 Exact Matrix-Vector Multiplications

Hutchinson's estimator is a widely used estimator to efficiently compute accurate estimates of $\operatorname{tr}(R)$ for any square matrix $R \in \mathbb{R}^{n\times n}$. Each instance of the estimator computes the quadratic form $g^\top Rg$ for a random vector $g \in \{-1,1\}^n$ whose entries are Rademacher random variables. This an unbiased estimator for $\operatorname{tr}(R)$ with variance $\le 2\|R\|_F^2$, and its error has been analyzed in several earlier results [2, 32]. We apply a standard high-probability bound from [24, 33]:

Lemma 4.1 (Lemma 2, [24])**.**[7] *Let $R \in \mathbb{R}^{n\times n}$, $\delta \in (0, 1/2]$, $l \in \mathbb{N}$. Let $g^{(1)}, \ldots, g^{(\ell)} \in \{-1,1\}^{n\times n}$ be ℓ random vectors with i.i.d $\{-1,+1\}$ random entries. For a fixed constant C, with probability at least $1-\delta$,*

$$\left|\operatorname{tr}(R) - \frac{1}{\ell}\sum_{i=1}^{l}(g^{(i)})^\top Rg^{(i)}\right| \le \frac{C\log(1/\delta)}{\sqrt{\ell}}\|R\|_F.$$

For a polynomial $p \in \mathcal{F}([-1,1],\mathbb{R})$ with degree k, applying Hutchinson's estimator to $R = p(A)$ requires computing $p(A)g$, which can always be done with k matrix-vector multiplies with A. If $p(x)$ admits a recursive construction, like the Chebyshev polynomials, then this recurrence can be used. Specifically, for the Chebyshev polynomials, we have:

$$T_0(A)g = g \qquad T_1(A)g = Ag$$
$$T_k(A)g = 2A\cdot T_{k-1}(A)g - T_{k-2}(A)g \qquad \text{for } k \ge 2. \tag{4}$$

A moment estimation algorithm based on Hutchinson's estimator is stated as Algorithm 2.

Algorithm 2 Hutchinson Moment Estimator

Input: Symmetric $A \in \mathbb{R}^{n\times n}$ with $\|A\|_2 \le 1$, degree $N \in 4\mathbb{N}^+$, number of repetitions $\ell \in \mathbb{N}^+$.

Output: Approximation $\tilde{\tau}_k$ to moment $\frac{1}{n}\operatorname{tr}(\bar{T}_k(A))$ for all $k \in 1,\ldots,N$.

1: Draw $g^{(1)}, \ldots, g^{(l)} \sim \text{Uniform}(\{-1,1\}^n)$.

2: For $k = 1,\ldots,N$, $\tilde{\tau}_k \leftarrow \frac{\sqrt{2/\pi}}{\ell n}\sum_{i=1}^{l}(g^{(i)})^\top T_k(A)g^{(i)}$. ▷ Computed using recurrence in (4)

3: Return $\tilde{\tau}_1, \ldots, \tilde{\tau}_N$.

Remark. In total, Algorithm 2 requires $N\cdot\ell$ matrix multiplications with A since for each i $T_1(A)g^{(i)}, \ldots, T_N(A)g^{(i)}$ can but computed using the same N steps of the (4) recurrence. It requires $O(n\ell N)$ additional runtime to compute and sum all inner products of the form $(g^{(i)})^T T_k(A)g^{(i)}$.

Our main bound on the accuracy of Algorithm 2 follows:

Lemma 4.2. *If Algorithm 2 is run with $\ell = \max(1, C\cdot\frac{\log^2(N/\delta)}{(n\Delta^2)})$, where C is a fixed positive constant, then with probability $1-\delta$ the approximate moments returned satisfy $|\tilde{\tau}_k - \frac{1}{n}\operatorname{tr}(\bar{T}_k(A))| \le \Delta$ for all $k = 1,\ldots,N$.*

Proof. Fix $k \in \{1,\ldots,N\}$. Note that $\frac{\operatorname{tr}(\bar{T}_k(A))}{n} = \frac{\sqrt{2/\pi}}{n}\operatorname{tr}(T_k(A))$. Let C be the constant from Lemma 4.1. If $\ell = \max(1, C^2\frac{\log^2(N/\delta)}{(n\Delta^2)})$, then by that lemma we have that with probability at least $1-\delta/N$:

$$\left|\tilde{\tau}_k - \frac{\sqrt{2/\pi}}{n}\operatorname{tr}(T_k(A))\right| \le \frac{1}{n}\frac{C\log(N/\delta)}{\sqrt{\ell}}\|T_k(A)\|_F$$
$$\le \frac{C\sqrt{2/\pi}}{\sqrt{n}}\sqrt{\frac{\log(N/\delta)}{\ell}} \le \Delta.$$

[7] In [24] the lemma is stated with an assumption that $\ell > O(1/\delta)$. However, it is easy to see that the same claim holds without this assumption, albeit with a quadratically worse $\log(1/\delta)$ dependence. The proof follows from same application of the Hanson-Wright inequality used in that work.

The second to last inequality follows from the fact that $\|T_k(A)\|_2 \leq 1$ and thus $\|T_k(A)\|_F \leq \sqrt{n}$. Applying a union bound over all $k \in 1, \ldots, N$ gives the claim. □

Theorem 1.4 immediately follows as a corollary of Lemma 4.2 and Lemma 3.4.

Proof of Theorem 1.4. We implement Algorithm 1 with Algorithm 2 as a subroutine to approximate the Chebyshev polynomial moments, which requires setting $\Delta = \frac{1}{N\ln(eN)}$. By Lemma 4.2, we conclude that we need to set $\ell = \max(1, \frac{CN^2}{n}\log^2(\frac{N}{\delta})\log^2(eN))$. Then, by Lemma 3.4, setting $N = O(1/\epsilon)$ ensures that Algorithm 1 returns a distribution q which is ϵ close to A's spectral density s in Wasserstein distance. □

4.2 Approximate Matrix-Vector Multiplications

Algorithm 2 assumes access to an oracle for computing exact matrix-vector multiplies with A. In this section, we show that the method continues to work well even when each term in Hutchinson's estimator, $g^\top T_k(A)g$, is computed using an approximate matrix-vector multiplication oracle for A (see Definition 1.2). As discussed in Section 1.1, the robustness of the estimator allows the approximate moment matching method to be applied in many settings where A can only be access implicitly. It also forms the basis of our sublinear time algorithm for computing the spectral density of a normalized graph adjacency or Laplacian matrix, which are presented in the Section 5.

To show that approximate matrix-vector multiplications suffice, we leverage well understood stability properties of the three-term forward recurrence for Chebyshev polynomials of the first kind [7, 27]. These properties allows us to analyze the cumulative error when $T_k(A)g$ is computed via this recurrence. Specifically, we analyze the following algorithm:

Algorithm 3 Hutchinson Moment Estimator w/ Approximate Multiplications

Input: Symmetric $A \in \mathbb{R}^{n\times n}$ with $\|A\|_2 \leq 1$, degree $N \in 4\mathbb{N}^+$, number of repetitions $\ell \in \mathbb{N}^+$, ϵ_{MV}-approximate matrix vector multiplication oracle AMV for A (see Definition 1.2).

Output: Approximation $\tilde{\tau}_k$ to moment $\frac{1}{n}\operatorname{tr}(\bar{T}_k(A))$ for all $k \in 1, \ldots, N$.

1: **for** $i = 1, \ldots, \ell$ iterations **do**
2: Draw $g \sim \text{Uniform}(\{-1, 1\}^n)$.
3: $\tilde{v}_0 \leftarrow g, \tilde{v}_1 \leftarrow \text{AMV}(A, g, \epsilon_{\text{MV}})$.
4: $\tilde{\tau}_{1,i} \leftarrow g^T\tilde{v}_1$
5: **for** $k = 2$ to N **do**
6: $\tilde{v}_k \leftarrow 2 \cdot \text{AMV}(A, \tilde{v}_{k-1}, \epsilon_{\text{MV}}) - \tilde{v}_{k-2}$.
7: $\tilde{\tau}_{k,i} \leftarrow g^T\tilde{v}_k$
8: For $k = 1, \ldots, N$, $\tilde{\tau}_k \leftarrow \frac{1}{\ell}\sum_{i=1}^{\ell}\tilde{\tau}_{k,i}$.
9: Return $\tilde{\tau}_1, \ldots, \tilde{\tau}_N$.

Algorithm 3 assumes access to an approximate matrix-vector multiplication oracle for A with error ϵ_{MV} (recall Definition 1.2). Since $\|A\|_2 \leq 1$, for any vector y, we have that:

$$\|\text{AMV}(A, y, \epsilon_{\text{MV}}) - Ay\|_2 \leq \epsilon_{\text{MV}}\|y\|_2. \tag{5}$$

The algorithm uses this oracle to apply the recurrence from (4), approximately computing each $T_k(A)g$ for $k = 1, \ldots, N$, which in turn allows us to approximately compute $g^\top T_k(A)g$. Note that when $\epsilon_{\text{MV}} = 0$, Algorithm 3 is exactly equivalent to Algorithm 2.

Notation. Analyzing this approach requires accounting for error accumulates across iterations. To do so, we introduce some basic notation. Let v_k denote the true value of $T_k(A)g$, and let $\tilde{v}_k$ denote our computed approximation. We initialize the recurrence with $\tilde{v}_{-1} = \vec{0}$ and $\tilde{v}_0 = v_0 = g$. For $k = 0, \ldots, N-1$, let $w_k = \text{AMV}(A, \tilde{v}_k, \epsilon_{\text{MV}})$ and note that $\|w_k - A\tilde{v}_k\|_2 \leq \epsilon_{\text{MV}}\|\tilde{v}_k\|_2$. In iteration k of the recurrence, we compute $\tilde{v}_{k+1}$ by applying the recurrence:

$$\tilde{v}_{k+1} := 2w_k - \tilde{v}_{k-1}.$$

For each $i \in 0, \ldots, N$ we denote:

- $\delta_k := v_k - \tilde{v}_k$, with $\delta_0 = \vec{0}$. This is the *accumulated error* up to iteration k.
- $\xi_{k+1} := A\tilde{v}_k - w_k$, with $\xi_0 = 0$. $2\xi_{k+1}$ is the *new error* introduced in iteration k due to approximate matrix-vector multiplication.

As in Clenshaw's classic work [7], it can be shown that δ_k *itself evolves according to a simple recurrence*, which ultimately lets us show that it can be expressed as a summation involving Chebyshev polynomials of the *second* kind, which are easily bounded. Specifically, we have:

Fact 4.3. *$\delta_1 = \xi_1$ and for $2 \leq k \leq N$, $\delta_k = 2A\delta_{k-1} - \delta_{k-2} + 2\xi_k$.*

Proof. The claim for δ_1 is direct since $v_0 = \tilde{v}_0$: we have $\delta_1 = v_1 - \tilde{v}_1 = Av_0 - w_0$. For $2 \leq k \leq N$, we prove the claim by writing the difference $\delta_k = v_k - \tilde{v}_k = v_k - 2(A\tilde{v}_{k-1} + \xi_k) + \tilde{v}_{k-2}$. We can then replace $v_k = 2Av_{k-1} - v_{k-2}$ and substitute in $(v_{k-1} - \tilde{v}_{k-1}) = \delta_{k-1}$ and $(v_{k-2} - \tilde{v}_{k-2}) = \delta_{k-2}$. □

The Chebyshev polynomials of the second kind are defined via the following recurrence:

Definition 4.4 (Chebyshev Polynomials of the Second Kind)**.** For $k \in \mathbb{N}^{\geq 0}$ the k-th Chebyshev polynomial of the second kind $U_k(x)$ is given by

$$U_0(x) = 1 \qquad U_1(x) = 2x$$
$$U_k(x) = 2x \cdot U_{k-1}(x) - U_{k-2}(x) \qquad \text{for } k \geq 2.$$

We also define $U_{-1}(x) = 0$, which is consistent with the recurrence.

Using these polynomials, we can characterize the accumulated error δ_k in terms of the error introduced in each of the prior iterations.

Lemma 4.5. *For $k = 1, \ldots, N$, we have*

$$\delta_k = U_{k-1}(A)\xi_1 + 2\sum_{i=2}^{k} U_{k-i}(A)\xi_i. \tag{6}$$

Proof. We prove the lemma by induction on $j \leq k$. For $j = 0$, the lemma is trivial since $\delta_0 = 0$ by definition and $U_{-1}(A) = 0$. For $j = 1$, $\delta_1 = \xi_1 = U_0(A)\xi_1$. By Fact 4.3, for $2 \leq j < k$, we have:

$$\delta_j = 2\xi_j + \underbrace{2A\delta_{j-1} - \delta_{j-2}}_{z_1}. \tag{7}$$

We can apply the inductive hypothesis on z_1 and recombine terms using Definition 4.4 to get:

$$z_1 = 2A\left(U_{j-2}(A)\xi_1 + 2\sum_{i=2}^{j-1} U_{j-1-i}(A)\xi_i\right) - U_{j-3}(A)\xi_1 - 2\sum_{i=2}^{j-2} U_{j-2-i}(A)\xi_i$$
$$= U_{j-1}(A)\xi_1 + U_1(A)2\xi_{j-1} + \sum_{i=2}^{j-2}\left(2AU_{j-1-i}(A) - U_{j-2-i}(A)\right)2\xi_i$$
$$= U_{j-1}(A)\xi_1 + \sum_{i=2}^{j-1} U_{j-i}(A)2\xi_i$$

Noting that plugging into (7) and noting that $2\xi_j = 2U_0(A)\xi_j$ completes the proof. □

Our goal is to use Lemma 4.5 to establish that δ_k is small because each ξ_i is small. It is well known that the Chebyshev polynomials of the second kind satisfy the following bounds for any $k \in \mathbb{N}$:

$$|U_k(x)| \le k+1 \qquad \text{for} \qquad x \in [-1,1]. \tag{8}$$

This is the upper bound we need to proceed. Specifically, we will show that each estimator using Algorithm 3, $g^\top \tilde{v}_k$, well approximates Hutchinson's estimator $g^\top T_k(A)g = g^\top v_k$.

Claim 4.6. *For quantities $v_k, \tilde{v}_k$ and $0 \le \epsilon_{MV} \le 1/2k^2$, we have*

$$\left|g^\top T_k(A)g - g^\top \tilde{v}_k\right| \le 2\,\epsilon_{MV}\cdot(k+1)^2\|g\|_2^2.$$

Proof. By the definition of δ_k, we have $|g^\top T_k(A)g - g^\top \tilde{v}_k| = |g^\top \delta_k|$. By Cauchy-Schwarz we can bound $|g^\top \delta_k| \le \|g\|_2\|\delta_k\|_2$. We are left to bound $\|\delta_k\|_2$. Applying Lemma 4.5 and triangle inequality, we have

$$\|\delta_k\|_2 \le \|U_{k-1}(A)\|_2\|\xi_1\|_2 + \sum_{i=2}^{k} 2\|U_{k-i}(A)\|_2\|\xi_i\|_2$$

Then applying (8) and the fact that $\|A\|_2 \le 1$, we have $\|U_{k-i}(A)\|_2 \le (k-i+1)$. Hence,

$$\|\delta_k\|_2 \le k\|\xi_1\|_2 + \sum_{i=2}^{k} 2(k-i+1)\|\xi_i\|_2 \le \sum_{i=1}^{k} 2(k-i+1)\|\xi_i\|_2.$$

Using that $\xi_i \le \epsilon_{\mathsf{MV}}\|\tilde{v}_{i-1}\|_2$, and that $\|T_i(A)\|_2 \le 1$ for all i and thus $\|v_i\|_2 \le \|g\|_2$, we have:

$\|\delta_k\|_2 \le \sum_{i=1}^{k} 2(k-i+1)\,\epsilon_{\mathsf{MV}}\|\tilde{v}_{i-1}\|_2 \le 2\,\epsilon_{\mathsf{MV}}\sum_{i=1}^{k}(k-i+1)(\|v_{i-1}\|_2 + \|\delta_{i-1}\|_2) \le \epsilon_{\mathsf{MV}}\,k(k+1)\left(\|g\|_2 + \max_{i<k}\|\delta_i\|_2\right)$. Inducting on δ_j for $j \le k$ gives us $\|\delta_k\|_2 \le 2\,\epsilon_{\mathsf{MV}}(k+1)^2\|g\|_2$, which completes the proof. □

Lemma 4.7. *If Algorithm 3 is run with $\ell = \max(1, C\frac{\log^2(N/\delta)}{(n\Delta^2)})$ and $\epsilon_{MV} = \Delta/4N^2$, where C is a fixed positive constant, then with probability $1-\delta$ the approximate moments returned satisfy $|\tilde{\tau}_k - \frac{1}{n}\operatorname{tr}(\bar{T}_k(A))| \le \Delta$ for all $k = 1,\ldots,N$.*

Proof. Fix $k \in \{1,\ldots,N\}$. Let $g^{(1)},\ldots,g^{(\ell)}$ be the random vectors drawn in the outer for-loop of Algorithm 3. Let $\{\tilde{v}_k^{(i)}\}_{i\in[\ell]}$ be the ℓ vectors computed by the inner for-loop and let $\{\delta_k^{(i)} := \tilde{v}_k^{(i)} - T_k(A)g^{(i)}\}_{i\in[\ell]}$ be the ℓ error vectors. Recalling that $\frac{1}{n}\operatorname{tr}(\bar{T}_k(A)) = \frac{\sqrt{2/\pi}}{n}\operatorname{tr}(T_k(A))$, we have:

$$\left|\tilde{\tau}_k - \frac{\sqrt{2/\pi}}{n}\operatorname{tr}(T_k(A))\right| \le \frac{\sqrt{2/\pi}}{n\ell}\sum_{i=1}^{\ell}\left|(g^{(i)})^\top\delta_k^{(i)}\right| + \left|\frac{\sqrt{2/\pi}}{n\ell}\sum_{i=1}^{\ell}(g^{(i)})^\top T_k(A)g^{(i)} - \frac{\operatorname{tr}(T_k(A))}{n}\right|.$$

Applying Claim 4.6 and Lemma 4.1, with probability at least $1-\delta/N$, we thus have

$$|\tilde{\tau}_k - \frac{\operatorname{tr}(\bar{T}_k(A))}{n}| \le 2(k+1)^2\,\epsilon_{\mathsf{MV}}\cdot\frac{\sqrt{2/\pi}}{n\ell}\sum_{i=1}^{\ell}\|g^{(i)}\|_2^2 + \Delta/2 \le \Delta.$$

The last inequality follows from the fact that $\|g^{(i)}\|_2^2 = n$ for all $i \in [\ell]$, and the choice of $\epsilon_{\mathsf{MV}} = \Delta/4N^2$. Applying a union bound over all $k = 1,\ldots,N$ gives the claim. □

Theorem 1.3 immediately follows.

Proof of Theorem 1.3. We implement Algorithm 1 with Algorithm 3 used as a subroutine to approximate the Chebyshev polynomial moments, which requires setting $\Delta = \frac{1}{N\ln(eN)}$. By Lemma 4.7, we conclude that we need to set $\ell = \max(1, C\frac{N^2}{n}\log^2(\frac{N}{\delta})\log^2(eN))$ and $\epsilon_{\mathsf{MV}} = 1/(4N^3\ln(eN))$. Then, by Lemma 3.4, setting $N = O(1/\epsilon)$ ensures that Algorithm 1 returns a distribution q which is ϵ close to A's spectral density s in Wasserstein distance. □

5 SUBLINEAR TIME METHODS FOR GRAPHS

With the proof of Theorem 1.3 in place, we are now ready to state our sublinear time result for adjacency matrices of graphs. The significance of Theorem 1.3 is that it allows for the approximate Chebyshev moment matching method in Algorithm 1 to be combined with any randomized algorithm for approximating matrix-vector multiplications with A. In this section we prove Theorem 1.1 by showing that for the normalized adjacency matrix of any undirected, un-weighted graph, such an algorithm can actually be implemented in *sublinear time*, leading to a sublinear time spectral density estimation (SDE) algorithm for computing graph spectra from these matrices.

Computational Model. Let $A \in \mathbb{R}^{n\times n}$ be the adjacency matrix for an unweighted, n-vertex graph $G = (V,E)$ and let $\bar{A} = D^{-1/2}AD^{-1/2}$ be the symmetric normalized adjacency matrix, where D is an $n\times n$ diagonal matrix containing the degree of each vertex in V. For a node i, let $\mathcal{N}(i) = \{j : (j,i) \in E\}$ denote the set of i's neighboring vertices. We assume a computational model where we can 1) uniformly sample a random vertex in constant time, 2) uniformly sample a random neighbor of any vertex i in constant time, and 3) for a vertex i with degree d_i, read off all neighbors of i in $O(d_i)$ time. A standard adjacency list representation of the graph would allow us to perform these operations but weaker access models would also suffice.[8]

[8] E.g., random crawl access to a network [19]. We also note that, if desired, assumption 3) can be removed entirely with a small logarithmic runtime overhead, as long as we know the degree of i. Specifically, 3) can be implemented with $O(d_i \log n)$ calls to 2): we simply randomly sample neighbors until all d_i are found. A standard

Using this model for accessing the adjacency matrix, we show that, for any $\epsilon_{\mathsf{MV}} \in (0,1)$ and failure probability $\delta \in (0,1)$, an ϵ_{MV}-approximate matrix-vector multiplication oracle for $\bar{A}$ can be implemented in $O(n\,{\epsilon_{\mathsf{MV}}}^{-2}\log(1/\delta))$ time. Via Theorem 1.3, this immediately yields an algorithm for computing an SDE that is ϵ close in Wasserstein-1 distance to $\bar{A}$'s spectral density in roughly $\tilde{O}(n/\epsilon^7)$ time for sufficiently large n, and at most $\tilde{O}(n/\epsilon^9)$ time, for fixed δ where the $\tilde{O}(\cdot)$ hides factors of $\mathrm{poly}(\log(1/\epsilon))$. Our main result is stated as Theorem 1.1 in Section 1.1.

The same algorithm can be used to approximate the spectral density of the normalized Laplacian of G by a simple shift and scaling. Specifically, $\bar{A}$ can be obtained from the normalized Laplacian $\bar{L}$ via $\bar{A} = I - \bar{L}$, and the spectral density of $\bar{L}$, $s_{\bar{L}}(x)$ satisfies $s_{\bar{L}}(1-x) = s_{\bar{A}}(x)$, where $s_{\bar{A}}$ is the spectral density of $\bar{A}$. So if we obtain an ϵ-approximate SDE q for $\bar{A}$ by Theorem 1.1, then the function p satisfying $p(1-x) = q(x)$ is an ϵ-approximate SDE for $s_{\bar{L}}$. We thus have:

Corollary 5.1. *Given the the normalized adjacency matrix of G, there exists an algorithm that takes $O(n\,\mathrm{poly}(\frac{\log(1/\delta)}{\epsilon}))$ expected time and outputs a density function q that is ϵ close to the spectral density of the normalized Laplacian of G with probability at least $1-\delta$.*

Approximate Matrix-Vector Multiplication for Adjacency Matrices. We implement an approximate matrix-vector multiplication oracle for $\bar{A}$ in Algorithm 4, which is inspired by a randomized matrix-multiplication method of [12]. Throughout this section, let $\bar{A}^i$ denote the i^{th} column of $\bar{A}$. Given a sampling budget $t \in \mathbb{N}$, the algorithm samples t indices from $1,\ldots,n$ independently and with replacement – i.e., the same index might be sample multiple times. For each index it samples, the algorithm decides to accept or reject the column corresponding to that index with some probability. To approximate $\bar{A}y$, the algorithm outputs the multiplication of the accepted columns, rescaled appropriately, with the corresponding elements of y.

Algorithm 4 AMV Multiplication Oracle for Normalized Adjacency Matrices

Input: Normalized adjacency matrix $\bar{A} \in \mathbb{R}^{n\times n}$, degrees $[d_1,\ldots,d_n]$, $y \in \mathbb{R}^n$, and parameter $t \in \mathbb{N}$.
Output: A vector $z \in \mathbb{R}^n$ that approximates $\bar{A}y$.

Initialize $z \leftarrow \vec{0}$.
for t iterations **do**
  Sample a node j uniformly at random from $\{1,\ldots,n\}$.
  Sample a neighbor $i \in \mathcal{N}(j)$ uniformly at random.
  Sample x uniformly at random from $[0,1]$.
  if $x \le \frac{1}{d_i}$ **then**
    $w \leftarrow \frac{1}{p_i}\cdot y_i\bar{A}^i$ where $p_i = \frac{1}{nd_i}\sum_{j\in\mathcal{N}(i)}\frac{1}{d_j}$.
  else
    $w \leftarrow \vec{0}$.
  $z \leftarrow z + w$.
return $\frac{1}{t}z$

analysis of the coupon collector problem [Section 3.6, 25] shows that that the expected number of samples will be $O(d_i\log d_i) \le O(d_i\log n)$.

The following lemma bounds the expected squared error of Algorithm 4's:

Lemma 5.2. *Let $z \in \mathbb{R}^n$ be the output of Algorithm 4 with sampling budget t. We have:*

$$\mathbf{E}[\|\bar{A}y - z\|_2^2] = \frac{n}{t}\|y\|_2^2 - \frac{1}{t}\|\bar{A}y\|_2^2$$

Proof. Let b denote $b = \bar{A}y$. Consider a single iteration of the main loop in Algorithm 4, which generates a vector w that is added to z. Let X_i be an indicator random variable that is 1 if w is set to a scaling of $\bar{A}^i$ on that iteration, and 0 otherwise. $X_i = 1$ if and only if 1) a neighbor of i is sampled at Line 3 of the algorithm, 2) i is sampled at Line 4 of the algorithm, and 3) the uniform random variable x satisfies $x < 1/d_i$. So, we see that $\mathbf{Pr}[X_i = 1]$ is exactly equal to $p_i = \frac{1}{nd_i}\sum_{j\in\mathcal{N}(i)}\frac{1}{d_j}$. It follows that, by the time we reach Line 11, w is an unbiased estimator for b. I.e., $\mathbf{E}[w] = b$. Of course, this also implies that $\mathbf{E}[z] = b$.

Our goal is to show that $\mathbf{E}[\|b-z\|^2] = \frac{n}{t}\|y\|_2^2 - \frac{1}{t}\|b\|_2^2$. Since the random vector $b - z$ has mean zero and is the average of t i.i.d. copies of the mean zero random vector $b - w$, it suffices that show:

$$\mathbf{E}[\|b-w\|_2^2] = n\|y\|_2^2 - \|b\|_2^2. \tag{9}$$

By linearity of expectation and the fact that $\mathbf{E}[w] = b$, we have

$$\mathbf{E}[\|b-w\|_2^2] = \|b\|_2^2 + \mathbf{E}[\|w\|_2^2] - 2\langle\mathbf{E}[w], b\rangle = \mathbf{E}[\|w\|_2^2] - \|b\|_2^2.$$

So to prove (9), we need to show that $\mathbf{E}[\|w\|_2^2] = n\|y\|_2^2$. We expand w in terms of the indicator random variables $X_1,\ldots,X_n$. Notice that since we only sample one column in each iteration, the random variable $X_iX_j = 0$ for all $i \ne j$. Thus we have

$$\begin{aligned}\mathbf{E}[\|w\|_2^2] &= \sum_{k=1}^n \mathbf{E}\left[\sum_{i,j\in[n]}\frac{X_iX_j}{p_ip_j}(\bar{A}^iy_i)_k(\bar{A}^jy_j)_k\right]\\ &= \sum_{k=1}^n \mathbf{E}\left[\sum_{i=1}^n\frac{X_i^2}{p_i^2}(\bar{A}^iy_i)_k^2\right] = \sum_{i=1}^n\frac{1}{p_i}\cdot\|\bar{A}^iy_i\|_2^2 = \sum_{i=1}^n ny_i^2.\end{aligned}$$

In the last equalities we used the fact that $\mathbf{E}[X_i^2] = p_i$ and that, for a normalized graph adjacency matrix, $\|\bar{A}^i\|_2^2 = \sum_{j\in\mathcal{N}(i)}\frac{1}{d_id_j} = np_i$. This proves (9), from which we conclude the lemma. □

Using Lemma 5.2, we show that there is an ϵ_{MV}-approximate matrix-vector oracle for $\bar{A}$ based on Algorithm 4 with success probability at least $1-\delta$ that runs in $O(n\,{\epsilon_{\mathsf{MV}}}^{-2}\log^2(\frac{1}{\delta}))$ time.

Proposition 5.3. *Let $\bar{A} \in \mathbb{R}^{n\times n}$ be the symmetric normalized adjacency matrix of an n-vertex graph G and let $\epsilon_{MV}, \delta \in (0,1)$ be fixed constants. There is an algorithm that, given a vector $y \in \mathbb{R}^n$, and access to G as described above, takes $O(n\,{\epsilon_{MV}}^{-2}\log(\frac{1}{\delta}))$ expected time and outputs a vector $z \in \mathbb{R}^n$ such that $\|z - \bar{A}y\|_2 \le \epsilon_{MV}\|y\|_2$ with probability at least $1-\delta$.*

Proof. By Lemma 5.2, we have that $\mathbf{E}[\|\bar{A}y - z\|_2^2] \le \frac{n}{t}\|y\|_2^2$. Fix $t = 48n\,{\epsilon_{\mathsf{MV}}}^{-2}$. Then, by Lemma 5.2 and Markov's inequality, we have that when Algorithm 4 is called on $\bar{A}$ with parameter t,

$$\mathbf{Pr}[\|\bar{A}y - z\|_2 > \frac{\epsilon_{\mathsf{MV}}}{4}\|y\|_2] \le \frac{16n\|y\|_2^2}{t\,{\epsilon_{\mathsf{MV}}}^2\|y\|_2^2} \le \frac{1}{4}. \tag{10}$$

In order improve our success probability from 3/4 to $1-\delta$, we use the standard trick of repeating the above process $r = c\log(\frac{1}{\delta})$ times for a constant c to be fixed later. Let $z_1, \ldots, z_r \in \mathbb{R}^n$ be the output of running Algorithm 4 r times with parameter t. We can return as our estimate for $\bar{A}y$ the first z_i such that $\|z_i - z_j\|_2 \leq \frac{\epsilon_{\mathsf{MV}}}{2}\|y\|_2$ for at least $r/2+1$ vectors z_j from $z_1, \ldots, z_n$.

To see why this works, note that a Chernoff bound can be used to claim that with probability $> 1-\delta$, at least $r/2+1$ vectors z_j from $z_1, \ldots, z_r$ have that $\|z_j - \bar{A}y\|_2 \leq \frac{\epsilon_{\mathsf{MV}}}{4}\|y\|_2$.

By a triangle inequality we have that for all such z_j and z_k,

$$\|z_j - z_k\|_2 \leq \|z_j - \bar{A}y\|_2 + \|z_k - \bar{A}y\|_2 \leq \frac{\epsilon_{\mathsf{MV}}}{2}\|y\|_2.$$

Thus, the z_i we picked must satisfy that $\|z_i - \bar{A}y\| \leq \frac{3\,\epsilon_{\mathsf{MV}}}{4}\|y\|_2$ by the triangle inequality.

All that remains is to bound the expected runtime of Algorithm 4, which we will run r separate times. To do so, note that all index sampling can be done in just $O(t)$ time, since sampling a random vertex and a random neighbor of the vertex are assumed to be $O(1)$ time operations. The costly part of the algorithm is computing the sampled column w at each iteration. In the case that $w = \vec{0}$, this cost is of course zero. However, when $w = \frac{1}{p_i}\bar{A}^i y_i$ for some i, computing the column and adding it to z takes $O(d_i)$ time, which can be large in the worst case. Nevertheless, we show that it is small in expectation. This may seem a bit surprising: while nodes with high degree are more likely to be sampled by Line 4 in Algorithm 4, they are rejected with higher probability in Line 6. Formally, let $\mathrm{nnz}(w)$ denote the number of non-zero entries in w. We have: $\mathbf{E}\left[\mathrm{nnz}(w)\right] = \sum_{i=1}^n \mathrm{nnz}(\bar{A}^i) \cdot p_i = \sum_{i=1}^n \sum_{j\in\mathcal{N}(i)} \frac{d_i}{n\cdot d_i d_j} = \frac{1}{n}\sum_{i=1}^n \sum_{j\in\mathcal{N}(i)} \frac{1}{d_j} = 1$.

The final equality follows from expanding the double sum: since node j has exactly d_j neighbors, $\frac{1}{d_j}$ appears exactly d_j times in the sum. So $\sum_{i=1}^n \sum_{j\in\mathcal{N}(i)} \frac{1}{d_j} = n$.

We run Algorithm 4 with $t = O(n/{\epsilon_{\mathsf{MV}}}^2)$ iterations, so it follows that the expected total sparsity of all w's constructed equals $O(n/{\epsilon_{\mathsf{MV}}}^2)$, which dominates the expected running time of our method. □

Proof of Theorem 1.1. The accuracy and running time claim follows from combining the ϵ_{MV}-approximate vector multiplication oracle described in Proposition 5.3 with Algorithm 1, which is analyzed in Theorem 1.3. □

As discussed in the introduction, Cohen et al. [8] prove a result which matches the guarantee of Theorem 1.1, but with runtime of $2^{O(1/\epsilon)}$ – i.e., with *no dependence* on n. In comparison, our result depends linearly on n, but only polynomially on $1/\epsilon$. In either case, the result is quite surprising, as the runtime is *sublinear* in the input size: A could have up to $O(n^2)$ non-zero entries.

6 EXPERIMENTS

We support our theoretical results by implementing our Chebyshev moment matching method (Algorithm 1). When using exact matrix-vector multiplications, the kernel polynomial method (KPM) of Algorithm 6 [5] and the stochastic Lanczos quadrature method (SLQ) studied in [6] have both been confirmed to work well empirically. So, one set of experiments is aimed at comparing these methods to the moment matching method (MM) implemented with exact matrix-vector multiplications. A second set of experiments evaluates the performance of the MM and KPM methods when implemented with approximate matrix-vector multiplies. Specifically, we use our sublinear time randomized method for multiplication by graph adjacency matrices from Section 5.

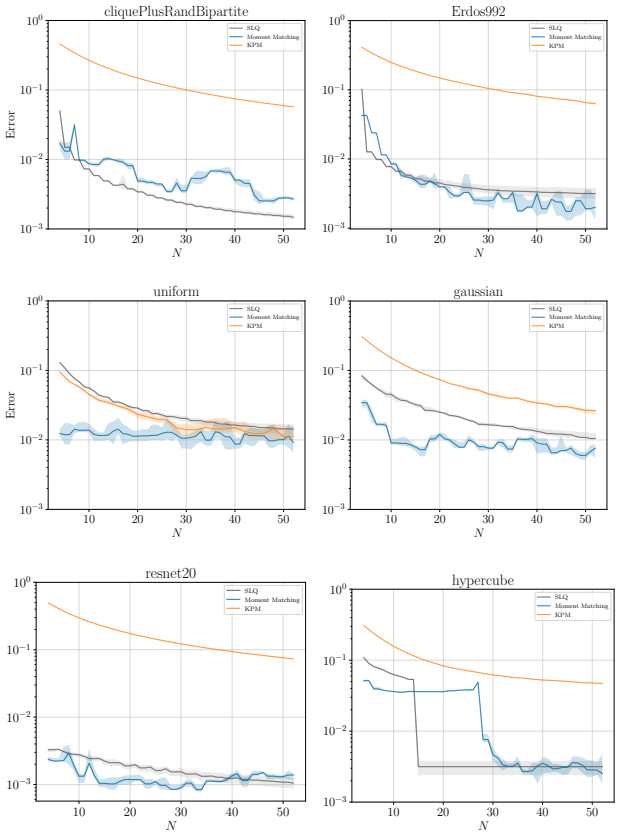

Figure 2: Wasserstein error of density estimate resulting from approximate Chebyshev moment matching method (MM), the Jackson damped kernel polynomial method (KPM) and Stochastic Lanczos Quadrature (SLQ) method. For MM and KPM, Hutchinson's estimator is used to estimate the Chebyshev moments. The x-axis corresponds to the number of moments computed for MM and KPM, and the number of Lanczos iterations used for SLQ. All methods use 5 (random) starting vectors except for resnet20 and hypercube that use 1 starting vector, so the x-axis is directly proportional to the number of matrix-vector multiplications used by each method. Each experiment is repeated 10 times; the solid line represents the median error of the 10 trials and the shaded regions represent the first and third quartiles.

We consider the normalized adjacency matrix of three graphs, two of which we construct and one which we obtain from a publicly available dataset for sparse matrices:

- `cliquePlusRandBipartite` is a graph with 10000 vertices, partitioned into two disconnected components. The first component is a clique with 5000 nodes and the second is

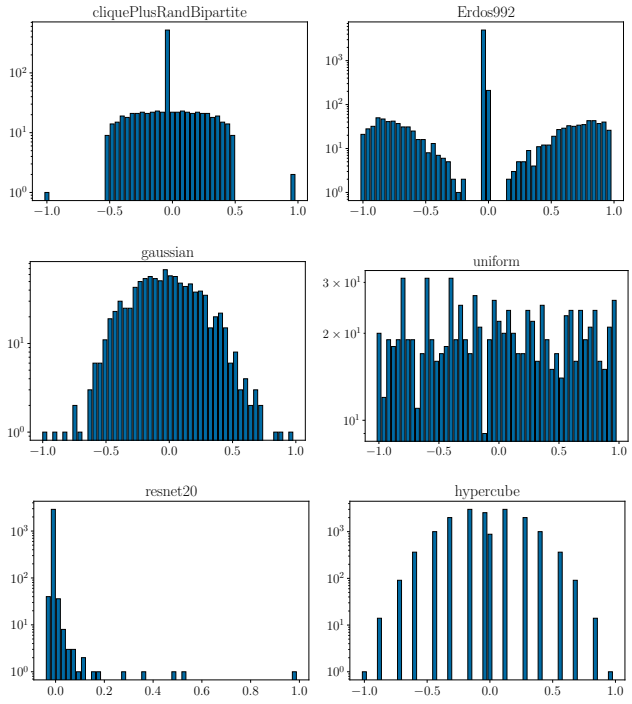

Figure 3: Histograms of the eigenvalues of `cliquePlusRandBipartite`, `Erdos992`, `gaussian`, `uniform`, `resnet20` and `hypercube` using 50 equally spaced buckets.

a bipartite graph with 2500 vertices in each partition, constructed by sampling each of the 2500^2 possible edges independently with probability 0.05. This graph has a normalized adjacency matrix with $\sim$ 5000 eigenvalues at 0, two eigenvalues at 1, one at -1 and the rest of its eigenvalues are roughly evenly spread out between -0.5 and 0.5.

- `hypercube` is a 16384 vertex boolean hypercube graph on 14 bit strings.[9] Its normalized adjacency matrix has eigenvalues at $-1, -\frac{6}{7}, \frac{-5}{7}, \ldots, 0, \ldots, \frac{6}{7}, 1$. The multiplicity of the 0 eigenvalue is largest, with eigenvalues closer to -1 and 1 having lower multiplicity.
- `Erdos992` is an undirected graph with 6100 vertices, containing 15030 edges from the sparse matrix suite of [9]. Its normalized adjacency matrix has $\sim$ 5000 eigenvalues at 0, one at 1 and the rest evenly spread out between -0.5 and 0.5.

We consider three additional matrices to evaluate the performance of MM against KPM and SLQ when exact matrix-vector multiplies are used to estimate the Chebyshev moments:

- `gaussian` is a 1000×1000 matrix constructed by drawing $n = 1000$ Gaussian random variables $\lambda_1, \cdots, \lambda_n \sim \mathcal{N}(0, 1)$ and a random orthogonal matrix $U \in \mathbb{R}^{n \times n}$, and outputting $U \Lambda U^\top$ where Λ is a $n \times n$ diagonal matrix with entries $\frac{\lambda_1}{\max_i \lambda_i}, \ldots, \frac{\lambda_n}{\max_i \lambda_i}$.
- `uniform` is a 1000×1000 matrix constructed identically to `gaussian` except with $\lambda_1, \ldots, \lambda_n$ drawn independently and uniformly from the interval $[-1, 1]$.
- `resnet20` is a Hessian for the ResNet20 network [14] trained on the Cifar-10 dataset. The matrix is 3000×3000 and its eigenvalues have been normalized to lie between $[-1, 1]$ for our experiments.

For reference, the histogram of the eigenvalues for each matrix are shown in Figure 3 by breaking the range of the eigenvalues into 50 equally spaced intervals for each matrix.

In the first set of experiments, we compute the normalized Chebyshev moments $\tau_1, \ldots, \tau_N$ of each of the six aforementioned matrices using Hutchinson's moment estimator as in Algorithm 2, and, compute a spectral density estimate by passing these moments into Algorithm 1 for approximate Chebyshev moment matching method (MM)[10] and into Algorithm 6 [5] for the Jackson damped kernel polynomial method (KPM). For KPM we compute the density with $N = 4, 6, 8, 10, \ldots, 52$ and for MM we compute it with $N = 4, 5, 6, 7, \ldots, 52$. We also compute the density estimate resulting from the stochastic Lanczos quadrature (SLQ) method of [6] with $N = 4, 5, 6, 7, \ldots, 52$ Lanczos iterations. We use $\ell = 5$ starting vectors (i.e., random vectors in Hutchinson's method, or random restarts of the SLQ method) for each method, except for the large `resnet20` and `hypercube` matrices, for which $\ell = 1$ random vector is used. Each experiment is repeated 10 times and the Wasserstein-error between the true density and the density estimate are shown in Figure 2. The results show that MM is more than 10x more accurate than KPM in almost all cases. The error of MM and SLQ are more comparable, except for `hypercube`, on which the errors are comparable for larger values of N. Both methods show an unusual convergence curve for this matrix, which we believe is related to the sparsify of its spectrum (a small number of distinct eigenvalues).

In our second set of experiments, we test the performance of our randomized sublinear time algorithm (Algorithm 4) for approximate matrix-vector multiplies with normalized graph adjacency matrices. This method is used to estimate Chebyshev moments in Algorithm 1 (MM) and in Algorithm 6 (KPM) of [5]. We compute the normalized Chebyshev moments $\tau_1, \ldots, \tau_N$ for $N = 12$ using various values of the oversampling parameter t in the approximate matrix-vector multiplication method. We then compute, for each value of t, the average number of non-zero elements of A accessed by the method for each matrix-vector product, which reflects the runtime improvement over a full matrix-vector product. Figure 4 plots the Wasserstein error of the density estimate (y-axis) and the average fraction of non-zeros used in each matrix-vector multiplication (x-axis) to estimate the Chebyshev moments used by MM and KPM respectively.

The results show that the KPM method can achieve error nearly identical to that obtained when using exact matrix-vector multiplications, while only using a small fraction of non-zero entries for each approximate matrix-vector multiplication. Specifically, on the dense `cliquePlusRandBipartite` graph and even the relatively sparse `hypercube` graph, KPM uses less than 15% of the non-zero entries on average to achieve nearly the same error as

[9] A boolean hypercube contains a vertex for each distinct b bit string, and an edge between two vertices if the corresponding strings differ on exactly 1 bit.

[10] We solve the optimization problem from Line 3 by formulating it as a linear program and using an off-the-shelf solver from `scipy`.

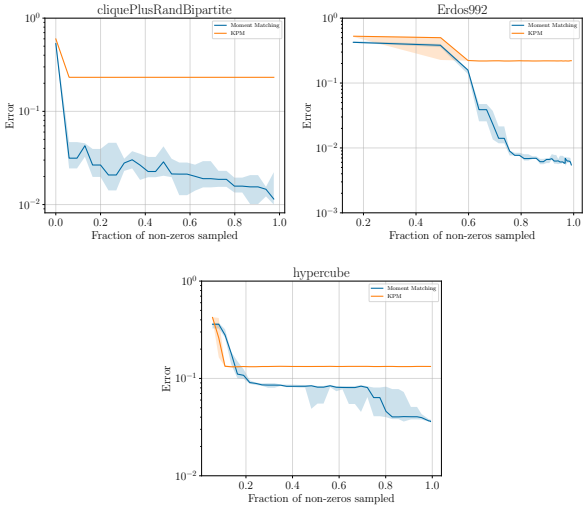

Figure 4: Wasserstein error of density estimate returned by MM and KPM on the `hypercube`, `cliquePlusRandBipartite` and `Erdos992` graphs using approximate matrix-vector multiplications (Algorithm 4) to estimate the Chebyshev moments. For both methods, $N = 32$ moments are computed using 5 random starting vectors for `cliquePlusRandBipartite` and `Erdos992` and 1 for `hypercube`. The x-axis corresponds to the average fraction of non-zeros sampled from the matrix and the y-axis is the Wasserstein error from the resulting density estimate. Each experiment is repeated for 10 trials: the solid line correponds to the median error of the 10 trials and the shaded region corresponds to the first and third quartiles.

when using exact multiplies. On `cliquePlusRandBipartite`, the MM method achieves error close to that of the exact method while using $\sim$ 20% of the non-zero entries on average. On the sparse `Erdos992` and `hypercube` graphs, the MM method requires $\sim$ 80% of the non-zero entries on average to achieve error comparable to exact matrix-vector multiplications. However, it still obtains a good approximation (consistently better than the KPM method) when coarse matrix-vector multiplications are used (i.e., fewer non-zeros are sampled).

ACKNOWLEDGMENTS

We thank Cameron Musco, Raphael Meyer, and Tyler Chen for helpful discussions and suggestions. This research was supported in part by NSF CAREER grants 2045590 and 1652257, ONR Award N00014-18-1-2364, and the Lifelong Learning Machines program from DARPA/MTO.

A PROOF OF FACT 3.3

Proof. We start by doing a change of variables; set $x = \cos\theta$ and note that $dx = -\sin\theta d\theta$. Substituting this into the expression for $\langle f, w \cdot \bar{T}_k \rangle$ and noting that $T_k(\cos\theta) = \cos k\theta$ gives us that

$$\sqrt{\frac{2}{\pi}} \int_{-1}^{1} f(x) \frac{T_k(x)}{\sqrt{1-x^2}} dx = \sqrt{\frac{2}{\pi}} \int_{-\pi}^{0} -f(\cos\theta)(\cos k\theta) d\theta$$

since $\sqrt{1-\cos^2\theta} = \sin\theta$ and $dx = -\sin\theta d\theta$. Integrating by parts and noting that $(f(\cos\theta) \int -\cos k\theta d\theta)|_{-\pi}^{0} = -f(\cos\theta)\frac{\sin k\theta}{k}|_{-\pi}^{0} = 0$ gives us that $\langle f, w \cdot \bar{T}_k \rangle = \sqrt{\frac{2}{\pi}} \int_{-\pi}^{0} \frac{\sin k\theta}{k}\, df(\cos\theta)$.

We use the definition of the Riemann-Stieltjes integral and let $M \in \mathbb{N}^+$ be a parameter and $\mathcal{P}_M = \{-\pi = x_0 \le \cdots \le x_M = 0\}$ be the set of all M intervals partitioning the interval $[-\pi, 0]$. Then for a partition $P \in \mathcal{P}_M$ we denote norm(P) to be the length of its longest sub-interval. The Riemann-Stieltjes integral $\int_{-\pi}^{0} \sin(k\theta)\, df(\cos\theta)$ can be written as

$$\lim_{\epsilon \to 0} \sup_{\substack{M,\, P \in \mathcal{P}_M \\ \text{s.t.norm}(P) \le \epsilon}} \sum_{i=0}^{m-1} (f(\cos x_{i+1}) - f(\cos x_i)) \sin k x_i.$$

Since $f(x) \in \text{lip}_1$ and $|\sin k\theta| \le 1$ we can bound the magnitude of the above summation as $\left|\sum_{i=0}^{m-1} (f(\cos x_{i+1}) - f(\cos x_i)) \sin k x_i\right| \le \sum_{i=0}^{m-1} \lambda |\cos x_{i+1} - \cos x_i| \le 2$.

The last inequality follows from the fact that $\cos(\theta)$ is 1-Lipschitz. Putting these bounds together gives us that $|\langle f, w \cdot \bar{T}_k \rangle| \le 2\lambda/k$. □

REFERENCES

[1] Jared L. Aurentz, Vassilis Kalantzis, and Yousef Saad. 2017. Cucheb: A GPU implementation of the filtered Lanczos procedure. *Computer Physics Communications* 220 (2017), 332 – 340.
[2] Haim Avron and Sivan Toledo. 2011. Randomized Algorithms for Estimating the Trace of an Implicit Symmetric Positive Semi-Definite Matrix. *J. ACM* 58, 2, Article 8 (2011).
[3] Jess Banks, Jorge Vargas, Archit Kulkarni, and Nikhil Srivastava. 2019. Pseudospectral Shattering, the Sign Function, and Diagonalization in Nearly Matrix Multiplication Time. In *Proceedings of the 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*.
[4] Mark Braverman, Elad Hazan, Max Simchowitz, and Blake Woodworth. 2020. The Gradient Complexity of Linear Regression. In *Proceedings of the 33rd Annual Conference on Computational Learning Theory (COLT)*, Vol. 125. 627–647.
[5] Vladimir Braverman, Aditya Krishnan, and Christopher Musco. 2021. Linear and sublinear time spectral density estimation. *arXiv preprint arXiv:2104.03461* (2021).
[6] Tyler Chen, Thomas Trogdon, and Shashanka Ubaru. 2021. Analysis of stochastic Lanczos quadrature for spectrum approximation. In *Proceedings of the International Congress of Mathematicians 2021 (ICM)*.
[7] C. W. Clenshaw. 1955. A note on the summation of Chebyshev series. *Math. Comp.* 9, 51 (1955), 118.
[8] David Cohen-Steiner, Weihao Kong, Christian Sohler, and Gregory Valiant. 2018. Approximating the Spectrum of a Graph. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 1263–1271.
[9] Timothy A Davis and Yifan Hu. 2011. The University of Florida sparse matrix collection. *ACM Transactions on Mathematical Software (TOMS)* 38, 1 (2011), 1–25.
[10] Prathamesh Dharangutte and Christopher Musco. 2021. Dynamic Trace Estimation. *Preprint* (2021).
[11] Kun Dong, Austin R Benson, and David Bindel. 2019. Network density of states. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 1152–1161.
[12] Petros Drineas, Ravi Kannan, and Michael W Mahoney. 2006. Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication. *SIAM J. Comput.* 36, 1 (2006), 132–157.
[13] Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. 2019. An Investigation into Neural Net Optimization via Hessian Eigenvalue Density. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, Vol. 97. 2232–2241.
[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.
[15] Michael F. Hutchinson. 1990. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics-Simulation and Computation* 19, 2 (1990), 433–450.
[16] Dunham Jackson. 1912. On approximation by trigonometric sums and polynomials. *Transactions of the American Mathematical society* 13, 4 (1912), 491–515.
[17] Dunham Jackson. 1930. *The Theory of Approximation*. Colloquium Publications, Vol. 11. American Mathematical Society.

[18] Leonid Vital'evich Kantorovich and Gennadii Shlemovich Rubinshtein. 1957. On a functional space and certain extremum problems. In *Doklady Akademii Nauk*, Vol. 115. Russian Academy of Sciences, 1058–1061.
[19] Liran Katzir, Edo Liberty, and Oren Somekh. 2011. Estimating Sizes of Social Networks via Biased Sampling. In *Proceedings of the 20th International World Wide Web Conference (WWW)*. 597–606.
[20] Weihao Kong and Gregory Valiant. 2017. Spectrum estimation from samples. *Ann. Statist.* 45, 5 (10 2017), 2218–2247.
[21] Ruipeng Li, Yuanzhe Xi, Lucas Erlandson, and Yousef Saad. 2019. The Eigenvalues Slicing Library (EVSL): Algorithms, Implementation, and Software. *SIAM Journal on Scientific Computing* 41, 4 (2019), C393–C415.
[22] Lin Lin, Yousef Saad, and Chao Yang. 2016. Approximating Spectral Densities of Large Matrices. *SIAM Rev.* 58, 1 (2016), 34–65.
[23] Michael Mahoney and Charles Martin. 2019. Traditional and Heavy Tailed Self Regularization in Neural Network Models. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.), Vol. 97. 4284–4293.
[24] Raphael A. Meyer, Cameron Musco, Christopher Musco, and David Woodruff. 2020. Hutch++: Optimal Stochastic Trace Estimation. *Proceedings of the 4th Symposium on Simplicity in Algorithms (SOSA)* (2020).
[25] Rajeev Motwani and Prabhakar Raghavan. 1995. *Randomized Algorithms*. Cambridge University Press. https://doi.org/10.1017/CBO9780511814075
[26] Cameron Musco and Christopher Musco. 2015. Randomized Block Krylov Methods for Stronger and Faster Approximate Singular Value Decomposition. In *Advances in Neural Information Processing Systems 28 (NeurIPS)*. 1396–1404.
[27] Cameron Musco, Christopher Musco, and Aaron Sidford. 2018. Stability of the Lanczos Method for Matrix Function Approximation. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 1605–1624.
[28] Cameron Musco, Praneeth Netrapalli, Aaron Sidford, Shashanka Ubaru, and David P. Woodruff. 2018. Spectrum Approximation Beyond Fast Matrix Multiplication: Algorithms and Hardness. *Proceedings of the 9th Conference on Innovations in Theoretical Computer Science (ITCS)* (2018).
[29] Beresford N. Parlett. 1998. *The symmetric eigenvalue problem*. SIAM.
[30] Barak A. Pearlmutter. 1994. Fast exact multiplication by the Hessian. *Neural computation* 6, 1 (1994), 147–160.
[31] Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. 2018. The emergence of spectral universality in deep networks. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS)*. 1924–1932.
[32] Farbod Roosta-Khorasani and Uri M. Ascher. 2015. Improved Bounds on Sample Size for Implicit Matrix Trace Estimators. *Foundations of Computational Mathematics* 15, 5 (2015), 1187–1212.
[33] Mark Rudelson and Roman Vershynin. 2013. Hanson-Wright inequality and sub-Gaussian concentration. *Electronic Communications in Probability* 18 (2013).
[34] R.N. Silver and H. R oder. 1994. Densities of States of Mega-Dimensional Hamiltonian Matrices. *International Journal of Modern Physics C* 5, 4 (1994), 735–753.
[35] Max Simchowitz, Ahmed El Alaoui, and Benjamin Recht. 2018. Tight query complexity lower bounds for PCA via finite sample deformed wigner law. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC)*. 1249–1259.
[36] John Skilling. 1989. *The Eigenvalues of Mega-dimensional Matrices*. Springer Netherlands, 455–466.
[37] Xiaoming Sun, David P. Woodruff, Guang Yang, and Jialin Zhang. 2019. Querying a Matrix Through Matrix-Vector Products. In *Proceedings of the 46th International Colloquium on Automata, Languages and Programming (ICALP)*, Vol. 132. 94:1–94:16.
[38] Lloyd N Trefethen. 2008. Is Gauss quadrature better than Clenshaw–Curtis? *SIAM review* 50, 1 (2008), 67–87.
[39] Lin-Wang Wang. 1994. Calculating the density of states and optical-absorption spectra of large quantum systems by the plane-wave moments method. *Phys. Rev. B* 49 (1994), 10154–10158. Issue 15.
[40] Weiran Wang and Miguel A Carreira-Perpinán. 2013. Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application. *arXiv preprint arXiv:1309.1541* (2013).
[41] Alexander Weiße, Gerhard Wellein, Andreas Alvermann, and Holger Fehske. 2006. The kernel polynomial method. *Reviews of modern physics* 78, 1 (2006), 275.
[42] David P. Woodruff. 2014. Sketching as a Tool for Numerical Linear Algebra. *Foundations and Trends in Theoretical Computer Science* 10, 1–2 (2014), 1–157.
[43] Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael W. Mahoney. 2020. PyHessian: Neural Networks Through the Lens of the Hessian. In *IEEE BigData*.