SIAM J. SCI. COMPUT.
Vol. 39, No. 4, pp. A1558–A1585

APPROXIMATING SPECTRAL SUMS OF LARGE-SCALE MATRICES USING STOCHASTIC CHEBYSHEV APPROXIMATIONS*

INSU HAN†, DMITRY MALIOUTOV‡, HAIM AVRON§, AND JINWOO SHIN¶

Abstract. Computation of the trace of a matrix function plays an important role in many scientific computing applications, including applications in machine learning, computational physics (e.g., lattice quantum chromodynamics), network analysis, and computational biology (e.g., protein folding), just to name a few application areas. We propose a linear-time randomized algorithm for approximating the trace of matrix functions of large symmetric matrices. Our algorithm is based on coupling function approximation using Chebyshev interpolation with stochastic trace estimators (Hutchinson's method), and as such requires only implicit access to the matrix, in the form of a function that maps a vector to the product of the matrix and the vector. We provide rigorous approximation error in terms of the extremal eigenvalue of the input matrix, and the Bernstein ellipse that corresponds to the function at hand. Based on our general scheme, we provide algorithms with provable guarantees for important matrix computations, including log-determinant, trace of matrix inverse, Estrada index, Schatten p-norm, and testing positive definiteness. We experimentally evaluate our algorithm and demonstrate its effectiveness on matrices with tens of millions dimensions.

Key words. spectral function, matrix computation, Chebyshev approximation, Hutchinson's method

AMS subject classification. 68W25

DOI. 10.1137/16M1078148

1. Introduction.

Given a symmetric matrix $A \in \mathbb{R}^{d\times d}$ and function $f : \mathbb{R} \to \mathbb{R}$, we study how to efficiently compute

$$\Sigma_f(A) = \mathtt{tr}(f(A)) = \sum_{i=1}^{d} f(\lambda_i), \tag{1}$$

where $\lambda_1, \ldots, \lambda_d$ are eigenvalues of A. We refer to such sums as *spectral sums* (also referred to as *trace functions*). Spectral sums depend only on the eigenvalues of A and so they are *spectral functions*, although not every spectral function is a spectral sum. Nevertheless, the class of spectral sums is rich and includes useful spectral functions. For example, if A is also positive definite then $\Sigma_{\log}(A) = \log\det(A)$, i.e. the log-determinant of A.

*Submitted to the journal's Methods and Algorithms for Scientific Computing section June 1, 2016; accepted for publication (in revised form) March 13, 2017; published electronically August 22, 2017. This article is partially based on preliminary results published in the proceeding of the 32nd International Conference on Machine Learning (ICML 2015).

http://www.siam.org/journals/sisc/39-4/M107814.html

Funding: The work of the third author was supported by the XDATA program of the Defense Advanced Research Projects Agency (DARPA), administered through Air Force Research Laboratory contract FA8750-12-C-0323.

†School of Electrical Engineering, Korea Advanced Institute of Science and Technology, Dacjeon Korea (hawki17@kaist.ac.kr).

‡Business Analytics and Mathematical Sciences, IBM Research, Yorktown Heights, NY, 10598 (dmalioutov@us.ibm.com).

§Department of Applied Mathematics, Tel Aviv University/Tel Aviv 6997801, Israel (haimav@post.tau.ac.il).

¶School of Electrical Engineering, Korea Advanced Institute of Science and Technology, Dacjeon Korea (jinwoos@kaist.ac.kr).

Indeed, there are many real-world applications in which spectral sums play an important role. For example, the log-determinant appears ubiquitously in machine learning applications including Gaussian graphical and Gaussian process models [38, 36, 13], partition functions of discrete graphical models [29], minimum-volume ellipsoids [44], metric learning, and kernel learning [10]. The trace of the matrix inverse ($\Sigma_f(A)$ for $f(x) = 1/x$) is frequently computed for the covariance matrix in uncertainty quantification [9, 27] and lattice quantum chromodynamics [39]. The Estrada index ($\Sigma_{\exp}(A)$) has been initially developed for topological index of protein folding in the study of protein functions and protein-ligand interactions [15, 12], and currently it appears in numerous other applications, e.g., statistical thermodynamics [18, 17], information theory [7], and network theory [19, 16] ; see Gutman et al. [22] for more applications. The Schatten p-norm ($\Sigma_f(A^\top A)^{1/p}$ for $f(x) = x^{p/2}$ for $p \geq 1$) has been applied to recover low-rank matrix [34] and sparse MRI reconstruction [30].

The computation of the aforementioned spectral sums for large-scale matrices is a challenging task. For example, the standard method for computing the log-determinant uses the Cholesky decomposition (if $A = LL^T$ is a Cholesky decomposition, then $\log\det(A) = 2\sum_i \log L_{ii}$). In general, the computational complexity of Cholesky decomposition is cubic with respect to the number of variables, i.e., $O(d^3)$. For large-scale applications involving more than tens of thousands of dimensions, this is obviously not feasible. If the matrix is sparse, one might try to take advantage of sparse decompositions. As long as the amount of fill-in during the factorizations is not too big, a substantial improvement in running time can be expected. Nevertheless, the worst case still requires $\Theta(d^3)$. In particular, if the sparsity structure of A is random-like, as is common in several of the aforementioned applications, then little improvement can be expected with sparse methods.

Our aim is to design an efficient algorithm that is able to compute accurate approximations to spectral sums for matrices with *tens of millions* of variables.

1.1. Contributions. We propose a randomized algorithm for approximating spectral sums based on a combination of stochastic trace-estimators and Chebyshev interpolation. Our algorithm first computes the coefficients of a Chebyshev approximation of f. This immediately leads to an approximation of the spectral sums as the trace of power series of the input matrix. We then use a stochastic trace-estimator to estimate this trace. In particular, we use *Hutchinson's method* [25].

One appealing aspect of Hutchinson's method is that it does not require an explicit representation of the input matrix; Hutchinson's method requires only an implicit representation of the matrix as an operation that maps a vector to the product of the matrix with the vector. In fact, this property is inherited by our algorithm to its entirety: our algorithm only needs access to an implicit representation of the matrix as an operation that maps a vector to the product of the matrix with the vector. In accordance, we measure the complexity of our algorithm in terms of the number of matrix-vector products that it requires. We establish rigorous bounds on the number of matrix-vector products for attaining a ε-multiplicative approximation of the spectral sums based on ε, the failure probability, and the range of the function over its Bernstein ellipse (see Theorem 3.1 for details). In particular, Theorem 3.1 implies that if the range is $\Theta(1)$, then the algorithm provides ε-multiplicative approximation guarantee using a constant amount of matrix-vector products for any constant $\varepsilon > 0$ and constant failure probability.

The overall time complexity of our algorithm is $O(t \cdot \|A\|_{\mathtt{mv}})$, where t is the number of matrix-vector products (as established by our analysis) and $\|A\|_{\mathtt{mv}}$ is the cost of

multiplying A by a vector. One overall assumption is that matrix-vector products can be computed efficiently, i.e., $\|A\|_{\mathtt{mv}}$ is small. For example, if A is sparse then $\|A\|_{\mathtt{mv}} = O(\mathtt{nnz}(A))$, i.e., the number of non-zero entries in A. Other cases that admit fast matrix-vector products are low-rank matrices (which allow fast multiplication by factorization), or Fourier (or Hadamard, Walsh, Toeplitz) matrices using the fast Fourier transform. The proposed algorithm is also very easy to parallelize.

We then proceed to discuss applications of the proposed algorithm. We give rigorous bounds for using our algorithm for approximating the log-determinant, trace of the inverse of a matrix, the Estrada index, and the Schatten p-norm. These correspond to continuous functions $f(x) = \log x$, $f(x) = 1/x$, $f(x) = \exp(x)$, and $f(x) = x^{p/2}$, respectively. We also use our algorithm to construct a novel algorithm for testing positive definiteness in the property testing framework. Our algorithm, which is based on approximating the spectral sums for $1 - \mathrm{sign}(x)$, is able to test positive definiteness of a matrix with a sublinear (in matrix size) number of matrix-vector products.

Our experiments show that our proposed algorithm is orders of magnitude faster than the standard methods for sparse matrices and provides approximations with less than 1% error for the examples we consider. It can also solve problems of tens of millions dimension in a few minutes on our single commodity computer with 32 GB memory. Furthermore, as reported in our experimental results, it achieves much better accuracy compared to a similar approach based on Talyor expansions [48], while both have similar running times. In addition, it outperforms the recent method based on Cauchy integral formula [1] in both running time and accuracy.[1] The proposed algorithm is also very easy to parallelize and hence has a potential to handle even larger problems. For example, the Schur method was used as a part of QUIC algorithm for sparse inverse covariance estimation with over a million variables [24], hence our log-determinant algorithm could be used to further improve its speed and scale.

1.2. Related Work. The first to consider the problem of approximating spectral sums was [3], and its specific use for approximating the log-determinant and the trace of the matrix inverse. Like our method, their method combines stochastic trace estimation with approximation of bilinear forms. However, their method for approximating bilinear forms is fundamentally different than our method and is based on a Gauss-type quadrature of a Riemann–Stieltjes integral. They do not provide rigorous bounds for the bilinear form approximation. In addition, recent progress on analyzing stochastic trace estimation [2, 37] allows us to provide rigorous bounds for the entire procedure.

Since then, several authors considered the use of stochastic trace estimators to compute certain spectral sums; [4, 31] consider the problem of computing the diagonal of a matrix or of the matrix inverse. Polynomial approximations and rational approximations of high-pass filter to count the number of eigenvalues in an input interval are used by [14]. They do not provide rigorous bounds. Stochastic approximations of score functions are used by [40] to learn large-scale Gaussian processes.

Approximation of the log-determinant in particular has received considerable treatment in the literature. Pace and LeSage [35] use both Taylor and Chebyshev based approximation to the logarithm function to design an algorithm for log-determinant approximation, but do not use stochastic trace estimation. Their method is determistic, can entertain only low-degree approximations, and has no rigorous bounds. Zhang and Leithead [48] consider the problem of approximating the

[1] Aune, Simpson, and Eidsvik's method [1] is implemented in the SHOGUN machine learning toolbox, http://www.shogun-toolbox.org.

log-determinant in the setting of Gaussian process parameter learning. They use Taylor expansion in conjunction with stochastic trace estimators, and propose novel error compensation methods. They do not provide rigorous bounds as we provide for our method. Boutsidis et al. [6] use a similar scheme based on Taylor expansion for approximating the log-determinant, and do provide rigorous bounds. Nevertheless, our experiments demonstrate that our Chebyshev interpolation based method provides superior accuracy. [1] approximates the log-determinant using a Cauchy integral formula. Their method requires the multiple use of a Krylov-subspace linear system solver, so their method is rather expensive. Furthermore, no rigorous bounds are provided.

Computation of the trace of the matrix inverse has also been researched extensively. One recent example is [46], which uses a combination of stochastic trace estimation and interpolating an approximate inverse. In another example, [8] considers how accurately linear systems should be solved when stochastic trace estimators are used to approximate the trace of the inverse.

To summarize, the main novelty of our work is combining Chebyshev interpolation with Hutchinson's trace estimator, which allows us to design a highly effective linear-time algorithm with rigorous approximation guarantees for general spectral sums.

1.3. Organization. The structure of the paper is as follows. We introduce the necessary background in section 2. Section 3 provides the description of our algorithm with approximation guarantees, and its applications to the log-determinant, the trace of matrix inverse, the Estrada index, the Schatten p-norm, and testing positive definiteness are described in section 4. We report experimental results in section 5.

2. Preliminaries. Throughout the paper, $A \in \mathbb{R}^{d\times d}$ is a symmetric matrix with eigenvalues $\lambda_1, \ldots, \lambda_d \in \mathbb{R}$ and I_d is the d-dimensional identity matrix. We use $\mathtt{tr}(\cdot)$ to denote the trace of the matrix. We denote the Schatten p-norm by $\|\cdot\|_{(p)}$, and the induced matrix p-norm by $\|\cdot\|_p$ (for $p = 1, 2, \infty$) . We also use $\lambda_{\min}(A)$ and $\lambda_{\max}(A)$ to denote the smallest and largest eigenvalue of A. In particular, we assume that an interval $[a, b]$ which contains all of A's eigenvalues is given. In some cases, such bounds are known a priori due to properties of the downstream use (e.g., the application considered in subsection 5.2). In others, a crude bound like $a = -\|A\|_\infty$ and $b = \|A\|_\infty$ or via Gershgorin's circle theorem [21, sect. 7.2] might be obtained. For some functions, our algorithm has additional requirements on a and b (e.g., for log-determinant, we need $a > 0$).

Our approach combines two techniques, which we discuss in detail in the next two subsections: (a) designing polynomial expansion for given function via Chebyshev interpolation [32] and (b) approximating the trace of matrix via Monte Carlo methods [25].

2.1. Function approximation using Chebyshev interpolation. Chebyshev interpolation approximates an analytic function by interpolating the function at the Chebyshev nodes using a polynomial. Conveniently, the interpolation can be expressed in terms of basis of Chebyshev polynomials. Specifically, the Chebyshev interpolation p_n of degree n for a given function $f : [-1, 1] \to \mathbb{R}$ is given by (see Mason and Handscomb [32]):

$$f(x) \approx p_n(x) = \sum_{j=0}^{n} c_j T_j(x), \tag{2}$$

where the coefficient c_j, the jth Chebyshev polynomial $T_j(x)$, and Chebyshev nodes $\{x_k\}_{k=0}^n$ are defined as

$$c_j = \begin{cases} \dfrac{1}{n+1}\displaystyle\sum_{k=0}^{n} f(x_k)\ T_0(x_k) & \text{if } \ j=0, \\ \dfrac{2}{n+1}\displaystyle\sum_{k=0}^{n} f(x_k)\ T_j(x_k) & \text{otherwise,} \end{cases} \tag{3}$$

$$\begin{aligned} T_0(x) &= 1, T_1(x) = x, \\ T_{j+1}(x) &= 2xT_j(x) - T_{j-1}(x) \qquad \text{for } \ j \geq 1, \\ x_k &= \cos\left(\frac{\pi(k+1/2)}{n+1}\right). \end{aligned} \tag{4}$$

Chebyshev interpolation better approximates the functions as the degree n increases. In particular, the following error bound is known [5, 47].

THEOREM 2.1. *Suppose f is analytic function with $|f(z)| \leq U$ in the region bounded by the so-called Bernstein ellipse with foci $+1, -1$ and sum of major and minor semi-axis lengths equal to $\rho > 1$. Let p_n denote the degree n Chebyshev interpolant of f as defined by* (2), (3), *and* (4). *We have*

$$\max_{x\in[-1,1]} |f(x) - p_n(x)| \leq \frac{4U}{(\rho-1)\,\rho^n}.$$

The interpolation scheme described so far assumed a domain of $[-1, 1]$. To allow a more general domain of $[a, b]$ one can use the linear mapping $g(x) = \frac{b-a}{2}x + \frac{b+a}{2}$ to map $[-1, 1]$ to $[a, b]$. Thus, $f \circ g$ is a function on $[-1, 1]$ which can be approximated using the scheme above. The approximation to f is then $\widetilde{p}_n = p_n \circ g^{-1}$, where p_n is the approximation to $f \circ g$. Note that $\widetilde{p}_n$ is a polynomial with degree n as well. In particular, we have the following approximation scheme for a general $f : [a, b] \to \mathbb{R}$:

$$f(x) \approx \widetilde{p}_n(x) = \sum_{j=0}^{n} \widetilde{c}_j T_j\left(\frac{2}{b-a}x - \frac{b+a}{b-a}\right), \tag{5}$$

where the coefficient $\widetilde{c}_j$ are defined as

$$\widetilde{c}_j = \begin{cases} \dfrac{1}{n+1}\displaystyle\sum_{k=0}^{n} f\left(\frac{b-a}{2}x_k + \frac{b+a}{2}\right)\ T_0(x_k) & \text{if } \ j=0, \\ \dfrac{2}{n+1}\displaystyle\sum_{k=0}^{n} f\left(\frac{b-a}{2}x_k + \frac{b+a}{2}\right)\ T_j(x_k) & \text{otherwise.} \end{cases} \tag{6}$$

The following is a simple corollary of Theorem 2.1.

COROLLARY 2.2. *Suppose that $a, b \in \mathbb{R}$ with $a < b$. Suppose f is an analytic function with $|f(\frac{b-a}{2}z + \frac{b+a}{2})| \leq U$ in the region bounded by the ellipse with foci $+1, -1$, and the sum of major and minor semi-axis lengths equals $\rho > 1$. Let $\widetilde{p}_n$ denote the degree n Chebyshev interpolant of f on $[a, b]$ as defined by* (4), (5) *and* (6). *We have*

$$\max_{x\in[a,b]} |f(x) - \widetilde{p}_n(x)| \leq \frac{4U}{(\rho-1)\,\rho^n}.$$

Proof. The proof follows immediately from Theorem 2.1 and observing that for $g(x) = \frac{b-a}{2}x + \frac{b+a}{2}$ we have

$$\max_{x\in[-1,1]} |(f \circ g)(x) - p_n(x)| = \max_{x\in[a,b]} |f(x) - \widetilde{p}_n(x)|. \qquad \square$$

Chebyshev interpolation for scalar functions can be naturally generalized to matrix functions [23]. Using the Chebyshev interpolation $\widetilde{p}_n$ for function f, we obtain the following approximation formula:

$$\begin{aligned}\Sigma_f(A) &= \sum_{i=1}^{d} f(\lambda_i) \approx \sum_{i=1}^{d} \widetilde{p}_n(\lambda_i) = \sum_{i=1}^{d}\sum_{j=0}^{n} \widetilde{c}_j T_j\left(\frac{2}{b-a}\lambda_i - \frac{b+a}{b-a}\right) \\ &= \sum_{j=0}^{n} \widetilde{c}_j \sum_{i=1}^{d} T_j\left(\frac{2}{b-a}\lambda_i - \frac{b+a}{b-a}\right) = \sum_{j=0}^{n} \widetilde{c}_j \mathtt{tr}\left(T_j\left(\frac{2}{b-a}A - \frac{b+a}{b-a}I_d\right)\right) \\ &= \mathtt{tr}\left(\sum_{j=0}^{n} \widetilde{c}_j T_j\left(\frac{2}{b-a}A - \frac{b+a}{b-a}I_d\right)\right),\end{aligned}$$

where the equality before the last follows from the fact that $\sum_{i=1}^{d} p(\lambda_i) = \mathtt{tr}(p(A))$ for any polynomial p, and the last equality from the linearity of the trace operation.

We remark that other polynomial approximations, e.g., Taylor, can also be used. However, it is known that Chebyshev interpolation, in addition to its simplicity, is nearly optimal [43] with respect to the ∞-norm is well-suited for our uses.

2.2. Stochastic trace estimation (Hutchinson's method). The main challenge in utilizing the approximation formula at the end of the last subsection is how to compute

$$\mathtt{tr}\left(\sum_{j=0}^{n} \widetilde{c}_j T_j\left(\frac{2}{b-a}A - \frac{b+a}{b-a}I_d\right)\right)$$

without actually computing the matrix involved (since the latter is expensive to compute). In this paper we turn to the stochastic trace estimation method. In essence, it is a Monte Carlo approach: to estimate the trace of an arbitrary matrix B, first a random vector $\mathbf{z}$ is drawn from some fixed distribution such that the expectation of $\mathbf{z}^\top B\mathbf{z}$ is equal to the trace of B. By sampling m such i.i.d. random vectors, and averaging we obtain an estimate of $\mathtt{tr}(B)$. Namely, given random vectors $\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(m)}$, the estimator is

$$\mathtt{tr}_m(B) = \frac{1}{m}\sum_{i=1}^{m} \mathbf{v}^{(i)\top} B \mathbf{v}^{(i)}.$$

Random vectors can be used for the above trace estimator as long as they have zero means and unit covariances [25]. Examples include those from Gaussian (normal) distribution and Rademacher distribution. The latter samples entries uniformly at random from $\{-1, +1\}$ which is known to have the smallest variance among such Monte Carlo methods [2]. This is called as the *Hutchinson estimator* and satisfies the following equalities:

$$\begin{aligned}\mathbf{E}\left[\mathtt{tr}_m(B)\right] &= \mathtt{tr}(B), \\ \mathbf{Var}\left[\mathtt{tr}_m(B)\right] &= \frac{2}{m}\left(\|B\|_F^2 - \sum_{i=1}^{d} B_{i,i}^2\right).\end{aligned}$$

However, (ε, ζ)-bounds, as introduced by [2], are more appropriate for our needs. Specifically, we use the following bound due to Roosta-Khorasani and Ascher [37].

THEOREM 2.3. *Let $B \in \mathbb{R}^{d\times d}$ be a positive (or negative) semi-definite matrix. Given $\varepsilon, \zeta \in (0,1)$,*

$$\Pr\left[|\mathtt{tr}_m(B) - \mathtt{tr}(B)| \le \varepsilon\,|\mathtt{tr}(B)|\right] \ge 1-\zeta$$

holds if sampling number m is larger than $6\varepsilon^{-2}\log(\frac{2}{\zeta})$.

Note that computing $\mathbf{v}^{(i)\top} B\mathbf{v}^{(i)}$ requires only multiplications between a matrix and a vector, which is particularly appealing when evaluating B itself is expensive, e.g.,

$$B = \sum_{j=0}^{n} \widetilde{c}_j T_j\left(\frac{2}{b-a}A - \frac{b+a}{b-a}I_d\right),$$

as in our case. In this case,

$$\mathbf{v}^{(i)\top} B\mathbf{v}^{(i)} = \sum_{j=0}^{n} \widetilde{c}_j \mathbf{v}^{(i)\top} T_j\left(\frac{2}{b-a}A - \frac{b+a}{b-a}I_d\right)\mathbf{v}^{(i)} = \sum_{j=0}^{n} \widetilde{c}_j \mathbf{v}^{(i)\top}\mathbf{w}_j^{(i)},$$

where

$$\mathbf{w}_j^{(i)} = T_j\left(\frac{2}{b-a}A - \frac{b+a}{b-a}I_d\right)\mathbf{v}^{(i)}.$$

The latter can be computed efficiently (using n matrix-vector products with A) by observing that due to (4) we have that

$$\mathbf{w}_0^{(i)} = \mathbf{v}^{(i)}, \mathbf{w}_1^{(i)} = \left(\frac{2}{b-a}A - \frac{b+a}{b-a}I_d\right)\mathbf{w}_0^{(i)},$$
$$\mathbf{w}_{j+1}^{(i)} = 2\left(\frac{2}{b-a}A - \frac{b+a}{b-a}I_d\right)\mathbf{w}_j^{(i)} - \mathbf{w}_{j-1}^{(i)}.$$

In order to apply Theorem 2.3 we need B to be positive (or negative) semi-definite. In our case $B = \widetilde{p}_n(A)$, and thus it is sufficient for $\widetilde{p}_n$ to be non-negative (non-positive) on $[a,b]$. The following lemma establishes a sufficient condition for non-negativity of $\widetilde{p}_n$, and a consequence positive (negative) semi-definiteness of $\widetilde{p}_n(A)$.

LEMMA 2.4. *Suppose f satisfies that $|f(x)| \ge L$ for $x \in [a,b]$. Then, linear transformed Chebyshev approximation $\widetilde{p}_n(x)$ of $f(x)$ is also non-negative on $[a,b]$ if*

$$\frac{4U}{(\rho-1)\,\rho^n} \le L \tag{7}$$

holds for all $n \ge 1$.

Proof. From Corollary 2.2, we have

$$\begin{aligned}\min_{[a,b]} \widetilde{p}_n(x) &= \min_{[a,b]} f(x) + (\widetilde{p}_n(x) - f(x)) \\ &\ge \min_{[a,b]} f(x) - \max_{[a,b]} |\widetilde{p}_n(x) - f(x)| \\ &\ge L - \frac{4U}{(\rho-1)\,\rho^n} \ge 0.\end{aligned}$$

This completes the proof of Lemma 2.4. ∎

Algorithm 1. Trace of matrix function f approximation.

Input: symmetric matrix $A \in \mathbb{R}^{d\times d}$ with eigenvalues in $[a,b]$, sampling number m and polynomial degree n
Initialize: $\Gamma \leftarrow 0$
for $j = 0$ **to** n **do**
  $\widetilde{c}_j \leftarrow j$th coefficient of the Chebyshev interpolation of f on $[a,b]$ (see equation 6)
end for
for $i = 1$ **to** m **do**
  Draw a random vector $\mathbf{v}^{(i)} \in \{-1,+1\}^d$ whose entries are uniformly distributed
  $\mathbf{w}_0^{(i)} \leftarrow \mathbf{v}^{(i)}$ and $\mathbf{w}_1^{(i)} \leftarrow \frac{2}{b-a}A\mathbf{v}^{(i)} - \frac{b+a}{b-a}\mathbf{v}^{(i)}$
  $\mathbf{u} \leftarrow \widetilde{c}_0\mathbf{w}_0^{(i)} + \widetilde{c}_1\mathbf{w}_1^{(i)}$
  for $j = 2$ **to** n **do**
    $\mathbf{w}_2^{(i)} \leftarrow \frac{4}{b-a}A\mathbf{w}_1^{(i)} - \frac{2(b+a)}{b-a}\mathbf{w}_1^{(i)} - \mathbf{w}_0^{(i)}$
    $\mathbf{u} \leftarrow \mathbf{u} + \widetilde{c}_j\ \mathbf{w}_2$
    $\mathbf{w}_0^{(i)} \leftarrow \mathbf{w}_1^{(i)}$ and $\mathbf{w}_1^{(i)} \leftarrow \mathbf{w}_2^{(i)}$
  end for
  $\Gamma \leftarrow \Gamma + \mathbf{v}^{(i)\top}\mathbf{u}/m$
end for
Output: Γ

3. Approximating spectral sums.

3.1. Algorithm description. Our algorithm brings together the components discussed in the previous section. A pseudo-code description appears as Algorithm 1. As mentioned before, we assume that eigenvalues of A are in the interval $[a,b]$ for some $b > a$.

In section 4, we provide five concrete applications of the above algorithm: approximating the log-determinant, the trace of matrix inverse, the Estrada index, the Schatten p-norm, and testing positive definiteness, which correspond to $\log x$, $1/x$, $\exp(x)$, $x^{p/2}$, and $1 - \operatorname{sign}(x)$, respectively.

3.2. Analysis. We establish the following theoretical guarantee on the proposed algorithm.

THEOREM 3.1. *Suppose function f satisfies the following:*

- *f is non-negative (or non-positive) on $[a,b]$.*
- *f is analytic with $|f(\frac{b-a}{2}z + \frac{b+a}{2})| \le U$ for some $U < \infty$ on the elliptic region E_ρ in the complex plane with foci at $-1,+1$ and ρ as the sum of semi-major and semi-minor lengths.*
- *$\min_{x\in[a,b]} |f(x)| \ge L$ for some $L > 0$.*

Given $\varepsilon, \zeta \in (0,1)$, if

$$m \ge 54\varepsilon^{-2}\log(2/\zeta),$$
$$n \ge \log\left(\frac{8}{\varepsilon(\rho-1)}\frac{U}{L}\right)/\log\rho,$$

then

$$\Pr\left(|\Sigma_f(A) - \Gamma| \leq \varepsilon\,|\Sigma_f(A)|\right) \geq 1 - \zeta,$$

where Γ *is the output of* Algorithm 1.

The number of matrix-vector products performed by Algorithm 1 is $O(mn)$, thus the time-complexity is $O(mn\|A\|_{\mathtt{mv}})$, where $\|A\|_{\mathtt{mv}}$ is that of the matrix-vector operation. In particular, if $m, n = O(1)$, the complexity is linear with respect to $\|A\|_{\mathtt{mv}}$. Therefore, Theorem 3.1 implies that if $U, L = \Theta(1)$, then one can choose $m, n = O(1)$ for ε-multiplicative approximation with probability of at least $1 - \zeta$ given constants $\varepsilon, \zeta > 0$.

Proof. The condition

$$n \geq \log\left(\frac{8}{\varepsilon(\rho - 1)}\frac{U}{L}\right) / \log\rho$$

implies that

$$\frac{4U}{(\rho - 1)\,\rho^n} \leq \frac{\varepsilon}{2}L\,. \tag{8}$$

Recall that the trace of a matrix is equal to the sum of its eigenvalues and that this also holds for a function of the matrix, i.e., $f(A)$. Under this observation, we establish a matrix version of Corollary 2.2. Let $\lambda_1, \ldots, \lambda_d \in [a, b]$ be the eigenvalues of A. We have

$$\begin{aligned}
|\Sigma_f(A) - \mathtt{tr}\left(\widetilde{p}_n(A)\right)| &= \left|\sum_{i=1}^{d} f(\lambda_i) - \widetilde{p}_n\left(\lambda_i\right)\right| \leq \sum_{i=1}^{d} |f(\lambda_i) - \widetilde{p}_n\left(\lambda_i\right)| \\
&\leq \sum_{i=1}^{d} \frac{4U}{(\rho - 1)\,\rho^n} = \frac{4dU}{(\rho - 1)\,\rho^n} && (9) \\
&\leq \frac{\varepsilon}{2}dL \leq \frac{\varepsilon}{2}d \min_{[a,b]} |f(x)| && (10) \\
&\leq \frac{\varepsilon}{2}\sum_{i=1}^{d} |f(\lambda_i)| = \frac{\varepsilon}{2}\,|\Sigma_f(A)|\,, && (11)
\end{aligned}$$

where the inequality (9) is due to Corollary 2.2, inequality (10) holds due to inequality (8), and the last equality is due to the fact that f is either non-negative or non-positive.

Moreover, the inequality of (11) shows

$$|\mathtt{tr}\left(\widetilde{p}_n(A)\right)| - |\Sigma_f(A)| \leq |\Sigma_f(A) - \mathtt{tr}\left(\widetilde{p}_n(A)\right)| \leq \frac{\varepsilon}{2}\,|\Sigma_f(A)|\,,$$

which implies for $\varepsilon \in (0, 1)$ that

$$|\mathtt{tr}\left(\widetilde{p}_n(A)\right)| \leq \left(\frac{\varepsilon}{2} + 1\right)|\Sigma_f(A)| \leq \frac{3}{2}\,|\Sigma_f(A)|\,. \tag{12}$$

A polynomial degree n that satisfies (8) also satisfies (7), and from this it follows that $\widetilde{p}_n(A)$ is a positive semi-definite matrix by Lemma 2.4. Hence, we can apply Theorem 2.3: for $m \geq 54\varepsilon^{-2}\log(2/\zeta)$ we have,

$$\Pr\left(|\mathtt{tr}\left(\widetilde{p}_n(A)\right) - \mathtt{tr}_m\left(\widetilde{p}_n(A)\right)| \leq \frac{\varepsilon}{3}\,|\mathtt{tr}\left(\widetilde{p}_n(A)\right)|\right) \geq 1 - \zeta\,.$$

In addition, this probability with (12) provides

$$\Pr\left(\left|\mathtt{tr}\left(\widetilde{p}_n(A)\right) - \mathtt{tr}_m\left(\widetilde{p}_n(A)\right)\right| \leq \frac{\varepsilon}{2}\left|\Sigma_f(A)\right|\right) \geq 1-\zeta. \tag{13}$$

Combining (11) with (13) we have

$$\begin{aligned}
1-\zeta &\leq \Pr\left(\left|\mathtt{tr}\left(\widetilde{p}_n(A)\right) - \mathtt{tr}_m\left(\widetilde{p}_n(A)\right)\right| \leq \frac{\varepsilon}{2}\left|\Sigma_f(A)\right|\right) \\
&\leq \Pr\Big(\left|\Sigma_f(A) - \mathtt{tr}\left(\widetilde{p}_n(A)\right)\right| + \left|\mathtt{tr}\left(\widetilde{p}_n(A)\right) - \mathtt{tr}_m\left(\widetilde{p}_n(A)\right)\right| \\
&\qquad\qquad \leq \frac{\varepsilon}{2}\left|\Sigma_f(A)\right| + \frac{\varepsilon}{2}\left|\mathtt{tr}\left(f(A)\right)\right|\Big) \\
&\leq \Pr\left(\left|\Sigma_f(A) - \mathtt{tr}_m\left(\widetilde{p}_n(A)\right)\right| \leq \varepsilon\left|\Sigma_f(A)\right|\right)
\end{aligned}$$

We complete the proof by observing that Algorithm 1 computes $\Gamma = \mathtt{tr}_m\left(\widetilde{p}_n(A)\right)$. ∎

4. Applications. In this section, we discuss several applications of Algorithm 1: approximating the log-determinant, trace of the matrix inverse, the Estrada index, the Schatten p-norm, and testing positive definiteness. Underlying these applications is executing Algorithm 1 with the following functions: $f(x) = \log x$ (for log-determinant), $f(x) = 1/x$ (for matrix inverse), $f(x) = \exp(x)$ (for the Estrada index), $f(x) = x^{p/2}$ (for the Schatten p-norm), and $f(x) = \frac{1}{2}\left(1 + \tanh\left(-\alpha x\right)\right)$, as a smooth approximation of $1 - \operatorname{sign}(x)$ (for testing positive definiteness).

4.1. Log-determinant of positive definite matrices. Since $\Sigma_{\log}(A) = \log\det A$ our algorithm can naturally be used to approximate the log-determinant. However, it is beneficial to observe that

$$\Sigma_{\log}(A) = \Sigma_{\log}(A/(a+b)) + d\log(a+b)$$

and use Algorithm 1 to approximate $\Sigma_{\log}(\overline{A})$ for $\overline{A} = A/(a+b)$. The reason we consider $\overline{A}$ instead of A as an input of Algorithm 1 is because all eigenvalues of $\overline{A}$ are strictly less than 1 and the constant $L > 0$ in Theorem 3.1 is guaranteed to exist for $\overline{A}$. The procedure is summarized in Algorithm 2. In the next subsection we generalize the algorithm for general non-singular matrices.

We note that Algorithm 2 requires us to know a positive lower bound $a > 0$ for the eigenvalues, which is in general harder to obtain than the upper bound b (e.g., one can choose $b = \|A\|_\infty$). In some special cases, the smallest eigenvalue of positive definite matrices are known, e.g., random matrices [42, 41] and diagonal-dominant matrices [20, 33]. Furthermore, it is sometimes explicitly given as a parameter in many machine learning log-determinant applications [45], e.g., $A = aI_d + B$ for some positive semi-definite matrix B, and this includes the application involving Gaussian Markov random fields (GMRF) in subsection 5.2.

Algorithm 2. Log-determinant approximation for positive definite matrices.

Input: positive definite matrix $A \in \mathbb{R}^{d\times d}$ with eigenvalues in $[a,b]$ for some $a, b > 0$, sampling number m and polynomial degree n

Initialize: $\overline{A} \leftarrow A/\left(a+b\right)$

$\Gamma \leftarrow$ Output of Algorithm 1 with inputs $\overline{A}, [\frac{a}{a+b}, \frac{b}{a+b}], m, n$ with $f(x) = \log x$

$\Gamma \leftarrow \Gamma + d\log\left(a+b\right)$

Output: Γ

We provide the following theoretical bound on the sampling number m and the polynomial degree n of Algorithm 2.

THEOREM 4.1. *Given $\varepsilon, \zeta \in (0,1)$, consider the following inputs for* Algorithm 2*:*

- $A \in \mathbb{R}^{d\times d}$ *is a positive definite matrix with eigenvalues in* $[a,b]$ *for* $a,b>0$.
- $m \geq 54\varepsilon^{-2}(\log(1+\frac{b}{a}))^2 \log(\frac{2}{\zeta})$.
- $n \geq \dfrac{\log\left(\frac{20}{\varepsilon}(\sqrt{\frac{2b}{a}+1}-1)\frac{\log(1+(b/a))\log(2+2(b/a))}{\log(1+(a/b))}\right)}{\log\left(\frac{\sqrt{2(b/a)+1}+1}{\sqrt{2(b/a)+1}-1}\right)} = O\left(\sqrt{\frac{b}{a}}\log\left(\frac{b}{\varepsilon a}\right)\right)$

Then, it follows that

$$\Pr[\ |\log\det A - \Gamma| \leq \varepsilon d] \geq 1-\zeta,$$

where Γ *is the output of* Algorithm 2.

Proof. The proof of Theorem 4.1 is straightforward using Theorem 3.1 with choice of upper bound U, lower bound L, and constant ρ for the function $\log x$. Denote $\delta = \frac{a}{a+b}$ and eigenvalues of $\overline{A}$ lie in the interval $[\delta, 1-\delta]$. We choose the ellipse region, denoted by E_ρ, in the complex plane with foci at $+1,-1$ and its semi-major axis length is $1/(1-\delta)$. Then,

$$\rho = \frac{1}{1-\delta} + \sqrt{\left(\frac{1}{1-\delta}\right)^2 - 1} = \frac{\sqrt{2-\delta}+\sqrt{\delta}}{\sqrt{2-\delta}-\sqrt{\delta}} > 1$$

and $\log(\frac{(1-2\delta)x+1}{2})$ is analytic on and inside E_ρ in the complex plane.

The upper bound U can be obtained as follows:

$$\begin{aligned}\max_{z\in E_\rho}\left|\log\left(\frac{(1-2\delta)\,z+1}{2}\right)\right| &\leq \max_{z\in E_\rho}\sqrt{\left(\log\left|\frac{(1-2\delta)\,z+1}{2}\right|\right)^2+\pi^2} \\ &= \sqrt{\left(\log\left|\frac{\delta}{2\,(1-\delta)}\right|\right)^2+\pi^2} \leq 5\log\left(\frac{2}{\delta}\right) := U,\end{aligned}$$

where the inequality in the first line holds because $|\log z| = |\log|z| + i\arg(z)| \leq \sqrt{(\log|z|)^2+\pi^2}$ for any $z\in\mathbb{C}$, and the equality in the second line holds by the maximum-modulus theorem. We also have the lower bound on $\log x$ in $[\delta, 1-\delta]$ as follows:

$$\min_{[\delta,1-\delta]}|\log x| = \log\left(\frac{1}{1-\delta}\right) := L.$$

With these constants, a simple calculation reveals that Theorem 3.1 implies that Algorithm 1 approximates $\left|\log\det\overline{A}\right|$ with $\varepsilon/\log(1/\delta)$-mulitipicative approximation. The additive error bound now follows by using the fact that $|\log\det\overline{A}| \leq d\log(1/\delta)$. ∎

The bound on polynomial degree n in the above theorem is relatively tight, e.g., $n=27$ for $\delta=0.1$ and $\varepsilon=0.01$. Our bound for m can yield very large numbers for the range of ε and ζ we are interested in. However, numerical experiments revealed that for the matrices we were interested in, the bound is not tight and $m\approx 50$ was sufficient for the accuracy levels we required in the experiments.

4.2. Log-determinant of non-singular matrices. One can apply the algorithm in the previous section to approximate the log-determinant of a non-symmetric

Algorithm 3. Log-determinant approximation for non-singular matrices.

Input: non-singular matrix $C \in \mathbb{R}^{d\times d}$ with singular values in $[\sigma_{\min}, \sigma_{\max}]$ for some $\sigma_{\min}, \sigma_{\max} > 0$, sampling number m and polynomial degree n
$\Gamma \leftarrow$ Output of Algorithm 2 for inputs $C^\top C, [\sigma_{\min}^2, \sigma_{\max}^2], m, n$
$\Gamma \leftarrow \Gamma/2$
Output: Γ

non-singular matrix $C \in \mathbb{R}^{d\times d}$. The idea is simple: run Algorithm 2 on the positive definite matrix $C^\top C$. The underlying observation is that

$$\log|\det C| = \frac{1}{2}\log\det C^\top C\,. \tag{14}$$

Without loss of generality, we assume that singular values of C are in the interval $[\sigma_{\min}, \sigma_{\max}]$ for some $\sigma_{\min}, \sigma_{\max} > 0$, i.e., the condition number $\kappa(C)$ is at most $\kappa_{\max} := \sigma_{\max}/\sigma_{\min}$. The proposed algorithm is not sensitive to tight knowledge of $\sigma_{\min}$ or $\sigma_{\max}$, but some loose lower and upper bounds on them, respectively, suffice. A pseudo-code description appears as Algorithm 3.

The time-complexity of Algorithm 3 is $O(mn\|C\|_{\mathtt{mv}}) = O(mn\|C^\top C\|_{\mathtt{mv}})$ as well since Algorithm 2 requires the computation of products of matrix $C^\top C$ and a vector, and that can be accomplished by first multiplying by C and then by $C^\top$. We state the following additive error bound of the above algorithm.

COROLLARY 4.2. *Given $\varepsilon, \zeta \in (0,1)$, consider the following inputs for* Algorithm 3*:*

- *$C \in \mathbb{R}^{d\times d}$ is a matrix with singular values in $[\sigma_{\min}, \sigma_{\max}]$ for some $\sigma_{\min}$, $\sigma_{\max} > 0$.*
- *$m \geq \mathcal{M}(\varepsilon, \frac{\sigma_{\max}}{\sigma_{\min}}, \zeta)$ and $n \geq \mathcal{N}(\varepsilon, \frac{\sigma_{\max}}{\sigma_{\min}})$, where*

$$\mathcal{M}(\varepsilon,\kappa,\zeta) := \frac{14}{\varepsilon^2}\left(\log\left(1+\kappa^2\right)\right)^2 \log\left(\frac{2}{\zeta}\right),$$

$$\mathcal{N}(\varepsilon,\kappa) := \frac{\log\left(\frac{10}{\varepsilon}\left(\sqrt{2\kappa^2+1}-1\right)\frac{\log(2+2\kappa^2)}{\log(1+\kappa^{-2})}\right)}{\log\left(\frac{\sqrt{2\kappa^2+1}+1}{\sqrt{2\kappa^2+1}-1}\right)} = O\left(\kappa\log\frac{\kappa}{\varepsilon}\right).$$

Then, it follows that

$$\Pr\left[\ |\log(|\det C|) - \Gamma| \leq \varepsilon d\ \right] \geq 1-\zeta,$$

where Γ is the output of Algorithm 3.

Proof. The proof follows immediately from (14) and Theorem 4.1, and observing that all the eigenvalues of $C^\top C$ are inside $[\sigma_{\min}^2, \sigma_{\max}^2]$. ∎

We remark that the condition number $\sigma_{\max}/\sigma_{\min}$ decides the complexity of Algorithm 3. As one can expect, the approximation quality and algorithm complexity become worse as the condition number increases, as polynomial approximation for log near the point 0 is challenging and requires higher polynomial degrees.

4.3. Trace of matrix inverse. In this section, we describe how to estimate the trace of matrix inverse. Since this task amounts to computing $\Sigma_f(A)$ for $f(x) = 1/x$, we propose Algorithm 4, which uses Algorithm 1 as a subroutine.

We provide the following theoretical bounds on sampling number m and polynomial degree n of Algorithm 4.

Algorithm 4. Trace of matrix inverse.

Input: positive definite matrix $A \in \mathbb{R}^{d\times d}$ with eigenvalues in $[a,b]$ for some $a,b>0$, sampling number m and polynomial degree n

$\Gamma \leftarrow$ Output of Algorithm 1 for inputs $A,[a,b],m,n$ with $f(x)=\frac{1}{x}$

Output: Γ

THEOREM 4.3. *Given $\varepsilon,\zeta\in(0,1)$, consider the following inputs for* Algorithm 4*:*

- $A\in\mathbb{R}^{d\times d}$ *is a positive definite matrix with eigenvalues in* $[a,b]$.
- $m \geq 54\varepsilon^{-2}\log\left(\frac{2}{\zeta}\right)$.
- $n \geq \log\left(\frac{8}{\varepsilon}\left(\sqrt{2\left(\frac{b}{a}\right)-1}-1\right)\frac{b}{a}\right)/\log\left(\frac{2}{\sqrt{2\left(\frac{b}{a}\right)-1}-1}+1\right) = O\left(\sqrt{\frac{b}{a}}\log\left(\frac{b}{\varepsilon a}\right)\right)$.

Then, it follows that

$$\Pr\left[\ \left|\mathtt{tr}\left(A^{-1}\right)-\Gamma\right| \leq \varepsilon\left|\mathtt{tr}\left(A^{-1}\right)\right|\right] \geq 1-\zeta,$$

where Γ *is the output of* Algorithm 4.

Proof. In order to apply Theorem 3.1, we define inverse function with linear transformation $\widetilde{f}$ as

$$\widetilde{f}(x) = \frac{1}{\frac{b-a}{2}x+\frac{b+a}{2}} \quad \text{for} \quad x\in[-1,1].$$

Avoiding singularities of $\widetilde{f}$, it is analytic on and inside the elliptic region in the complex plane passing through $\frac{b}{b-a}$ whose foci are $+1$ and -1. The sum of the length of semi-major and semi-minor axes is equal to

$$\rho = \frac{b}{b-a}+\sqrt{\frac{b^2}{(b-a)^2}-1} = \frac{2}{\sqrt{2\left(\frac{b}{a}\right)-1}-1}+1.$$

For the maximum absolute value on this region, $\widetilde{f}$ has maximum value $U=2/a$ at $-\frac{b}{b-a}$. The lower bound is $L=1/b$. Putting those together, Theorem 3.1, implies the bounds stated in the theorem statement. ∎

4.4. Estrada index. Given a (undirected) graph $G=(V,E)$, the Estrada index $\mathrm{EE}(G)$ is defined as

$$\mathrm{EE}(G) := \Sigma_{\exp}(A_G) = \sum_{i=1}^{d}\exp(\lambda_i),$$

where A_G is the adjacency matrix of G and $\lambda_1,\ldots,\lambda_{|V|}$ are the eigenvalues of A_G. It is a well-known result in spectral graph theory that the eigenvalues of A_G are contained in $[-\Delta_G,\Delta_G]$, where Δ_G is maximum degree of a vertex in G. Thus, the Estrada index G can be computed using Algorithm 1 with the choice of $f(x)=\exp(x)$, $a=-\Delta_G$, and $b=\Delta_G$. However, we state our algorithm and theoretical bounds in terms of a general interval $[a,b]$ that bounds the eigenvalues of A_G, to allow for an a priori tighter bound on the eigenvalues (note, however, that it is well known that always $\lambda_{\max}\geq\sqrt{\Delta_G}$).

We provide the following theoretical bounds on sampling number m and polynomial degree n of Algorithm 5.

Algorithm 5. Estrada index approximation.

Input: adjacency matrix $A_G \in \mathbb{R}^{d\times d}$ with eigenvalues in $[a,b]$, sampling number m and polynomial degree n

{If Δ_G is the maximum degree of G, then $a = -\Delta_G, b = \Delta_G$ can be used as default.}

$\Gamma \leftarrow$ Output of Algorithm 1 for inputs $A, [a,b], m, n$ with $f(x) = \exp(x)$

Output: Γ

THEOREM 4.4. *Given $\varepsilon, \zeta \in (0,1)$, consider the following inputs for* Algorithm 5*:*

- $A_G \in \mathbb{R}^{d\times d}$ *is an adjacency matrix of a graph with eigenvalues in* $[a,b]$.
- $m \geq 54\varepsilon^{-2}\log\left(\frac{2}{\zeta}\right)$.
- $n \geq \log\left(\frac{2}{\pi\varepsilon}(b-a)\exp\left(\frac{\sqrt{16\pi^2+(b-a)^2}+(b-a)}{2}\right)\right)/\log\left(\frac{4\pi}{b-a}+1\right) = O\left(\frac{b-a+\log\frac{1}{\varepsilon}}{\log\left(\frac{1}{b-a}\right)}\right)$.

Then, it follows that

$$\Pr\left[\ |\mathrm{EE}\,(G) - \Gamma| \leq \varepsilon\,|\mathrm{EE}\,(G)|\right] \geq 1-\zeta,$$

where Γ *is the output of* Algorithm 5.

Proof. We consider exponential function with linear transformation as

$$\widetilde{f}\,(x) = \exp\left(\frac{b-a}{2}x + \frac{b+a}{2}\right) \quad \text{for} \quad x \in [-1,1].$$

The function $\widetilde{f}$ is analytic on and inside the elliptic region in the complex plane which has foci ± 1 and passes through $\frac{4\pi i}{(b-a)}$. The sum of length of the semi-major and semi-minor axes becomes

$$\frac{4\pi}{b-a} + \sqrt{\frac{16\pi^2}{(b-a)^2}+1},$$

and we may choose ρ as $\frac{4\pi}{(b-a)}+1$.

By the maximum-modulus theorem, the absolute value of $\widetilde{f}$ on this elliptic region is maximized at $\sqrt{\frac{16\pi^2}{(b-a)^2}+1}$ with value $U = \exp(\frac{\sqrt{16\pi^2+(b-a)^2}+(b+a)}{2})$ and the lower bound has the value $L = \exp(a)$. Putting those all together in Theorem 3.1, we could obtain above the bound for approximation polynomial degree. This completes the proof of Theorem 4.4. ∎

4.5. Schatten *p*-norm. The Schatten p-norm for $p \geq 1$ of a matrix $M \in \mathbb{R}^{d_1\times d_2}$ is defined as

$$\|M\|_{(p)} = \left(\sum_{i=1}^{\min\{d_1,d_2\}} \sigma_i^p\right)^{1/p},$$

where σ_i is the ith singular value of M for $1 \leq i \leq \min\{d_1,d_2\}$. Schatten p-norm is widely used in linear algebric applications such as nuclear norm (also known as the trace norm) for $p = 1$:

$$\|M\|_{(1)} = \mathtt{tr}\left(\sqrt{M^\top M}\right) = \sum_{i=1}^{\min\{d_1,d_2\}} \sigma_i.$$

Algorithm 6. Schatten p-norm approximation.

Input: matrix $M \in \mathbb{R}^{d_1 \times d_2}$ with singular values in $[\sigma_{\min}, \sigma_{\max}]$, sampling number m and polynomial degree n

$\Gamma \leftarrow$ Output of Algorithm 1 for inputs $M^\top M$, $\left[\sigma_{\min}^2, \sigma_{\max}^2\right], m, n$ with $f(x) = x^{p/2}$

$\Gamma \leftarrow \Gamma^{1/p}$

Output: Γ

The Schatten p-norm corresponds to the spectral function $x^{p/2}$ of matrix $M^\top M$ since singular values of M are square roots of eigenvalues of $M^\top M$. In this section, we assume that general (possibly, non-symmetric) non-singular matrix $M \in \mathbb{R}^{d_1 \times d_2}$ has singular values in the interval $[\sigma_{\min}, \sigma_{\max}]$ for some $\sigma_{\min}, \sigma_{\max} > 0$, and propose Algorithm 6, which uses Algorithm 1 as a subroutine.

We provide the following theoretical bounds on sampling number m and polynomial degree n of Algorithm 6.

THEOREM 4.5. *Given $\varepsilon, \zeta \in (0,1)$, consider the following inputs for* Algorithm 6*:*

- $M \in \mathbb{R}^{d_1 \times d_2}$ *is a matrix with singular values in* $[\sigma_{\min}, \sigma_{\max}]$.
- $m \geq 54\varepsilon^{-2} \log\left(\frac{2}{\zeta}\right)$.
- $n \geq \mathcal{N}(\varepsilon, p, \frac{\sigma_{\max}}{\sigma_{\min}})$, *where*

$$\begin{aligned}\mathcal{N}(\varepsilon, p, \kappa) &:= \log\left(\frac{16(\kappa-1)}{\varepsilon}\left(\kappa^2+1\right)^{p/2}\right) \Big/ \log\left(\frac{\kappa+1}{\kappa-1}\right) \\ &= O\left(\kappa\left(p\log\kappa + \log\frac{1}{\varepsilon}\right)\right).\end{aligned}$$

Then, it follows that

$$\Pr\left[\left|\|M\|_{(p)}^p - \Gamma^p\right| \leq \varepsilon \|M\|_{(p)}^p\right] \geq 1-\zeta,$$

where Γ *is the output of* Algorithm 6.

Proof. Consider the following function as

$$\widetilde{f}(x) = \left(\frac{\sigma_{\max}^2 - \sigma_{\min}^2}{2}x + \frac{\sigma_{\max}^2 + \sigma_{\min}^2}{2}\right)^{p/2} \quad \text{for} \quad x \in [-1,1].$$

In general, $x^{p/2}$ for arbitrary $p \geq 1$ is defined on $x \geq 0$. We choose elliptic region E_ρ in the complex plane such that it is passing through $-(\sigma_{\max}^2 + \sigma_{\min}^2)/(\sigma_{\max}^2 - \sigma_{\min}^2)$ and having foci $+1, -1$ on real axis so that $\widetilde{f}$ is analytic on and inside E_ρ. The length of the semi-axes can be computed as

$$\rho = \frac{\sigma_{\max}^2 + \sigma_{\min}^2}{\sigma_{\max}^2 - \sigma_{\min}^2} + \sqrt{\left(\frac{\sigma_{\max}^2 + \sigma_{\min}^2}{\sigma_{\max}^2 - \sigma_{\min}^2}\right)^2 - 1} = \frac{\sigma_{\max} + \sigma_{\min}}{\sigma_{\max} - \sigma_{\min}} = \frac{\kappa_{\max}+1}{\kappa_{\max}-1},$$

where $\kappa_{\max} = \sigma_{\max}/\sigma_{\min}$.

The maximum absolute value is occurring at $(\sigma_{\max}^2 + \sigma_{\min}^2)/(\sigma_{\max}^2 - \sigma_{\min}^2)$ and its value is $U = (\sigma_{\max}^2 + \sigma_{\min}^2)^{p/2}$. Also, the lower bound is obtained as $L = {\sigma_{\min}}^p$. Applying Theorem 3.1 together with choices of ρ, U, and L, the bound of degree for polynomial approximation n can be achieved. This completes the proof of Theorem 4.5. ∎

4.6. Testing positive definiteness. In this section we consider the problem of determining if a given symmetric matrix $A \in \mathbb{R}^{d\times d}$ is positive definite. This can be useful in several scenarios. For example, when solving a linear system $Ax = b$, the determination of whether A is positive definite can drive algorithmic choices like whether to use Cholesky decomposition or use LU decomposition, or alternatively, if an iterative method is preferred, whether to use CG or MINRES. In another example, checking if the Hessian is positive or negative definite can help determine if a critical point is a local maximum/minimum or a saddle point.

In general, positive definiteness can be tested in $O(d^3)$ operations by attempting a Cholesky decomposition of the matrix. If the operation succeeds then the matrix is positive definite, and if it fails (i.e., a negative diagonal is encountered) the matrix is indefinite. If the matrix is sparse, running time can be improved as long as the fill-in during the sparse Cholesky factorization is not too big, but in general the worst case is still $\Theta(d^3)$. More in line with this paper is to consider the matrix implicit, that is, accessible only via matrix-vector products. In this case, one can reduce the matrix to tridiagonal form by doing n iterations of Lanczos, and then test positive definiteness of the reduced matrix. This requires d matrix-vector multiplications, and thus running time $\Theta(\|A\|_{\mathtt{mv}} \cdot d)$. However, we note that this algorithm is not a practical algorithm since it suffers from severe numerical instability.

In this paper we consider testing positive definiteness under the property testing framework. Property testing algorithms relax the requirements of decision problems by allowing them to issue arbitrary answers for inputs that are on the boundary of the class. That is, for decision problem on a class L (in this case, the set of positive definite matrices) the algorithm is required to accept x with high probability if $x \in L$, and reject x if $x \notin L$ and x is ε-far from any $y \in L$. For x's that are not in L but are less than ε far away, the algorithm is free to return any answer. We say that such x's are in the *indifference region*. In this section we show that testing positive definiteness in the property testing framework can be accomplished using $o(d)$ matrix-vector products.

Using the spectral norm of a matrix to measure distance, this suggests the following property testing variant of determining if a matrix is positive definite.

PROBLEM 1. *Given a symmetric matrix $A \in \mathbb{R}^{d\times d}$, $\varepsilon > 0$, and $\zeta \in (0,1)$,*
- *If A is positive definite, accept the input with probability of at least $1-\zeta$.*
- *If $\lambda_{\min} \le -\varepsilon\|A\|_2$, reject the input with probability of at least $1-\zeta$.*

For ease of presentation, it will be more convenient to restrict the norm of A to be at most 1, and for the indifference region to be symmetric around 0.

PROBLEM 2. *Given a symmetric $A \in \mathbb{R}^{n\times n}$ with $\|A\|_2 \le 1$, $\varepsilon > 0$, and $\zeta \in (0,1)$,*
- *If $\lambda_{\min} \ge \varepsilon/2$, accept the input with probability of at least $1-\zeta$.*
- *If $\lambda_{\min} \le -\varepsilon/2$, reject the input with probability of at least $1-\zeta$.*

It is quite easy to translate an instance of Problem 1 to an instance of Problem 2. First we use power-iteration to approximate $\|A\|_2$. Specifically, we use enough power iterations with a normally distributed random initial vector to find a λ' such that $|\lambda' - \|A\|_2| \le (\varepsilon/2)\|A\|_2$ with probability at least $1-\zeta/2$. Due to a bound by Klien and Lu [28, sect. 4.4] we need to perform

$$\left\lceil \frac{2}{\varepsilon}\left(\log^2(2d) + \log\left(\frac{8}{\varepsilon\zeta^2}\right)\right)\right\rceil$$

iterations (matrix-vector products) to find such an λ'. Let $\lambda = \lambda'/(1-\varepsilon/2)$ and consider

$$B = \frac{A - \frac{\lambda\varepsilon}{2} I_d}{(1+\frac{\varepsilon}{2})\lambda}.$$

It is easy to verify that $\|B\|_2 \le 1$ and $\lambda/\|A\|_2 \ge 1/2$ for $\varepsilon > 0$. If $\lambda_{\min}(A) \in [0, \varepsilon\|A\|_2]$ then $\lambda_{\min}(B) \in [-\varepsilon'/2, \varepsilon'/2]$, where $\varepsilon' = \varepsilon/(1+\varepsilon/2)$. Therefore, by solving Problem 2 on B with ε' and $\zeta' = \zeta/2$ we have a solution to Problem 1 with ε and ζ.

We call the region $[-1, -\varepsilon/2] \cup [\varepsilon/2, 1]$ the *active region* $\mathcal{A}_\varepsilon$, and the interval $[-\varepsilon/2, \varepsilon/2]$ as the *indifference region* $\mathcal{I}_\varepsilon$.

Let S be the reverse-step function, that is,

$$S(x) = \begin{cases} 1 & \text{if } x \le 0, \\ 0 & \text{if } x > 0. \end{cases}$$

Now note that a matrix $A \in \mathbb{R}^{d\times d}$ is positive definite if and only if

$$\Sigma_S(A) \le \gamma \tag{15}$$

for any fixed $\gamma \in (0,1)$. This already suggests using Algorithm 1 to test positive definite; however, the discontinuity of S at 0 poses problems.

To circumvent this issue we use a two-stage approximation. First, we approximate the reverse-step function using a smooth function f (based on the hyperbolic tangent), and then use Algorithm 1 to approximate $\Sigma_f(A)$. By carefully controlling the transition in f, the degree in the polynomial approximation and the quality of the trace estimation, we guarantee that as long as the smallest eigenvalue is not in the indifference region, Algorithm 1 will return less than 1/4 with high probability if A is positive definite and will return more than 1/4 with high probability if A is not positive definite. The procedure is summarized as Algorithm 7.

The correctness of the algorithm is established in the following theorem. While we use Algorithm 1, the indifference region requires a more careful analysis so the proof does not rely on Theorem 3.1.

THEOREM 4.6. *Given $\varepsilon, \zeta \in (0,1)$, consider the following inputs for* Algorithm 7*:*

- *$A \in \mathbb{R}^{d\times d}$ be a symmetric matrix with eigenvalues in $[-1,1]$ and $\lambda_{\min}(A) \notin \mathcal{I}_\varepsilon$, where $\lambda_{\min}(A)$ is the minimum eigenvalue of A.*

Algorithm 7. Testing positive definiteness.

Input: symmetric matrix $A \in \mathbb{R}^{d\times d}$ with eigenvalues in $[-1,1]$, sampling number m and polynomial degree n
Choose $\varepsilon > 0$ as the distance of active region
$\Gamma \leftarrow$ Output of Algorithm 1 for inputs $A, [-1,1], m, n$ with $f(x) = \frac{1}{2}(1 + \tanh(-\frac{\log(16d)}{\varepsilon}x))$
if $\Gamma < \frac{1}{4}$ **then**
  return PD
else
  return NOT PD
end if

- $m \geq 24 \log\left(\frac{2}{\zeta}\right)$.
- $n \geq \frac{\log(32\sqrt{2}\log(16d))+\log(1/\varepsilon)-\log(\pi/8d)}{\log(1+\frac{\pi\varepsilon}{4\log(16d)})} = O\left(\frac{\log^2(d)+\log(d)\log(1/\varepsilon)}{\varepsilon}\right)$.

Then the answer returned by Algorithm 7 *is correct with probability of at least* $1-\zeta$.

The number of matrix-vector products in Algorithm 7 is $O((\frac{\log^2(d)+\log(d)\log(1/\varepsilon)}{\varepsilon})\log(1/\zeta))$ as compared with $O(d)$ that are required with non-property testing previous methods.

Proof. Let p_n be the degree Chebyshev interpolation of f. We begin by showing that

$$\max_{x\in\mathcal{A}_\varepsilon} |S(x)-p_n(x)| \leq \frac{1}{8d}.$$

To see this, we first observe that

$$\max_{x\in\mathcal{A}_\varepsilon} |S(x)-p_n(x)| \leq \max_{x\in\mathcal{A}_\varepsilon} |S(x)-f(x)| + \max_{x\in\mathcal{A}_\varepsilon} |f(x)-p_n(x)|,$$

and thus it is enough to bound each term by $1/16d$.

For the first term, let

$$\alpha = \frac{1}{\varepsilon}\log(16d) \tag{16}$$

and note that $f(x) = \frac{1}{2}(1+\tanh(-\alpha x))$. We have

$$\begin{aligned}\max_{x\in\mathcal{A}_\varepsilon} |S(x)-f(x)| &= \frac{1}{2}\max_{x\in[\varepsilon/2,1]} |1-\tanh(\alpha x)| \\ &= \frac{1}{2}\left(1-\tanh\left(\frac{\alpha\varepsilon}{2}\right)\right) \\ &= \frac{e^{-\alpha\varepsilon}}{1+e^{-\alpha\varepsilon}} \\ &\leq e^{-\alpha\varepsilon} = \frac{1}{16d}.\end{aligned}$$

To bound the second term we use Corollary 2.2. To that end we need to define an appropriate ellipse. Let E_ρ be the ellipse with foci $-1,+1$ passing through $\frac{i\pi}{4\alpha}$. The sum of semi-major and semi-minor axes is equal to

$$\rho = \frac{\pi+\sqrt{\pi^2+16\alpha^2}}{4\alpha}.$$

The poles of tanh are of the form $i\pi/2 \pm ik\pi$ so f is analytic inside E_ρ. It is always the case that $|\tanh(z)| \leq 1$ if $\Im(z) \leq \pi/4$ [2], so $|f(z)| \leq 1$ for $z \in E_\rho$. Applying Corollary 2.2 and noticing that $\rho \geq 1+\pi/4\alpha$, we have

$$\max_{x\in[-1,1]} |p_n(x)-f(x)| \leq \frac{4}{(\rho-1)\rho^d} \leq \frac{16\alpha}{\pi(1+\frac{\pi}{4\alpha})^d}.$$

[2] To see this, note that using simple algebraic manipulations it is possible to show that $|\tanh(z)| = (e^{2\Re(z)}+e^{2\Re(z)}-2\cos(2\Im(z)))/(e^{2\Re(z)}+e^{2\Re(z)}-2\cos(2\Im(z)))$, from which the bound easily follows.

Thus, $\max_{x\in[-1,1]} |p_n(x) - f(x)| \leq \frac{1}{16d}$ provided that

$$n \geq \frac{\log(32\alpha) - \log(\pi/8d)}{\log(1 + \frac{\pi}{4\alpha})},$$

which is exactly the lower bound on n in the theorem statement.

Let

$$B = p_n(A) + \frac{1}{8d} I_d;$$

then B is symmetric positive semi-definite since $p_n(x) \geq -1/8d$ due to the fact that $|f(x)| \geq 0$ for every x. According to Theorem 2.3,

$$\Pr\left(|\mathtt{tr}_m(B) - \mathtt{tr}(B)| \leq \frac{\mathtt{tr}(B)}{2}\right) \geq 1 - \zeta$$

if $m \geq 24\log(2/\zeta)$ as assumed in the theorem statement.

Since $\mathtt{tr}_m(B) = \mathtt{tr}_m(p_n(A)) + 1/8$, $\mathtt{tr}(B) = \mathtt{tr}(p_n(A)) + 1/8$, and $\Gamma = \mathtt{tr}_m(p_n(A))$, we have

$$\Pr\left(|\Gamma - \mathtt{tr}(p_n(A))| \leq \frac{\mathtt{tr}(p_n(A))}{2} + \frac{1}{16}\right) \geq 1 - \zeta\,. \tag{17}$$

If $\lambda_{\min}(A) \geq \varepsilon/2$, then all eigenvalues of $S(A)$ are zero and so all eigenvalues of $p_n(A)$ are bounded by $1/8d$, and thus $\mathtt{tr}(p_n(A)) \leq 1/8$. Inequality (17) then implies that

$$\Pr(\Gamma \leq 1/4) \geq 1 - \zeta\,.$$

If $\lambda_{\min}(A) \leq -\varepsilon/2$, $S(A)$ has at least one eigenvalue that is 1 and is mapped in $p_n(A)$ to at least $1 - 1/8d \geq 7/8$. All other eigenvalues in $p_n(A)$ are at the very least $-1/8d$, and thus $\mathtt{tr}(p_n(A)) \geq 3/4$. Inequality (17) then implies that

$$\Pr(\Gamma \geq 1/4) \geq 1 - \zeta\,.$$

The conditions $\lambda_{\min}(A) \geq \varepsilon/2$ and $\lambda_{\min}(A) \leq -\varepsilon/2$ together cover all cases for $\lambda_{\min}(A) \notin \mathcal{I}_\varepsilon$ thereby completing the proof. ∎

5. Experiments. The experiments were performed using a machine with 3.5GHz Intel i7-5930K processor with 12 cores and 32 GB RAM. We choose $m = 50$, $n = 25$ in our algorithm unless stated otherwise.

5.1. Log-determinant. In this section, we report the performance of our algorithm compared to other methods for computing the log-determinant of positive definite matrices. We first investigate the empirical performance of the proposed algorithm on large sparse random matrices. We generate a random matrix $A \in \mathbb{R}^{d\times d}$, where the number of non-zero entries per each row is around 10. We first select non-zero off-diagonal entries in each row with values drawn from the standard normal distribution. To make the matrix symmetric, we set the entries in transposed positions to the same values. Finally, to guarantee positive definiteness, we set its diagonal entries to absolute row-sums and add a small margin value 0.1. Thus, the lower bound for eigenvalues can be chosen as $a = 0.1$ and the upper bound is set to the infinite norm of a matrix.

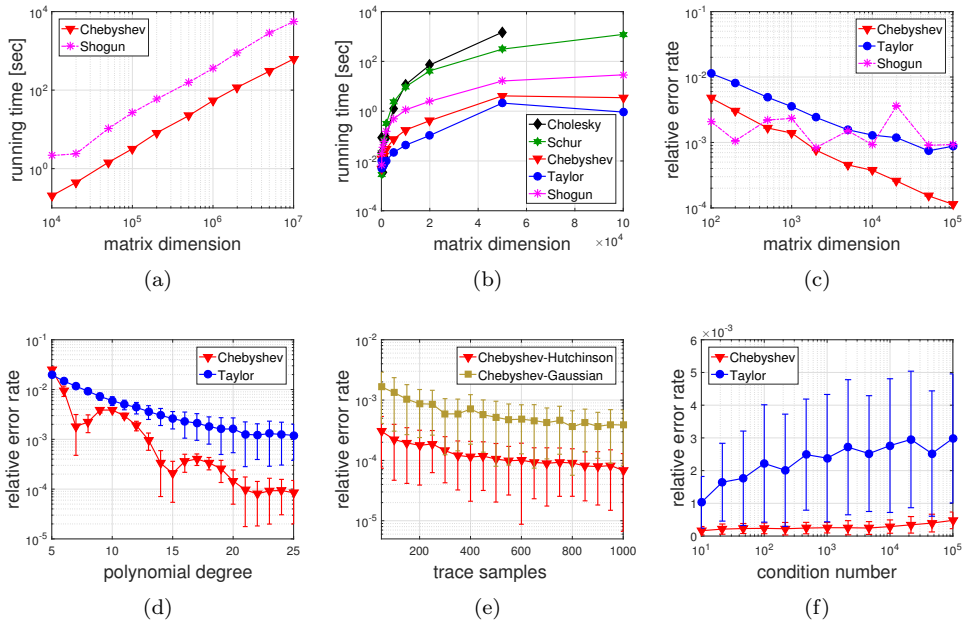
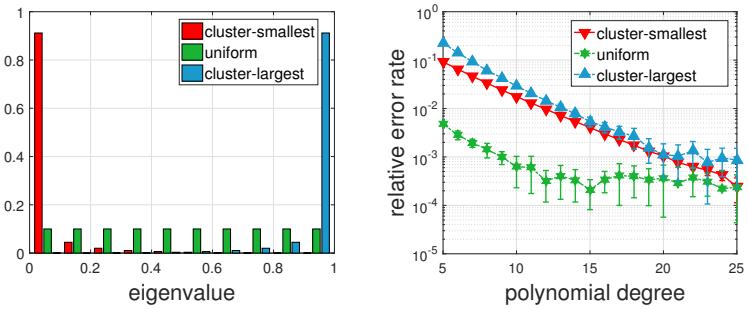


FIG. 1. *Performance evaluations of* Algorithm 2 *(i.e., Chebyshev) and comparisons with other algorithms:* (a) *running time varying matrix dimension;* (b) *comparison in running time among Cholesky decomposition, Schur complement* [24]*, Cauchy integral formula* [1]*, and Taylor-based algorithm* [48]*;* (c) *relative error varying matrix dimension;* (d) *relative error varying polynomial degree;* (e) *relative error varying the number of trace samples;* (f) *relative error varying condition number. The relative error means a ratio between the absolute error of the output of an approximation algorithm and the actual value of log-determinant.*

Figure 1 (a) shows the running time of Algorithm 2 from matrix dimension $d = 10^4$ to 10^7. The algorithm scales roughly linearly over a large range of matrix sizes, as expected. In particular, it takes only 600 seconds for a matrix of dimension 10^7 with 10^8 non-zero entries. Under the same setup, we also compare the running time of our algorithm with other ones, including Cholesky decomposition and Schur complement. The latter was used for sparse inverse covariance estimation with over a million variables [24] and we run the code implemented by those authors. The running time of the algorithms are reported in Figure 1 (b). Our algorithm is dramatically faster than both exact methods. Moreover, our algorithm is an order of magnitude faster than the recent approach based on the Cauchy integral formula [1], while it achieves better accuracy as reported in Figure 1 (c).[3]

We also compare the relative accuracies between our algorithm and that using Taylor expansions [48] with the same sampling number $m = 50$ and polynomial degree $n = 25$, as reported in Figure 1 (c). We see that the Chebyshev interpolation based method is more accurate than the one based on Taylor approximations. To complete the picture, we also use a large number of samples for trace estimator, $m = 1000$, for both algorithms to focus on the polynomial approximation errors. The results are reported in Figure 1 (d), showing that our algorithm using Chebyshev expansions is superior in accuracy compared to the Taylor-based algorithm.

[3]The method [1] is implemented in the SHOGUN machine learning toolbox, http://www.shogun-toolbox.org.



(a) (b)

FIG. 2. *Performance evaluations of* Algorithm 2 *when eigenvalue distributions are uniform (star), clustered on the smallest one (down triangle), and clustered on the largest one (up triangle):* (a) *distribution of eigenvalues,* (b) *relative error varying polynomial degree.*

In Figure 1 (e), we compare two different trace estimators, Gaussian and Hutchinson, under the choice of polynomial degree $n = 100$. We see that the Hutchinson estimator outperforms the Gaussian estimator. Finally, in Figure 1 (f) we report the results of experiments with varying condition number. We see that the Taylor-based method is more sensitive to the condition number than the Chebyshev-based method.

Chebyshev expansions have extreme points more likely around the end points of the approximating interval since the absolute values of their derivatives are larger. Hence, one can expect that if eigenvalues are clustered on the smallest (or largest) one, the quality of approximation becomes worse. To see this, we run Algorithm 2 for matrices having uniformly distributed eigenvalues and eigenvalues clustered on the smallest (or largest) one, which is reported in Figure 2. We observe that if the polynomial degree is small, the clustering effect causes larger errors, but the error decaying rate with respect to polynomial degree is not sensitive to it.

5.2. Maximum likelihood estimation for GMRF using log-determinant. In this section, we apply our proposed algorithm approximating log-determinants for maximum likelihood (ML) estimation in Gaussian Markov random fields (GMRF) [38]. GMRF is a multivariate joint Gaussian distribution defined with respect to a graph. Each node of the graph corresponds to a random variable in the Gaussian distribution, where the graph captures the conditional independence relationships (Markov properties) among the random variables. The model has been extensively used in many applications in computer vision, spatial statistics, and other fields. The inverse covariance matrix J (also called information or precision matrix) is positive definite and sparse: J_{ij} is non-zero only if the edge $\{i, j\}$ is contained in the graph. We are specifically interested in the problem of parameter estimation from data (fully or partially observed samples from the GMRF), where we would like to find the maximum likelihood estimates of the non-zero entries of the information matrix.

GMRF with 100 million variables on synthetic data. We first consider a GMRF on a square grid of size 5000×5000 with precision matrix $J \in \mathbb{R}^{d \times d}$ with $d = 25 \times 10^6$, which is parameterized by η, i.e., each node has four neighbors with

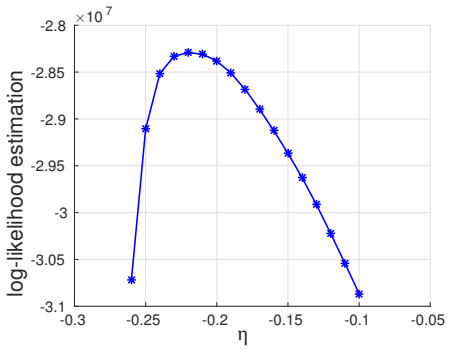


FIG. 3. *Log-likelihood estimation for hidden parameter η for square GMRF model of size* 5000×5000.

partial correlation η. We generate a sample $\mathbf{x}$ from the GMRF model (using a Gibbs sampler) for parameter $\eta = -0.22$. The log-likelihood of the sample is

$$\log p(\mathbf{x}|\eta) = \frac{1}{2}\log\det J(\eta) - \frac{1}{2}\mathbf{x}^\top J(\eta)\mathbf{x} - \frac{d}{2}\log(2\pi),$$

where $J(\eta)$ is a matrix of dimension 25×10^6 and 10^8 non-zero entries. Hence, the ML estimation requires us to solve

$$\max_\eta \left(\frac{1}{2}\log\det J(\eta) - \frac{1}{2}\mathbf{x}^\top J(\eta)\mathbf{x} - \frac{d}{2}\log(2\pi)\right).$$

We use Algorithm 2 to estimate the log-likelihood as a function of η, as reported in Figure 3. This confirms that the estimated log-likelihood is maximized at the correct (hidden) value $\eta = -0.22$.

GMRF with 6 million variables for ozone data. We also consider a similar GMRF parameter estimation from real spatial data with missing values. We use the data-set from [1] that provides satellite measurements of ozone levels over the entire earth following the satellite tracks. We use a resolution of 0.1 degrees in latitude and longitude, giving a spatial field of size 1681×3601, with over 6 million variables. The data-set includes 172,000 measurements. To estimate the log-likelihood in the presence of missing values, we use the Schur complement for determinants. Let the precision matrix for the entire field be $J = \left(\begin{smallmatrix} J_o & J_{o,z} \\ J_{z,o} & J_z \end{smallmatrix}\right)$, where subsets $\mathbf{x}_o$ and $\mathbf{x}_z$ denote the observed and unobserved components of $\mathbf{x}$. Then, our goal is to find some parameter η such that

$$\max_\eta \int_{\mathbf{x}_z} p(\mathbf{x}_o, \mathbf{x}_z|\eta)\, d\mathbf{x}_z.$$

We estimate the marginal probability using the fact that the marginal precision matrix of $\mathbf{x}_o$ is $\bar{J}_o = J_o - J_{o,z}J_z^{-1}J_{z,o}$ and its log-determinant is computed as $\log\det(\bar{J}_o) = \log\det(J) - \log\det(J_z)$ via Schur complements. To evaluate the quadratic term $x_o'\bar{J}_o x_o$ of the log-likelihood we need a single linear solve using an iterative solver. We use a linear combination of the thin-plate model and the thin-membrane models [38], with two parameters $\eta = (\alpha, \beta)$: $J = \alpha I + \beta J_{tp} + (1-\beta)J_{tm}$ and obtain ML estimates using Algorithm 2. Note that smallest eigenvalue of J is equal to α. We show the sparse measurements in Figure 4 (a) and the GMRF interpolation using fitted values of parameters in Figure 4 (b). We can see that the proposed log-determinant estimation

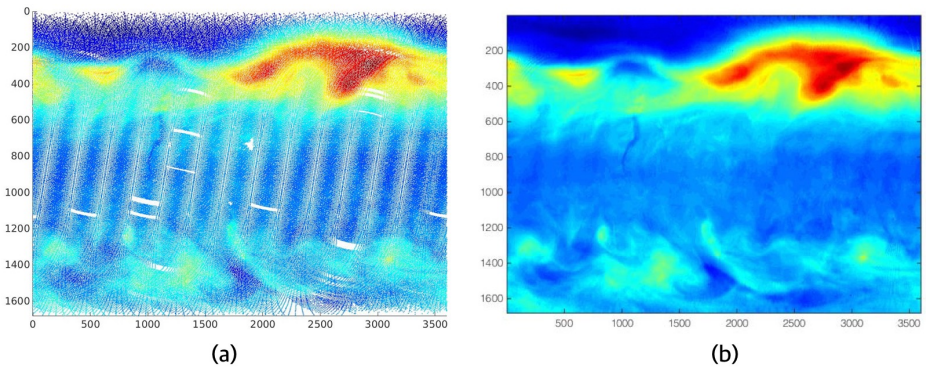


FIG. 4. *GMRF interpolation of ozone measurements:* (a) *original sparse measurements and* (b) *interpolated values using a GMRF with parameters fitted using* Algorithm 2.

algorithm allows us to do efficient estimation and inference in GMRFs of very large size, with sparse information matrices of size over 6 millions variables.

5.3. Other spectral functions. In this section, we report the performance of our scheme for four other choices of function f: the trace of matrix inverse, the Estrada index, the matrix nuclear norm, and testing positive definiteness, which correspond to $f(x) = 1/x$, $f(x) = \exp(x)$, $f(x) = x^{1/2}$, and $f(x) = \frac{1}{2}(1 + \tanh(-\alpha x))$, respectively. The detailed algorithm description for each function is given in section 4. Since the running times of our algorithms are "almost" independent of the choice of function f, i.e., it is the same as the case $f(x) = \log x$ reported in the previous section, we focus on measuring the accuracy of our algorithm.

In Figure 5, we report the approximation error of our algorithm for the trace of matrix inverse, the Estrada index, the matrix nuclear norm, and testing positive definiteness. All experiments were conducted on random 5000-by-5000 matrices. The particular setups for the different matrix functions are:

- The input matrix for the trace of matrix inverse is generated in the same way with the log-determinant case in the previous section.
- For the Estrada index, we generate the random regular graphs with 5000 vertices and degree $\Delta_G = 10$.
- For the nuclear norm, we generate random nonsymmetric matrices and estimate its nuclear norm (which is equal to the sum of all singular values). We first select the 10 positions of non-zero entries in each row and their values are drawn from the standard normal distribution. The reason why we consider nonsymmetric matrices is because the nuclear norm of a symmetric matrix is much easier to compute, e.g., the nuclear norm of a positive definite matrix is just its trace. We choose $\sigma_{\min} = 10^{-4}$ and $\sigma_{\max} = \sqrt{\|A\|_1 \|A\|_\infty}$ for input matrix A.
- For testing positive definiteness, we first create random symmetric matrices whose smallest eigenvalue varies from 10^{-1} to 10^{-4} and the largest eigenvalue is less than 1 (via appropriate normalizations). Namely, the condition number is between 10 and 10^4. We choose the same sampling number $m = 50$ and three different polynomial degrees: $n = 200, 1800$, and 16000. For each degree n, Algorithm 7 detects correctly positive definiteness of matrices with condition numbers at most 10^2, 10^3, and 10^4, respectively. The error rate is measured as a ratio of incorrect results among 20 random instances.

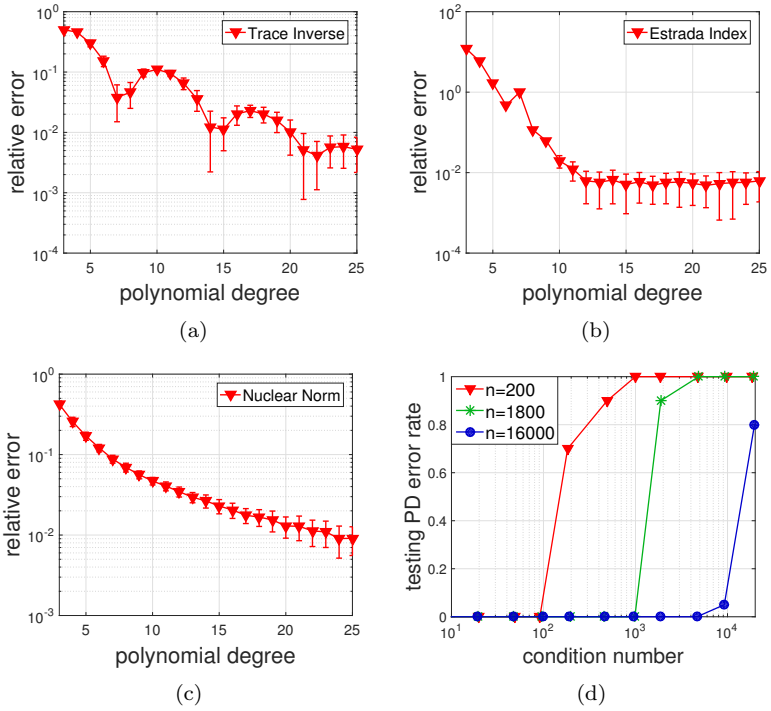


FIG. 5. *Accuracy of the proposed algorithm:* (a) *the trace of matrix inverse,* (b) *the Estrada index,* (c) *the nuclear norm (Schatten* 1*-norm) and* (d) *testing positive definiteness.*

For experiments of the trace of matrix inverse, the Estrada index, and the nuclear norm, we plot the relative error of the proposed algorithms varying polynomial degrees in Figure 5 (a), (b), and (c), respectively. Each of them achieves less than 1% error with polynomial degree at most $n = 25$ and sampling number $m = 50$. Figure 5 (d) shows the results of testing positive definiteness. When n is set according to the condition number the proposed algorithm is almost always correct in detecting positive definiteness. For example, if the decision problem involves the active region $\mathcal{A}_\varepsilon$ for $\varepsilon = 0.02$, which is the case that matrices having the condition number at most 100, polynomial degree $n = 200$ is enough for the correct decision.

We tested the proposed algorithm for testing positive definiteness on real-world matrices from the University of Florida Sparse Matrix Collection [11], selecting various symmetric matrices. We use $m = 50$ and three choices for n: $n = 200, 1800, 16000$. The results are reported in Table 1. We observe that the algorithm is always correct when declaring positive definiteness, but seems to declare indefiniteness when the matrix is too ill-conditioned for it to detect definiteness correctly. In addition, with two exceptions (`crankseg_1` and `pwtk`), when $n = 16000$ the algorithm was correct in declaring whether the matrix is positive definite or not. We remark that while $n = 16000$ is rather large it is still smaller than the dimension of most of the matrices that were tested (recall that our goal was to develop an algorithm that requires a small number of matrix products, i.e., it does not grow with respect to the matrix dimension). We also note that even when the algorithm fails it still provides useful information about both positive definiteness and the condition number of an input matrix, while standard methods such as Cholesky decomposition (as mentioned in

TABLE 1

Testing positive definiteness for real-world matrices. Algorithm 7 *outputs PD or NOT PD, i.e., the input matrix is either* (1) *positive definite (PD) or* (2) *not positive definite or its smallest eigenvalue is in the indifference region (NOT PD). The MATLAB* `eigs` *and* `condest` *functions output the smallest eigenvalue and an estimate for the condition number of the input matrix, respectively.*

matrix	dimension	number of nonzeros	positive definite	Algorithm 7 $n = 200$	Algorithm 7 $n = 1800$	Algorithm 7 $n = 16000$	MATLAB `eigs`	MATLAB `condest`
Chem97ZtZ	2,541	7,361	yes	PD	PD	PD	diverge	462.6
fv1	9,604	85,264	yes	PD	PD	PD	0.5122	12.76
fv2	9,801	87,025	yes	PD	PD	PD	0.5120	12.76
fv3	9,801	87,025	yes	NOT PD	NOT PD	PD	0.0020	4420
CurlCurl_0	11,083	113,343	no	NOT PD	NOT PD	NOT PD	diverge	6.2×10^{21}
barth5	15,606	107,362	no	NOT PD	NOT PD	NOT PD	-2.1066	84292
Dubcova1	16,129	253,009	yes	NOT PD	NOT PD	PD	0.0048	2624
cvxqp3	17,500	114,962	no	NOT PD	NOT PD	NOT PD	diverge	2.2×10^{16}
bodyy4	17,546	121,550	yes	NOT PD	NOT PD	PD	diverge	1017
t3dl_e	20,360	20360	yes	NOT PD	NOT PD	PD	diverge	6031
bcsstm36	23,052	320,060	no	NOT PD	NOT PD	NOT PD	diverge	∞
crystm03	24,696	583,770	yes	NOT PD	PD	PD	3.7×10^{-15}	467.7
aug2d	29,008	76,832	no	NOT PD	NOT PD	NOT PD	-2.8281	∞
wathen100	30,401	471,601	yes	NOT PD	NOT PD	PD	0.0636	8247
aug3dcqp	35,543	128,115	no	NOT PD	NOT PD	NOT PD	diverge	4.9×10^{15}
wathen120	36,441	565,761	yes	NOT PD	NOT PD	PD	0.1433	4055
bcsstk39	46,772	2,060,662	no	NOT PD	NOT PD	NOT PD	diverge	3.1×10^{8}
crankseg_1	52,804	10,614,210	yes	NOT PD	NOT PD	NOT PD	diverge	2.2×10^{8}
blockqp1	60,012	640,033	no	NOT PD	NOT PD	NOT PD	-446.636	8.0×10^{5}
Dubcova2	65,025	1,030,225	yes	NOT PD	NOT PD	PD	0.0012	10411
thermomech_TC	102,158	711,558	yes	NOT PD	PD	PD	0.0005	125.5
Dubcova3	146,689	3,636,643	yes	NOT PD	NOT PD	PD	0.0012	11482
thermomech_dM	204,316	1,423,116	yes	NOT PD	PD	PD	9.1×10^{-7}	125.487
pwtk	217,918	11,524,432	yes	NOT PD	NOT PD	NOT PD	diverge	5.0×10^{12}
bmw3_2	227,362	11,288,630	no	NOT PD	NOT PD	NOT PD	diverge	1.2×10^{20}

subsection 4.6) are intractable for large matrices. Furthermore, one can first run an algorithm to estimate the condition number, e.g., the MATLAB `condest` function, and then choose an appropriate degree n. We also run the MATLAB `eigs` function, which is able to estimate the smallest eigenvalue using iterative methods [26] (hence, it can be used for testing positive definitenesss). Unfortunately, the iterative method often does not converge, i.e., residual tolerance may not go to zero, as reported in Table 1. One advantage of our algorithm is that it does not depend on a convergence criterion.

6. Conclusion. Recent years have a seen a surge in the need for various computations on large-scale unstructured matrices. The lack of structure poses a significant challenge for traditional decomposition based methods. Randomized methods are a natural candidate for such tasks as they are mostly oblivious to structure. In this paper, we proposed and analyzed a linear-time approximation algorithm for spectral sums of symmetric matrices, where the exact computation requires cubic-time in the worst case. Furthermore, our algorithm is very easy to parallelize since it requires only (separable) matrix-vector multiplications. We believe that the proposed algorithm will find important theoretical and computational roles in a variety of applications ranging from statistics and machine learning to applied science and engineering.

Acknowledgments. The authors thank Peder Oslen and Sivan Toledo for helpful discussions.

REFERENCES

[1] E. Aune, D. Simpson, and J. Eidsvik, *Parameter estimation in high dimensional Gaussian distributions*, Stat. Comput., 24 (2014), pp. 247–263.
[2] H. Avron and S. Toledo, *Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix*, J. ACM, 58 (2011), p. 8.
[3] Z. Bai, G. Fahey, and G. Golub, *Some large-scale matrix computation problems*, J. Comput. Appl. Math., 74 (1996), pp. 71–89, https://doi.org/10.1016/0377-0427(96)00018-0, http://www.sciencedirect.com/science/article/pii/0377042796000180.
[4] C. Bekas, E. Kokiopoulou, and Y. Saad, *An estimator for the diagonal of a matrix*, Appl. Numer. Math., 57 (2007), pp. 1214–1229.
[5] J. P. Berrut and L. N. Trefethen, *Barycentric Lagrange interpolation*, SIAM Rev., 46 (2004), pp. 501–517.
[6] C. Boutsidis, P. Drineas, P. Kambadur, and A. Zouzias, *A Randomized Algorithm for Approximating the Log Determinant of a Symmetric Positive Definite Matrix*, preprint arXiv:1503.00374, 2015.
[7] R. Carbó-Dorca, *Smooth function topological structure descriptors based on graph-spectra*, J. Math. Chem., 44 (2008), pp. 373–378.
[8] J. Chen, *How accurately should I compute implicit matrix-vector products when applying the Hutchinson trace estimator?*, SIAM J. Sci. Comput., 38 (2016), pp. A3515–A3539, https://doi.org/10.1137/15M1051506.
[9] M. Dashti and A. M. Stuart, *Uncertainty quantification and weak approximation of an elliptic inverse problem*, SIAM J. Numer. Anal., 49 (2011), pp. 2524–2542.
[10] J. Davis, B. Kulis, P. Jain, S. Sra, and I. Dhillon, *Information-theoretic metric learning*, in Proceedings of the 24th International Conference on Machine Learning, Corvallis, OR, 2007.
[11] T. A. Davis and Y. Hu, *The University of Florida sparse matrix collection*, ACM Trans. Math., Software (TOMS), 38 (2011), pp. 1–25, http://www.cise.ufl.edu/research/sparse/matrices.
[12] J. A. de la Peña, I. Gutman, and J. Rada, *Estimating the Estrada index*, Linear Algebra Appl., 427 (2007), pp. 70–76.
[13] A. P. Dempster, *Covariance selection*, Biometrics, (1972), pp. 157–175.
[14] E. Di Napoli, E. Polizzi, and Y. Saad, *Efficient estimation of eigenvalue counts in an interval*, Numer. Linear Algebra Appl., 23 (2016), pp. 674–692.

[15] E. Estrada, *Characterization of 3D molecular structure*, Chemical Physics Letters, 319 (2000), pp. 713–718.
[16] E. Estrada, *Topological structural classes of complex networks*, Phys. Rev. E, 75 (2007), p. 016103.
[17] E. Estrada, *Atom–bond connectivity and the energetic of branched alkanes*, Chemical Physics Letters, 463 (2008), pp. 422–425.
[18] E. Estrada and N. Hatano, *Statistical-mechanical approach to subgraph centrality in complex networks*, Chemical Physics Letters, 439 (2007), pp. 247–251.
[19] E. Estrada and J. A. Rodríguez-Velázquez, *Spectral measures of bipartivity in complex networks*, Phys. Rev. E, 72 (2005), p. 046105.
[20] S. A. Gershgorin, *Uber die abgrenzung der eigenwerte einer matrix*, Izvestiya or Russian Academy of Sciences, (1931), pp. 749–754.
[21] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Vol. 3, JHU Press, 2012.
[22] I. Gutman, H. Deng, and S. Radenković, *The Estrada index: An updated survey*, Selected Topics on Appl. Graph Spectra, Math. Inst., Beograd, (2011), pp. 155–174.
[23] N. Higham, *Functions of Matrices*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2008, https://doi.org/10.1137/1.9780898717778.
[24] C. Hsieh, M. A. Sustik, I. S. Dhillon, P. K. Ravikumar, and R. Poldrack, *BIG & QUIC: Sparse inverse covariance estimation for a million variables*, in Adv. Neural Inf. Process. Syst., 26 (2013), pp. 3165–3173.
[25] M. Hutchinson, *A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines*, Commun. Statistics-Simulation Comput., 19 (1990), pp. 433–450.
[26] I. C. Ipsen, *Computing an eigenvector with inverse iteration*, SIAM Rev., 39 (1997), pp. 254–291.
[27] V. Kalantzis, C. Bekas, A. Curioni, and E. Gallopoulos, *Accelerating data uncertainty quantification by solving linear systems with multiple right-hand sides*, Numer., Algorithms, 62 (2013), pp. 637–653.
[28] P. Klein and H.-I. Lu, *Efficient approximation algorithms for semidefinite programs arising from max cut and coloring*, in Proceedings of the 28th Annual ACM Symposium on Theory of Computing, Philadelphia, PA, 1996, pp. 338–347, https://doi.org/10.1145/237814.237980.
[29] J. Ma, J. Peng, S. Wang, and J. Xu, *Estimating the partition function of graphical models using Langevin importance sampling*, in Proceedings of the 16th International Conference on Artificial Intelligence and Statistics, Scottsdale, AZ 2013, pp. 433–441.
[30] A. Majumdar and R. K. Ward, *An algorithm for sparse MRI reconstruction by Schatten p-norm minimization*, Magnetic Resonance Imaging, 29 (2011), pp. 408–417.
[31] D. M. Malioutov, J. K. Johnson, and A. Willsky, *Low-rank variance estimation in large-scale GMRF models*, in IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Vol. 3, Toulouse, FR 2006, pp. III–III.
[32] J. C. Mason and D. C. Handscomb, *Chebyshev Polynomials*, CRC Press, Boca Raton, FL, 2002.
[33] N. Morača, *Bounds for norms of the matrix inverse and the smallest singular value*, Linear Algebra Appl., 429 (2008), pp. 2589–2601.
[34] F. Nie, H. Huang, and C. Ding, *Low-rank matrix recovery via efficient Schatten p-norm minimization*, in Proceedings of the 26th AAAI Conference on Artificial Intelligence, Toronto, ON, 2012, pp. 655–661, http://dl.acm.org/citation.cfm?id=2900728.2900822.
[35] R. K. Pace and J. P. LeSage, *Chebyshev approximation of log-determinants of spatial weight matrices*, Comput. Statist. Data Anal., 45 (2004), pp. 179–196.
[36] C. E. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*, MIT Press, Cambridge, MA, 2005.
[37] F. Roosta-Khorasani and U. M. Ascher, *Improved bounds on sample size for implicit matrix trace estimators*, Found. Comput. Math., 15 (2015), pp. 1187–1212, https://doi.org/10.1007/s10208-014-9220-1.
[38] H. Rue and L. Held, *Gaussian Markov Random Fields: Theory and Applications*, CRC Press, Boca Raton, FL, 2005.
[39] A. Stathopoulos, J. Laeuchli, and K. Orginos, *Hierarchical probing for estimating the trace of the matrix inverse on toroidal lattices*, SIAM J. Sci. Comput., 35 (2013), pp. S299–S322.
[40] M. L. Stein, J. Chen, and M. Anitescu, *Stochastic approximation of score functions for Gaussian processes*, Annals Appl. Statist., 7 (2013), pp. 1162–1191.
[41] T. Tao and V. Vu, *Random matrices: The distribution of the smallest singular values*, Geom. Funct. Anal., 20 (2010), pp. 260–297.
[42] T. Tao and V. H. Vu, *Inverse Littlewood-Offord theorems and the condition number of random discrete matrices*, Ann. Math., 169 (2009), pp. 595–632.

[43] L. N. Trefethen, *Approximation Theory and Approximation Practice*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2012.

[44] S. Van Aelst and P. Rousseeuw, *Minimum volume ellipsoid*, Wiley Interdisciplinary Reviews: Computational Statistics, 1 (2009), pp. 71–82.

[45] M. J. Wainwright and M. I. Jordan, *Log-determinant relaxation for approximate inference in discrete Markov random fields*, IEEE Trans. Signal Process., 54 (2006), pp. 2099–2109.

[46] L. Wu, J. Laeuchli, V. Kalantzis, A. Stathopoulos, and E. Gallopoulos, *Estimating the trace of the matrix inverse by interpolating from the diagonal of an approximate inverse*, J. Comput. Phys., 326 (2016), pp. 828–844, https://doi.org/10.1016/j.jcp.2016.09.001, http://www.sciencedirect.com/science/article/pii/S0021999116304120.

[47] S. Xiang, X. Chen, and H. Wang, *Error bounds for approximation in Chebyshev points*, Numer. Math., 116 (2010), pp. 463–491.

[48] Y. Zhang and W. E. Leithead, *Approximate implementation of the logarithm of the matrix determinant in Gaussian process regression*, J. Stat. Comput. Simul., 77 (2007), pp. 329–348.