RANDOMIZED NUMERICAL LINEAR ALGEBRA APPROACHES FOR APPROXIMATING MATRIX FUNCTIONS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Evgenia-Maria S. Kontopoulou

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2020

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

To Mum and Dad

ACKNOWLEDGMENTS

With the completion of this work I would like to thank several people who in one way or another helped me along the way and contributed to the completion of this dissertation.

First and foremost, I would like to express my deepest appreciation, gratitude and admiration to my advisor, Professor Petros Drineas. Petros believed in me and trusted me from the first moment. He was always encouraging, supporting and the true definition of an advisor. I could not have asked for a better advisor and I would always be grateful to him for all the opportunities he offered me and, of course, for putting up with me.

I would like to extend my deepest gratitude to Professor Efstratios Gallopoulos. Without him this journey would not have been possible. He was the one who passed on to me the love for linear algebra and scientific computing and the one who introduced me to randomized numerical linear algebra. His persistence on understanding basic concepts of our research area and being consistent in notation are still two of the most valuable qualifications that I have acquired over the years.

I am extremely thankful to the members of my Ph.D. committee; Professor David Gleich, and Professor Hemanta Maji. Professor Gleich, is an inspiration to me since my senior undergraduate years. His valuable comments are always “food for thought” and motives to improve my research. I am deeply grateful to Professor Maji’s lectures on mathematical toolkits. Through the extensive discussions during the lectures I was able to acquire a solid background on various aspects of my dissertation and be able to carry on extended technical proofs that some years ago were too difficult for me to conceptualize. I could not be any luckier to have both of them serving as members of my Ph.D. committee. My grateful thanks are also extended to Professor Jianlin Xia and Professor Kent Quanrud for their kindness to serve as external members of my preliminary and final exams committees, respectively.

My graduate studies at Purdue University would not have been possible without the financial support from the National Foundation of Science and Purdue University. I am thankful to both institutions for their generous financial contributions and support. I am also grateful for the financial support I have received from the Gerondelis Foundation and the John Rice Fellowship.

During my graduate studies I had the great pleasure to work/collaborate with many people whom I would like to acknowledge and thank for their contributions; Christos Boutsidis, Aritra Bose, Gregory Dexter, Mai Elkady, Kimon Fountoulakis, Professor Ananth Grama, Professor Ilse Ipsen, Vassilis Kalantzis, Prabhanjan Kambadur, Abhisek Kundu, Professor Malik Magdon-Ismail, Professor Peristera Paschou, Professor Wojciech Szpankowski and Anastasios Zouzias.

I am particularly thankful for all the friends I have met here at Purdue who they greatly contributed, in their own special way, to the completion of this Ph.D. Thank you Apostolia, Stella, Mai and Lydia; the days I have passed with each one of you are some of the most joyful memories I would always cherish. The most thankful I am for my best friends Kyriakos, Marios and Vassilis. Kyriakos for all the moments we shared at Purdue supporting each other, Marios for the endless meetings at Starbucks discussing research and life in general, and Vassilis for always being by my side, supporting and helping me as well as answering my weird questions about linear algebra and iterative methods since 2010. These three people have taught me that when true friendship exists, distance does not matter.

I cannot begin to express my gratitude to Petros, Peri and Venia for being my “family” away from home. Especially to Venia for always making me laugh and for teaching me the correct English pronunciation. I would be always indebted to all three of them for their kindness and support during the difficult times I’ve been through while at Purdue.

Finally, I am deeply grateful and thankful for my family. My mother Astrini, my sister Ioanna, my brother Kostas, my grandmother Dina and my uncle Michalis. Being so far away from them was the hardest thing I've ever done in my life. Their endless love and support were my reference points in all the difficult times I have been through. The biggest thank you, though, should be given to my father Spiros. Although not in life, he still is my driving force. The lessons he and my mother taught me along with the values they have given me, will accompany me for the rest of my life making me every day a better person. My only wish and desire is that they both are proud of me and all of my achievements.

West Lafayette, July 2020

Eugenia Kontopoulou

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

ABSTRACT

Kontopoulou, Evgenia-Maria PhD, Purdue University, August 2020. Randomized Numerical Linear Algebra Approaches for Approximating Matrix Functions. Major Professor: Petros Drineas.

This work explores how randomization can be exploited to deliver sophisticated algorithms with provable bounds for: (i) The approximation of matrix functions, such as the log-determinant and the Von-Neumann entropy; and (ii) The low-rank approximation of matrices. Our algorithms are inspired by recent advances in Randomized Numerical Linear Algebra (RandNLA), an interdisciplinary research area that exploits randomization as a computational resource to develop improved algorithms for large-scale linear algebra problems. The main goal of this work is to encourage the practical use of RandNLA approaches to solve Big Data bottlenecks at industrial level. Our extensive evaluation tests are complemented by a thorough theoretical analysis that proves the accuracy of the proposed algorithms and highlights their scalability as the volume of data increases. Finally, the low computational time and memory consumption, combined with simple implementation schemes that can easily be extended in parallel and distributed environments, render our algorithms suitable for use in the development of highly efficient real-world software.

PUBLICATIONS LIST

1. *Randomized Linear Algebra Approaches to Estimate the Von Neumann Entropy of Density Matrices.* E-M. Kontopoulou, G. Dexter, A. Grama, W. Szpankowski, P. Drineas. IEEE Transactions on Information Theory (2020), to appear

2. *TeraPCA: a Fast and Scalable Software Package to Study Genetic Variation in Tera-scale Genotypes.* A. Bose, V. Kalantzis, E-M. Kontopoulou, M. Elkady, P. Paschou and P. Drineas. Oxford Bioinformatics (2019), Vol. 35(19), pp. 3679-3683

3. *Randomized Linear Algebra Approaches to Estimate the Von Neumann Entropy of Density Matrices.* E-M. Kontopoulou, A. Grama, W. Szpankowski, P. Drineas. In Proceedings of the 2018 IEEE International Symposium on Information Theory (2018), pp. 2486-2490

4. *Structural Convergence Results for Approximation of Dominant Subspaces from Block Krylov Spaces.* P. Drineas, I. Ipsen, E-M. Kontopoulou, M. Magdon-Ismail. SIAM Journal on Matrix Analysis and Applications (2018), Vol. 39(2), pp. 567-586

5. *A Randomized Algorithm for Approximating the Log Determinant of a Symmetric Positive Definite Matrix.* C. Boutsidis, P. Drineas, P. Kambadur, E-M. Kontopoulou, A. Zouzias. Linear Algebra and its Applications (2017), Vol. 533, pp.95-117

6. *A Randomized Rounding Algorithm for Sparse PCA.* K. Fountoulakis, A. Kundu, E-M. Kontopoulou, P. Drineas. ACM Transactions on Knowledge Discovery from Data (2017), Vol. 11(3), No. 38, pp. 1-26

1 INTRODUCTION

Matrices arise in many scientific areas (e.g. genetics, physics, econometrics, text analysis etc.) as mediums to encode information between observations (usually represented by the rows of the matrix) and attributes (usually represented by the columns of the matrix). After the information matrix is constructed, functions are applied on it, to extract useful information such as important features, principal directions, etc.

Nowadays, the explosion of data and "the curse of dimensionality" favor the generation of large-scale and extremely sparse matrices. Although the volume of data increases rapidly, hardware improvements are not occurring at the same pace. Modern datasets can not fit in a regular-size main memory and state-of-the art algorithms for their management are no longer applicable.

The following example highlights some of the reasons why traditional algorithms do not scale in the case of large data. The exact computation of most matrix functions boils down to the computation of their Singular Value Decomposition (SVD). Except from its cubic computational complexity, SVD becomes prohibitive for large-scale matrices for two more reasons: (i) The entire matrix **must** reside in the main memory, and (ii) The often dense SVD factors require space larger than the sparse input data, leading to memory overuse.

However, few are the applications where high accuracy is crucial, i.e. to accurately compute all 16 decimal digits available. In most cases three to five accurate decimal digits are enough. Therefore, the target has shifted from exact computations to accurate approximations of values with low communication cost between main and secondary memory. Over the years many solutions have been proposed, e.g. block iterative methods [Saa11], communication efficient algorithms [BDHS11] etc.

This work explores how randomization can be exploited to deliver sophisticated algorithms with provable bounds for:

1. **The approximation of logarithm-based matrix functions (see, [BDK+17] and [KDS+20])**
2. **The low-rank approximation of matrices (see, [DIKMI18] and [FKKD17]).**

Our algorithms are inspired by recent advances in Randomized Numerical Linear Algebra (RandNLA), an interdisciplinary research area that exploits randomization as a computational resource to develop improved algorithms for large-scale linear algebra problems [DM18].

1.1 Approximation of logarithm-based matrix functions

The matrix logarithm is one of the most popular matrix functions. From the computation of differential equations to control and graph theory, and quantum mechanics various forms of the matrix logarithm are ubiquitous. In this work we concentrate on log-based functions of the form:

$$f(\log(g(\mathbf{A}))) = \gamma \tag{1.1}$$

where $f(\cdot)$ and $g(\cdot)$ are matrix or scalar functions, $\gamma \in \mathbb{R}$ and $\mathbf{A} \in \mathbb{R}^{n\times n}$. We further assume $\mathbf{A}$ to be a Symmetric Positive Semidefinite matrix (SPSD).

The naive solution to Eqn. (1.1) would have been to use a classic SVD algorithm[1] to compute the singular values of $\mathbf{A}$ [Hig08]. However, as already discussed, such computation, using conventional computational resources, is infeasible for large-scale matrices hence a different approach should be considered.

We propose approximation algorithms for two logarithm-based quantities: (i) $\log\left(\mathbf{det}\left(\mathbf{A}\right)\right)$, the logarithm of the determinant of $\mathbf{A}$, with applications in interior point methods, maximum likelihood computation, Gaussian graphical and Gaussian process models, metric and kernel learning etc. and (ii) $-\mathbf{Tr}\left(\mathbf{A}\log(\mathbf{A})\right)$, the Von

[1]Classic SVD algorithm refers to the trivial $\mathcal{O}\left(n^3\right)$ solutions.

Neumann entropy of the density matrix $\mathbf{A}$ with application in quantum mechanics and information theory.

Careful mathematical manipulation of the two quantities shows that both can be estimated using *classic approximation theory* (function approximations using Taylor or Chebyshev polynomials) and *RandNLA tools* (e.g. provably accurate random trace estimators [AT11], and provably accurate power method [Tre11, BDK$^+$17]). Furthermore, in certain cases when the matrix is singular (which is meaningful only for the Von Neumann entropy setting) we derive algorithms that leverage *random projections*, a powerful RandNLA tool, that reduces the dimensionality of the matrix in a way that the distance between the multidimensional points is preserved. Then, the sought quantity is approximated using the definition formula applied on the singular values extracted from the lower dimensional space. Experimental evaluation indicates that our algorithms can approximate both quantities accurately and considerably faster than the trivial $\mathcal{O}(n^3)$ approaches [BDK$^+$17, KDS$^+$20].

1.2 Low-rank approximation of matrices

Low-rank approximations are broadly used in data analysis as a simple yet effective filter to extract important information from noisy data. It is known that the best rank-k approximation is given by truncating the SVD [EY36]. However, as already discussed, SVD's cubic runtime highlights the demand for faster and accurate approximation methods. This dissertation is concerned with two issues; (i) the quality of the approximation to the top-k left singular vectors constructed from a block Krylov subspace [DIKMI18], and (ii) the extraction of sparse principal components [FKKD17] using a randomized rounding technique.

Krylov subspace methods are broadly used to accurately compute singular vectors and the corresponding singular values of matrices. In this work we are particularly interested in understanding the quality of the approximation of the top-k left singular vectors, $\mathbf{U}_k$ from a block Krylov subspace of block size q, $\mathcal{K}_q$, and derive bounds that can potentially be informative and useful in implementing block-Lanczos type methods in the future.

We derive two kind of bounds. First we bound the distance, in terms of angles, between the spaces spanned by $\mathbf{U}_k$ and $\widehat{\mathbf{U}_k}$. This distance measure can be ill-posed if $\mathbf{U}_k$ is not unique, meaning there is no singular gap between the k-th and the $k+1$-st singular values of $\mathbf{A}$. Second, we bound the error between $\mathbf{A}$ and its orthogonal projection into the space spanned by $\widehat{\mathbf{U}_k}$. This bound is gap-independent. The novelty of our work lies in the proofing techniques used. The traditional Lanczos convergence analysis [Saa11] was combined with near optimal low rank approximations using least squares. The critical point during this procedure was the connection of the principal angles of the two spaces with the least squares residuals, which are broadly studied in RandNLA.

The second problem we tackle, PCA, is a popular tool in learning information from high dimensional data, because it extracts the directions towards which the observations form clusters. Computationally, PCA is nothing more than the SVD of a correlation matrix. Motivated by recent results in sparse PCA we investigate how a combination of iterative optimization methods and simple randomized rounding algorithms can produce principal components that preserve the sparsity of the initial matrix and are inexpensive in terms of run time [FKKD17]. Experimental evaluations show that the principal components we derive are sparse and meaningful, but not pairwise orthogonal.

1.3 Structure

The dissertation is structured as follows. Sections 1.5, 1.6, 1.7 and 1.8 briefly introduce the four topics, that will be discussed in the chapters that follow. Each topic is accompanied by summaries of prior work and our results. Chapter 2 introduces basic notation used throughout the dissertation, the required background on linear algebra and probability theory as well as the basic RandNLA tools that will be useful for our algorithms and their analysis. Chapters 3 and 4 introduce the theoretical analysis and empirical evaluation of our algorithms for the approximation of the logarithm determinant of a symmetric positive definite matrix (SPD) and the approximation of the Von Neumann entropy of a density matrix, respectively. Chapter 5 presents the analysis and empirical evaluation of our partially randomized algorithm for sparse PCA. Chapter 6 discusses theoretical results for the approximation of dominant singular spaces from block Krylov subspaces. Finally, Chapter 7 includes future directions for the aforementioned problems.

1.4 Bibliographic note

Parts of section 1.5 and chapter 3 have been published in [BDK$^+$17]. Parts of section 1.6 and chapter 4 have been published in [KDS$^+$20]. Parts of section 1.7 and chapter 5 have been published in [FKKD17]. Parts of section 1.8 and chapter 6 have been published in [DIKMI18].

1.5 Approximation of the logarithm determinant of a symmetric positive definite matrix

Given a matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, the determinant of $\mathbf{A}$, denoted by $\mathbf{det}\,(\mathbf{A})$, is one of the most fundamental quantities associated with $\mathbf{A}$. Since its invention by Cardano and Leibniz in the late 16th century, the determinant has been an important mathematical concept with countless applications in numerical linear algebra and scientific computing. The advent of Big Data, increased the applicability of algorithms that compute, exactly or approximately, matrix determinants; see, for example, [LZL05,ZLLW08,ZL07,dBEG08,HSD$^+$13] for machine learning applications (e.g., Gaussian process regression) and [LP01,KL13,FHT08,PB97,PBGS00] for several data mining applications (e.g., spatial-temporal time series analysis).

Formal definitions of the determinant include the well-known formulas derived by Leibniz and Laplace; however, neither the Laplace expansion nor the Leibniz formula can be used to design an efficient, **polynomial-time**[2], algorithm to compute the determinant of $\mathbf{A}$. To achieve polynomial-time complexity for the computation of the determinant, one should rely on other properties of the determinant. For example, a standard approach would be to leverage the *LU* matrix decomposition (or the Cholesky decomposition for SPD matrices) to get an $\mathcal{O}\left(n^3\right)$ deterministic algorithm for the computation of $\mathbf{det}\left(\mathbf{A}\right)$.

Problem setup Our focus in this work is to approximate the logarithm of the determinant of a SPD matrix $\mathbf{A}$. The logarithm of the determinant, instead of the determinant itself, is important in several settings like the computation of the max-likelihood, Gaussian processes etc. (see, [LZL05, ZLLW08, ZL07, dBEG08, HSD+13, LP01, KL13, FHT08, PB97, PBGS00]).

Definition 1. [THE LOGDET PROBLEM] *Given a SPD matrix* $\mathbf{A} \in \mathbb{R}^{n\times n}$*, compute, exactly or approximately,* $\text{logdet}\left(\mathbf{A}\right) = \log\left(\mathbf{det}\left(\mathbf{A}\right)\right)$.

Related work Despite the importance of the problem, few methods have been proposed, for the approximation of the logarithm of the determinant of **general** SPD matrices. Most methods, including ours, given an SPD matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ with eigenvalues $\lambda_i,\ i - 1, \ldots, n$, are based on the following observation:

$$\text{logdet}\left(\mathbf{A}\right) = \log\left(\mathbf{det}\left(\mathbf{A}\right)\right) = \log\left(\prod_{i=1}^{n}\lambda_i\left(\mathbf{A}\right)\right) = \sum_{i=1}^{n}\log\left(\lambda_i\left(\mathbf{A}\right)\right) = \mathbf{Tr}\left(\log\left[\mathbf{A}\right]\right) \tag{1.2}$$

In words eqn. (1.2) states that the logarithm of the determinant of $\mathbf{A}$ equals the trace of its logarithm. Given this observation, a popular technique is to break the estimation of $\mathbf{Tr}\left(\log\left[\mathbf{A}\right]\right)$ into two parts. **First,** to use polynomial expansions (see, Section 2.3) to approximate powers of a matrix that is closely connected to $\log\left[\mathbf{A}\right]$ like, the Martin expansion (see, [Mar92, BP99, HAB14]), Chebyshev polynomials (see, [PL04, HMS15]) or Taylor polynomials (see, [BDK+17]). **Second**, to employ a trace

[2]Indeed both methods, are computationally inefficient as they both require factorial computational time.

estimation technique like the Huntchinson's trace estimator (see, [HMAS17]), the Gaussian trace estimator (see [HAB14]) or some kind of loose lower bound (see, [BP99]). In [PL04], the authors use closed formulas to compute the trace of small powers (up to 4) of a matrix.

Most of the tools used require matrices with bounded spectrum, limiting the applicability of most methods to SPD matrices of special structure, e.g. large sparce matrices with eigenvalues in the interval $[-1, 1]$ and structure of the form $\mathbf{I}_n - \alpha\mathbf{D}$ with $0 < \alpha < 1$ (see, [BP99]), diagonally dominant (SDD) SPD matrices (see, [HAB14]) or SPD matrices with eigenvalues in a more restricted interval $(\theta_1, 1-\theta_1)$ where $0 < \theta_1 < 1/2$ (see, [HMS15]). Table 1.1 summarizes the above discussion, and help to directly compare our results with previous work.

Table 1.1.: Summary of prior work on the approximation of the logarithm of the determinant of an SPD matrix that can directly be compared to our approach. $\mathbf{I}_n$ is the $n \times n$ identity matrix, $\alpha \in \mathbb{R}$ with $0 < \alpha < 1$ and $\mathbf{D} \in \mathbb{R}^{n \times n}$ is a SPD matrix with eigenvalues in the interval $[-1, 1]$. m refers to the number of expansion terms retained, p refers to the number of algorithm repetitions or equivalently to the number of random vectors created for the trace estimation, $\varepsilon < 1$ is the user defined accuracy parameter and $\kappa(\mathbf{A})$ refers to the condition number of the SPD matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$.

[BP99]	**Matrix type**	**Expansion type**
	$\mathbf{I}_n - \alpha\mathbf{D}$	Martin [Mar92]
	Run time	**Bound**
	$\mathcal{O}(n \log n)$	$\frac{n \cdot \alpha^{m-1}}{(m+1)(1-\alpha)} + 1.96 \cdot \sqrt{\frac{\sigma^2}{p}}$
[HAB14]	**Matrix type**	**Expansion type**
	SDD	Martin [Mar92]
	Run time	**Bound**
	$\mathcal{O}\left(\mathrm{nnz}(\mathbf{A}) \frac{1}{\varepsilon^2} \log^3(\mathrm{n}) \log^2\left(\frac{\mathrm{n} \cdot \kappa(\mathbf{A})}{\varepsilon}\right)\right)$	$\varepsilon \cdot n$
[HMS15]	**Matrix type**	**Expansion type**
	$\lambda_i(\mathbf{A}) \in [-1, 1]$	Stochastic Chebyshev
	Run time	**Bound**
	$\mathcal{O}\left(\sqrt{\frac{1}{\theta_1}} \log \frac{1}{\theta_1}\right)$	$\varepsilon \cdot n$

A slightly different, but interesting, approach is to explicitly estimate $\mathbf{Tr}(\log[\mathbf{A}])$. This idea appeared in [HMAS17] in which the authors use Chebyshev polynomial interpolation to estimate spectral sums of the form $\mathbf{Tr}\left(f(\mathbf{X})\right)$ where $f(x)$ is a function with special characteristics. They apply their method on the approximation of the logarithm determinant of SPD matrices with eigenvalues in the interval $[a, b]$ $a, b > 0$ and in general non-singular matrices. Although the novel approach, their algorithms require knowledge of the maximum and minimum eigenvalues, or some rough bounds for them, which is not always easy to acquire. Both algorithms achieve additive error bounds with running time that depends on the condition number of the matrix. Along the lines of [HMAS17], the authors of [UCS17] propose a method that instead of Chebyshev interpolation utilizes stochastic Lanczos quadrature to estimate traces of matrix functions. They also apply their approach on the estimation of the logarithm determinant of general SPD matrices deriving additive error bounds. Unlike [HMAS17], the method in [UCS17] doesn't seem to require prior knowledge of any eigenvalues.

Finally, a completely different method has appeared in [SAI17]. The authors approximate the logarithm of the determinant of an SPD matrix, using a randomized subspace iteration approach. The derived bounds are not directly comparable to our bounds. The work of [Reu02] uses an approximate matrix inverse to compute the n-th root of $\mathbf{det}\left(\mathbf{A}\right)$ for large sparse SPD matrices. The error bounds in this work are a-posteriori and thus not directly comparable to our bounds.

Our contributions We present a fast approximation algorithm for the problem of Definition 1. Our main algorithm (Algorithm 7) is randomized and runs in time

$$\mathcal{O}\left(\left(\frac{m}{\varepsilon^2} + \log(n)\right) \log\left(\frac{1}{\delta}\right) \mathrm{nnz}\left(\mathbf{A}\right)\right),$$

where $\mathrm{nnz}\left(\mathbf{A}\right)$ denotes the number of non-zero elements in $\mathbf{A}$, $0 < \delta < 1$ denotes the failure probability of our algorithm, $m > 0$ and $\varepsilon > 0$ are user-controlledled accuracy parameters, specified in the input of the algorithm.

The first step of our approximation algorithm uses the power method (see, Section 2.5.1) to compute an approximation to the dominant eigenvalue of $\mathbf{A}$ that will be denoted as $\lambda_{\max}$. This value will be used in a normalization (preconditioning) step in order to compute a convergent matrix-Taylor expansion. The second step of our algorithm leverages a truncated matrix-Taylor expansion (see, Section 2.3.1) of a suitably constructed matrix in order to compute an approximation of the log determinant. This second step leverages a randomized trace estimation algorithm from [AT11] (see, Section 2.5.2).

Let $\widehat{\operatorname{logdet}}(\mathbf{A})$ be the value returned by our approximation algorithm (Algorithm 7); let $\operatorname{logdet}(\mathbf{A})$ be the true value of the logarithm determinant of $\mathbf{A}$; let $\lambda_i(\mathbf{A})$ denote the i-th eigenvalue of $\mathbf{A}$ for all $i = 1, \ldots, n$ with $\lambda_1(\mathbf{A}) \geq \lambda_2(\mathbf{A}) \geq \ldots \geq \lambda_n(\mathbf{A}) > 0$; let $\kappa(\mathbf{A}) = \lambda_1(\mathbf{A})/\lambda_n(\mathbf{A})$ be the condition number of $\mathbf{A}$. Our main result, proven in Lemma 10, states that if,

$$m \geq \left\lceil 7\kappa(\mathbf{A}) \log\left(\frac{1}{\varepsilon}\right) \right\rceil, \tag{1.3}$$

where m is the number of Taylor terms retained, then, with probability at least $1-2\delta$,

$$\left|\widehat{\operatorname{logdet}}(\mathbf{A}) - \operatorname{logdet}(\mathbf{A})\right| \leq 2\varepsilon \underbrace{\sum_{i=1}^{n} \log\left(7 \cdot \frac{\lambda_1(\mathbf{A})}{\lambda_i(\mathbf{A})}\right)}_{\Gamma}. \tag{1.4}$$

Given our choice of m (see, eqn. (1.3)) the running time of the algorithm becomes

$$\mathcal{O}\left(\left(\kappa(\mathbf{A}) \log\left(\frac{1}{\varepsilon}\right) \frac{1}{\varepsilon^2} + \log n\right) \log\left(\frac{1}{\delta}\right) \operatorname{nnz}(\mathbf{A})\right). \tag{1.5}$$

eqn. (1.5) shows that the runtime depends **linearly** on the condition number, $\kappa(\mathbf{A})$. The error of our algorithm scales with Γ, a quantity that is not immediately comparable to $\operatorname{logdet}(\mathbf{A})$. It is worth noting that the Γ term increases **logarithmically** with respect to the ratios $\lambda_1(\mathbf{A})/\lambda_i(\mathbf{A}) \geq 1$ $i = 1, \ldots, n$. Observe that

$$\frac{\lambda_1(\mathbf{A})}{\lambda_i(\mathbf{A})} \leq \frac{\lambda_1(\mathbf{A})}{\lambda_n(\mathbf{A})} = \kappa(\mathbf{A}).$$

Thus, an obvious, but potentially loose upper bound for Γ, is

$$\Gamma = \sum_{i=1}^{n} \log\left(7 \cdot \frac{\lambda_1(\mathbf{A})}{\lambda_i(\mathbf{A})}\right) \leq n \cdot \log\left(7\kappa(\mathbf{A})\right). \tag{1.6}$$

Our second result handles the family of SPD matrices whose eigenvalues all lie in the interval $(\theta_1, 1)$, with $0 < \theta_1 < 1$; this setting was proposed in [HMS15]. In this case, a simplified version of Algorithm 7 returns a relative error approximation to the log-determinant of the input matrix. Indeed, Lemma 12 proves that, with probability at least $1 - \delta$,

$$\left|\widehat{\mathrm{logdet}}\left(\mathbf{A}\right) - \mathrm{logdet}\left(\mathbf{A}\right)\right| \leq 2\varepsilon |\mathrm{logdet}\left(\mathbf{A}\right)|.$$

The running time of the simplified algorithm is

$$\mathcal{O}\left(\frac{1}{\theta_1}\log\left(\frac{1}{\varepsilon}\right)\frac{1}{\varepsilon^2}\log\left(\frac{1}{\delta}\right)\mathrm{nnz}\left(\mathbf{A}\right)\right). \tag{1.7}$$

Observe, that

$$\frac{1}{\theta_1} \geq \frac{\lambda_1}{\lambda_n} = \kappa\left(\mathbf{A}\right).$$

Thus, the run time depends on the condition number, $\kappa\left(\mathbf{A}\right)$.

1.6 Estimation of the Von Neumann entropy of density matrices

Entropy is a fundamental quantity in many areas of science and engineering. The *Von Neumann entropy*, named after John Von Neumann, is an extension of classical entropy concepts to the field of quantum mechanics. In his work, Von Neumann introduced the notion of a *density matrix*, which facilitated extension of the tools of classical statistical mechanics to the quantum domain in order to develop a theory of quantum mechanics.

From a linear algebraic perspective (see, Section 4.1 for details) the real density matrix $\mathbf{R}$ is a SPSD matrix in $\mathbb{R}^{n\times n}$ with unit trace. Let p_i, $i = 1\dots n$ be the eigenvalues of $\mathbf{R}$ in decreasing order; then, the entropy of $\mathbf{R}$ is defined as[3]

$$\mathcal{H}(\mathbf{R}) = -\sum_{i=1}^{n} p_i \log p_i. \tag{1.8}$$

The above definition is a proper extension of both the Gibbs entropy and the Shannon entropy to the quantum case. It implies an obvious algorithm to compute $\mathcal{H}(\mathbf{R})$ by computing the eigendecomposition of $\mathbf{R}$; known algorithms for this task can be prohibitively expensive for large values of n, particularly when the matrix becomes dense [GV96], which is usually the case in our setting.

Problem setup Motivated by the high computational cost, we seek numerical algorithms that approximate the Von Neumann entropy of large density matrices that are faster than the trivial $\mathcal{O}(n^3)$ approach.

Definition 2. [The Von Neumann Entropy Problem] *Given a density matrix* $\mathbf{R} \in \mathbb{R}^{n\times n}$*, compute, exactly or approximately, the Von Neumann entropy of* $\mathbf{R}$*,* $\mathcal{H}(\mathbf{R})$.

Related work The first non-trivial algorithm to approximate the Von Neumann entropy of a density matrix appeared in [WBS14]. Their approach is essentially similar in spirit to our second approach (see, Section 4.3). Indeed, Algorithm 12 was inspired by their approach.

Independently and in parallel with our work, [MNS+18] presented a multipoint interpolation algorithm (building upon [HNO08]) to compute a relative error approximation for the entropy of a real matrix with bounded condition number. The proposed running time of Theorem 35 in [MNS+18] does not depend on the condition number of the input matrix, which is a clear advantage in the case of ill-conditioned

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

matrices. However, the dependence of their algorithm on terms like $(\log n/\varepsilon)^6$ or $n^{1/3}\mathrm{nnz}(\mathbf{A}) + \mathrm{n}\sqrt{\mathrm{nnz}(\mathbf{A})}$ (where $\mathrm{nnz}(\mathbf{A})$ represents the number of non-zero elements of the matrix $\mathbf{A}$) could blow up the running time of the proposed algorithm for reasonably conditioned matrices.

In [CSS18] the authors follow a different approach to approximate the Von Neumann entropy. They describe an algorithm that builds upon the randomized subspace iteration to compute a reduced full-rank matrix, $\hat{\mathbf{A}}$. Then the eigenvalues of $\hat{\mathbf{A}}$ are computed and the approximation to the Von Neumann entropy follows, using the definition. This approach is comparable to our third approach (see, Section 4.5), but it is limited in matrices with a singular gap between the k-th and $k+1$-st eigenvalues, for some $k < \mathrm{rank}(\mathbf{A})$. The main result (see, [CSS18, Theorem 3]) shows that the approximation error is proportional to the Von Neumann entropy of the best rank-k approximation up to a multiplicative factor that depends on the singular gap between the k-th and $k+1$-st eigenvalues of $\mathbf{A}$. This dependency on the singular gap is an artifact of the subspace iteration and the Krylov subspace methods in general. We will extend on this in Chapter 6. We should note that (i) the method in [CSS18] is extended to compute approximations to the traces of matrix functions, and (ii) the method works for HPSD matrices [4].

Finally, an interesting line of work concerns the approximation of the Von Neumann entropy of the graph Laplacian. It has been proven that a graph can be characterized by a density matrix of pure states [BGS06], cultivating a connection between graph theory and the field of quantum theory. Interested readers can refer to [CWLR19] or [CHHS20] for approaches to approximate the Von Neumann entropy of the combinatorial graph Laplacian.

Our contributions We present and analyze **three** randomized algorithms to approximate the Von Neumann entropy of density matrices as defined in Definition 2.

[4]This should be considered with caution as there is not a thorough analysis to support this claim.

The first two algorithms are heavily based on polynomial approximations; the first approach (see, Section 4.2) uses Taylor polynomials (see, Section 2.3.1), while the second approach (see, Section 4.3) uses Chebyschev polynomials (see, 2.3.2). More specifically,

Approach 1: Given a density matrix $\mathbf{R} \in \mathbb{R}^{n \times n}$ Algorithm 10 runs in time

$$\mathcal{O}\left(\left(m\frac{1}{\varepsilon^2} + \log(n)\right)\log\left(\frac{1}{\delta}\right) \cdot \operatorname{nnz}(\mathbf{R})\right),$$

where $\operatorname{nnz}(\mathbf{R})$ denotes the number of non-zero elements in $\mathbf{R}$. $0 < \delta < 1$ denotes the failure probability of the algorithm and $\varepsilon > 0$ and $m > 0$ are two user-controlledled accuracy parameters specified in the input of the algorithm.

The first step of our approximation algorithm uses the power method (see, Section 2.5.1) to compute an approximation, u, to the dominant eigenvalue of $\mathbf{R}$, p_1. This value will be used in a normalization (preconditioning) step in order to compute a convergent matrix-Taylor expansion. The second step of our algorithm leverages a truncated matrix-Taylor expansion (see, Section 2.3.1 of a suitably constructed matrix (closely related to $\log[\mathbf{R}]$) in order to compute an approximation of the Von Neumann entropy. This second step leverages a randomized trace estimation algorithm from [AT11] (see, Section 2.5.2).

Let $\widehat{\mathcal{H}}(\mathbf{R})$ be the value returned by Algorithm 10; let $\mathcal{H}(\mathbf{R})$ be the true value of the Von Neumann entropy of $\mathbf{R}$; let $p_i(\mathbf{R})$ denote the i-th eigenvalue of $\mathbf{R}$ for all $i = 1, \ldots, n$ with $1 > u \geq p_1(\mathbf{R}) \geq p_2(\mathbf{R}) \geq \ldots \geq p_n(\mathbf{R}) \geq \ell > 0$. Our main result, proven in Lemma 4, states that if

$$m = \left\lceil \frac{u}{\ell} \log \frac{1}{\varepsilon} \right\rceil \tag{1.9}$$

then, with probability at least $1 - 2\delta$

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

where m is the number of Taylor terms retained. Given our choice of m in eqn. (1.9), the running time of the algorithm becomes:

$$\mathcal{O}\left(\left(\frac{u}{\ell}\cdot\log\left(\frac{1}{\varepsilon}\right)\frac{1}{\varepsilon^2}+\log(n)\right)\log\left(\frac{1}{\delta}\right)\cdot\operatorname{nnz}\left(\mathbf{R}\right)\right).$$

Observe that,

$$\frac{u}{\ell}\geq\frac{p_1}{p_n}=\kappa\left(\mathbf{R}\right).$$

Thus, it is obvious that the running time depends on the condition number, $\kappa\left(\mathbf{R}\right)$.

Approach 2: Given a density matrix $\mathbf{R}\in\mathbb{R}^{n\times n}$ the randomized Algorithm 12 runs in time

$$\mathcal{O}\left(\left(m\frac{1}{\varepsilon^2}+\log\left(n\right)\right)\log\left(\frac{1}{\delta}\right)\cdot\operatorname{nnz}\left(\mathbf{R}\right)\right)$$

where $\operatorname{nnz}\left(\mathbf{R}\right)$ denotes the number of non-zero elements in $\mathbf{R}$. $0<\delta<1$ denotes the failure probability of our algorithm and $\varepsilon>0$ and $m>0$ are two user-controlled accuracy parameters specified in the input of the algorithm.

The first step of our approximation algorithm uses the power method (see, Section 2.5.1) to compute an approximation, u, to the dominant eigenvalue of $\mathbf{R}$, p_1. This value will be used in a normalization (preconditioning) step in order to be able to use Chebyshev polynomials. The second step of our algorithm leverages an expansion of Chebyshev polynomials (see, Section 2.3.2) of the first kind to approximate the matrix analog of the function $f(x)=x\log\left(x\right)$. The third step leverages a randomized trace estimation algorithm from [AT11] to approximate the Von Neumann entropy (see, Section 2.5.2).

Let $\widehat{\mathcal{H}}\left(\mathbf{R}\right)$ be the value returned by Algorithm 12; let $\mathcal{H}\left(\mathbf{R}\right)$ be the true value of the Von Neumann entropy of $\mathbf{R}$; let $p_i\left(\mathbf{R}\right)$ denote the i-th eigenvalue of $\mathbf{R}$

for all $i = 1, \ldots, n$ with $1 > u \geq p_1(\mathbf{R}) \geq p_2(\mathbf{R}) \geq \ldots \geq p_n(\mathbf{R} \geq \ell > 0$. Our main result, proven in Lemma 5, states that if

$$m = \left\lceil \sqrt{\frac{u}{2 \cdot \varepsilon \cdot \ell \cdot \log(1/(1-\ell))}} \right\rceil \tag{1.10}$$

then, with probability at least $1 - 2\delta$

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

Where m is the number of Chebyshev polynomials of the first kind that were used for the approximation. Given our choice of m in eqn. (1.10), the running time of the algorithm becomes:

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

Observe that,

$$\frac{u}{\ell} \geq \frac{p_1}{p_n} = \kappa\left(\mathbf{R}\right).$$

Thus, it is obvious that the running time depends on the square root of the condition number, $\kappa\left(\mathbf{R}\right)$. Therefore, we expect the Chebyshev-based approach to be faster than the Taylor-based one.

Our third approach (see, Section 4.5) is fundamentally different, if not orthogonal, to the previous two approaches. It leverages the power of random projections [DM16, Woo14] to approximate numerical linear algebra quantities, such as the eigenvalues of a matrix. More specifically,

Approach 3:

Given a density matrix $\mathbf{R} \in \mathbb{R}^{n \times n}$ Algorithm 15 tackles a case when the two previous polynomial-based algorithms do not apply. That is when the density matrix is low rank. In that case Theorems 4 and 5 fail as they both require the eigenvalues of $\mathbf{R}$ to be strictly greater than zero. Algorithm 15 runs in time

$$\mathcal{O}\left(\mathrm{nnz}\left(\mathbf{R}\right)+\mathrm{ns}^2\right),$$

where $\mathrm{nnz}\left(\mathbf{R}\right)$ denotes the number of non-zero elements in $\mathbf{R}$; $0 < s \ll n$ is a user controlled accuracy parameter specified in the input of the algorithm.

The first step of Algorithm 15 uses random projections (see, Section 2.5.3) to reduce the dimensionality of the input matrix while preserving the relative distance between the multidimensional points. Towards this end we propose two constructions for random projectors; (i) the fast subsampled Hadamard transform (see, Algorithm 4) and (ii) the input sparsity transform (see, Algorithm 5). In the second step of Algorithm 15 at most k singular values from the new subspace are computed and eqn. (1.8) is used to return an approximation to the Von Neumann entropy of $\mathbf{R}$.

Let $\widehat{\mathcal{H}}\left(\mathbf{R}\right)$ be the value returned by Algorithm 15; let $\mathcal{H}\left(\mathbf{R}\right)$ be the true value of the Von Neumann entropy of $\mathbf{R}$; let p_i and $\tilde{p}_i$ be the i-th largest eigenvalue of $\mathbf{R}$ and its approximation respectively. Our main result, proven in Theorem 10, states that if the number of non-zero eigenvalues of $\mathbf{R}$ is at most $k \ll n$ and $\varepsilon < 1/2$ is the accuracy parameter, then with probability at least 0.9:

$$\left|p_i^2 - \tilde{p}_i^2\right| \leq \varepsilon p_i^2 \tag{1.11}$$

for all $i = 1 \ldots k$ and:

$$\left|\mathcal{H}\left(\mathbf{R}\right) - \widehat{\mathcal{H}}\left(\mathbf{R}\right)\right| \leq \sqrt{\varepsilon}\mathcal{H}\left(\mathbf{R}\right) + \sqrt{\frac{3}{2}}\varepsilon. \tag{1.12}$$

In words eqn. (1.11) states that Algorithm 15 returns approximations to the eigenvalues of $\mathbf{R}$ such that each approximate eigenvalue squared is relatively close to the corresponding actual eigenvalue squared. It further states that

Algorithm 15 returns a relative-additive approximation to the Von Neumann entropy of **R**. Using the input sparsity transform (see, Section 2.5.3) to construct the random projector Algorithm 15 runs in time

$$\mathcal{O}\left(\mathrm{nnz}\left(\mathrm{R}\right)+\mathrm{nk}^4\varepsilon^4\right).$$

Finally, to the best of our knowledge, we provide **the first** coherent analysis for the approximation of the Von Neumann entropy of **Hermitian** density matrices. This analysis is only available for our polynomial-based algorithms (see, Theorems 8 and 9). We leave it as an open problem the extension of our third approach on the Hermitian case.

1.7 A randomized rounding algorithm for sparse principal component analysis (PCA)

The Principal Components Analysis (PCA) and the Singular Value Decomposition (SVD) are fundamental data analysis tools, expressing a data matrix in terms of a sequence of orthogonal vectors of decreasing importance. While these vectors exhibit strong optimality properties and are often interpreted as fundamental latent factors that underlie the observed data, they are linear combinations of up to all the data points and features. As a result, they are notoriously difficult to interpret in terms of the underlying processes generating the data [MD09].

The seminal work of [dGJ07] introduced the concept of Sparse PCA, where sparsity constraints are enforced on the singular vectors in order to improve interpretability. As noted in [dGJ07, MD09, PDK13], an example where sparsity implies interpretability is document analysis, where sparse principal components can be mapped to specific topics by inspecting the (few) keywords in their support.

Problem setup Formally, Sparse PCA can be defined as a maximization problem as shown in Definition 3.

Definition 3 (The Sparce PCA Problem). *Given an input matrix* $\mathbf{X} \in \mathbb{R}^{m\times n}$ *with covariance matrix* $\mathbf{A} = \mathbf{X}^\top\mathbf{X} \in \mathbb{R}^{n\times n}$ *and an integer parameter* $k > 0$*, we seek to find a vector* $\mathbf{x}_{opt}$ *that is the solution to:*

$$\begin{aligned} \mathcal{Z}_{opt} = \max_{\mathbf{x}\in\mathbb{R}^n} \quad & \mathbf{x}^\top\mathbf{A}\,\mathbf{x} && \text{(1.13a)}\\ \text{s.t.} \quad & \|\mathbf{x}\|_0 = k && \text{(1.13b)}\\ & \|\mathbf{x}\|_2 = 1 && \text{(1.13c)} \end{aligned}$$

First, it is pretty obvious that removing (1.13b) problem (1.13) reduces to the exact PCA problem (see, Section 2.2.3) that has a well known solution; $\mathbf{x}_{opt}$, the eigenvector of $\mathbf{A}$ that corresponds to the dominant eigenvalue, $\mathcal{Z}_{opt}$, of $\mathbf{A}$. Introducing, the ℓ_0 constraint (1.13b) transforms problem (1.13) to NP-hard [MWA12]. Even simple relaxations of the constraints such as the ones in problem (1.14) are still NP-hard due to the restriction that $\mathbf{x}_{opt}$ must contain at most k non-zero entries [dGJ07].

$$\begin{aligned} \mathcal{Z}_{opt} = \max_{\mathbf{x}\in\mathbb{R}^n} \quad & \mathbf{x}^\top\mathbf{A}\,\mathbf{x} && \text{(1.14a)}\\ \text{s.t.} \quad & \|\mathbf{x}\|_0 \leq k && \text{(1.14b)}\\ & \|\mathbf{x}\|_2 \leq 1 && \text{(1.14c)} \end{aligned}$$

Notice that, although $\mathbf{x}^\top\mathbf{A}\mathbf{x}$ [5] is convex, the same does not hold for (1.14b). It is known that no ℓ_0 constraint can be convex; indeed $\|\cdot\|_0$ is not an actual norm[6]. On the other hand, constraint (1.14c) is convex; the ℓ_1 unit ball is a hypercube therefore a convex set. Common approaches that approximate $\mathbf{x}_{opt}$ include semi-definite programming, sparsification of the top singular vector using a threshold, convex relaxation of the constraints, e.t.c.

[5]Also interpreted as the maximization of the variance of $\mathbf{x}$.
[6]This can be also proven by a simple counterexample.

Related work The simplest sparse PCA approaches are to either rotate [Jol95] or threshold [CJ95] the top singular vector of the matrix $\mathbf{A}$. Such simple methods are computationally efficient and tend to perform very well in practice (see, Section 5.3). However, there exist cases where they fail (see, [CJ95] and Section 5.3). An alternative line of research focused on solving relaxations of eqn. (1.13). For example, an ℓ_1 relaxation of eqn. (1.13) was first used in SCoTLASS [JTU03]. Another possible relaxation is a regression-type approximation [ZHT06], which was implemented in [SCLE18]. Finally, efficient optimization methods have been developed for the sparse PCA problem. For example, the generalized power method was proposed in [JNRS10]: this method calculates stationary points for penalized versions of eqn. (1.13).

Despite the many approaches that were developed for sparse PCA, only a handful of them provide any type of theoretical guarantees regarding the quality of the obtained (approximate) solution. For example, the semi-definite relaxation of [dGJ07] was analyzed in [AW08], albeit for the special case where $\mathbf{A}$ is a spiked covariance matrix with a sparse maximal singular vector. Briefly, [AW08] studies conditions for the dimensions m and n of the initial data matrix $\mathbf{X}$, and the sparsity parameter k, so that the semi-definite relaxation of [dGJ07] recovers the sparsity pattern of the optimal solution of eqn. (1.13). Other attempts for provable results include the work of [dBG08], which was later analyzed in [dBG14]. In the latter paper, the authors show bounds for the semi-definite relaxation of [dBG08], in the special case that the data points are sampled using Gaussian models with a single sparse leading singular vector. Strong compressed-sensing-type conditions were used in [YZ13] to guarantee recovery of the optimal solution of eqn. (1.13) using a truncated power method. However, [YZ13] requires that the optimal solution is approximately sparse and also that the noise matrix has sparse submatrices with small spectral norm. The work of [PDK13] describes a greedy combinatorial approach for sparse PCA and provides relative-error bounds for the resulting solution under the assumption that the covariance matrix $\mathbf{A}$ has a decaying spectrum. It is important to note that in all the above papers special assumptions are necessary regarding the input data in order to guarantee the theoretical bounds.

The last couple of years Sparse PCA has regained interest. Although, multiple works have appeared, we selected to report few of the peer-reviewed ones. In [EZM$^+$20] the authors present a method for sparse PCA via variable projections. In this case, the sparse PCA problem is formulated as a value-function optimization problem and is approximately solved using proximal gradient methods to find a stationary point. In [VDTC$^+$19] the authors describe an approach to the sparse PCA problem that adds weights to the elements. This approach is equivalent to adding noise or regularization to the input data. Finally, in [PZ19] the authors present and analyze a sparse PCA approach that is suitable for data with missing entries.

There are also connections between sparse approximations and subspace learning methods, which are widely used in machine learning and data mining. Methods that enforce sparsity have been developed for Human Activity Recognition [LZW$^+$16], Image Classification [LWGX16] and Natural Language Processing [DL20]. Moreover, some of these methods have been applied to multidimensional data [TLWM07].

Our contributions We present and analyze a simple, two-step algorithm to approximate the optimal solution of the problem of eqn. (1.14). The proposed approach first finds a stationary point of an ℓ_1-penalized version of problem (1.14). Then, a randomized rounding strategy is employed to sparsify the resulting dense solution of the ℓ_1-penalized problem.

Our guarantee is that given an input matrix $\mathbf{X}$, its covariance matrix $\mathbf{A} = \mathbf{X}^\top\mathbf{X}$, a sampling factor $s > 0$ and an accuracy parameter $\varepsilon > 0$, the solution $\hat{\mathbf{x}}_{opt}$ of our two-step algorithm follows the properties:

1. $\mathbf{E}\left[\|\hat{\mathbf{x}}_{opt}\|_0\right] \leq s$.

2. With probability at least 3/4,

$$\|\hat{\mathbf{x}}_{opt}\|_2 \leq 1 + 0.15\varepsilon.$$

3. With probability at least 3/4,

$$\hat{\mathbf{x}}_{opt}^\top \mathbf{A} \hat{\mathbf{x}}_{opt} \geq \mathbf{x}_{opt}^\top \mathbf{A} \mathbf{x}_{opt} - \varepsilon. \tag{1.15}$$

In words, the above theorem states that our sparse vector $\hat{\mathbf{x}}_{opt}$ is almost as good as the optimal vector $\mathbf{x}_{opt}$ in terms of capturing (with constant probability) almost as much of the spectrum of $\mathbf{A}$ as $\mathbf{x}_{opt}$ does.

1.8 Structural convergence results for approximation of dominant subspaces from block Krylov spaces

Recently there has been an increased interest in Theoretical Computer Science, to analyze randomized methods for the approximation of dominant subspaces and low rank approximations from block Krylov spaces [MM15, WZZ15].

At this point it is crucial to describe the difference between low-rank approximation and the approximation of subspaces of an arbitrary matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$. The objective of a **low-rank approximation** is to find a matrix $\mathbf{Z}$ with orthonormal columns that makes $\|\mathbf{A} - \mathbf{Z}\mathbf{Z}^\top \mathbf{A}\|$ small in some unitarily invariant norm [HMT11, Woo14]. In contrast, a **subspace approximation** aims to find a space $\mathcal{K}$ that has a small angle with the dominant target space, which in our case would be the space spanned by the k left singular vectors associated with the top-k singular vectors of $\mathbf{A}$. The problem of subspace approximation is a much harder problem than the low-rank approximation one. A dominant subspace is said to be well-defined when, the top k singular values are separated by a gap from the remaining singular values of $\mathbf{A}$. In contrast, low-rank approximations can be defined even in the absence of a singular value gap. Furthermore, accuracy results for dominant subspace approximations are automatically informative for low-rank approximations, but not vice versa.

Problem setup This work will follow a line of research in which block Krylov space methods are emerged to compute dominant left singular vector spaces of general rectangular matrices.

Definition 4. [The Dominant Subspace Approximation and Low-Rank Approximation Problem] *Given a matrix* $\mathbf{A} \in \mathbb{R}^{m\times n}$ *and a positive integer* $k < \operatorname{rank}(\mathbf{A})$. *Let* $\mathbf{U}_k$ *be the top-*k *left singular vectors of* $\mathbf{A}$. *The objective is to construct approximations* $\hat{\mathbf{U}}_k \in \mathbb{R}^{m\times k}$ *for* $\mathbf{U}_k$ *for the following problems:*

Dominant subspace approximation *bound the angles between the column space of* $\mathbf{U}_k$, $\operatorname{range}(\mathbf{U}_\mathrm{k})$ *and the column space of* $\hat{\mathbf{U}}_k$, $\operatorname{range}\left(\hat{\mathbf{U}}_\mathrm{k}\right)$.

Low rank approximation *bound the approximation error between* $\mathbf{A}$ *and its projection into* $\operatorname{range}\left(\hat{\mathbf{U}}_\mathrm{k}\right)$.

Related work Analyses of numerical methods that compute dominant subspaces and eigenvectors from randomized starting vectors date back at least to the 1980s. They include the power method and inverse iteration [Dix83, JI92], and information theoretic analyses of Lanczos methods [KW92,KW94]. Well known analyses in Theoretical Computer Science focus on low-rank approximations [HMT11, Woo14], rather than subspace computations, and as such tend not to produce bounds for the accuracy of subspaces. A popular approach towards low-rank approximation is subspace iteration, which makes use of only the last iterate $(\mathbf{A}\mathbf{A}^\top)^q\mathbf{A}\mathbf{X}$ of a block Krylov subspace method [HMT11, Woo14]. More recently [Sai19] proves bounds, in terms of unitarilly invariant and Schatten-p norms, for the angles between singular subspaces and the subspace spanned by $(\mathbf{A}\mathbf{A}^\top)^q\mathbf{A}\mathbf{X}$, for the low-rank approximation of singular spaces from $(\mathbf{A}\mathbf{A}^\top)^q\mathbf{A}\mathbf{X}$ and for the singular values.

Then came block Krylov methods, which exploit all of the iterates $(\mathbf{A}\mathbf{A}^\top)^j\mathbf{A}\mathbf{X}$, $0 \le j \le q$. The analysis in [MM15] relies on generalized matrix functions [ABF16, HBI73], but is limited to Gaussian random matrices for starting guesses $\mathbf{X} \in \mathbb{R}^{n\times k}$, and Chebyshev polynomials for ϕ. The *gap-dependent bound* [MM15, Theorem 13] requires a gap between the k-th and $(k+1)$-st singular values. However, [MM15, Theorems 10, 11, and 12] do not require a singular value gap. Such *gap-independent bounds* are informative for low-rank approximations, but not for computations of specific subspaces. In lines of [MM15], [WZZ15] focuses on gap-independent bounds and random Gaussian starting guesses (see, [WZZ15, Theorem 3.1]). The proof techniques in [WZZ15] are similar to ours, and leverage [BDMI11]. In [DI19] the authors extend our "gap-dependent" angle bounds to bounds in terms of general Schatten-p norms. They prove "gap-independent" bounds in terms of general Schatten-p norms for the quality of the low-rank approximations under additive perturbations in the projector basis and the matrix, or under additive and multiplicative perturbations that change the number of columns of the matrix. They further establish a connection between the quality of the low-rank approximation and the quality of the subspace.

The analyses in [BER04, BES05] target vectors rather than block methods, for eigenvalues and invariant subspaces of non-Hermitian matrices, with a concern for restarting. The block methods in [LZ15, Saa80] are Lanczos methods for Hermitian eigenvalue problems, and the analyses exploit the (block) tridiagonal structure resulting from recursions. Although singular value problems with block methods are considered in [BR05] and in [BR06], the Krylov spaces are different and the focus is on algorithmic issues of augmenting and restarting the Lanczos process, rather than subspace distances. Krylov spaces For the solution of ill-posed least squares problems via LSQR, [Jia17] analyzes the accuracy of a regularized solution, by bounding the sine between $\mathcal{K}_j(\mathbf{A}^\top\mathbf{A}, \mathbf{A}^\top\mathbf{v})$ and a dominant right singular vector space; however all singular values must be distinct. In the context of low-rank approximations, [SZ00] proposed a Lanczos bidiagonalization with one-sided reorthogonalization.

Our contributions Our work on subspace computations is motivated by the probabilistic approach for low-rank approximations via block Krylov spaces of [MM15] and the standard Lanczos convergence analysis [Saa11, Section 6.6] combined with optimal low-rank approximations via least squares problems [BDMI11, BDMI14] which is also our innovative feature. We present structural, deterministic bounds on the quality of the subspaces for general starting guesses. Our results are based on the following assumptions:

1. The block Krylov spaces have maximal dimension.

2. The analysis assumes exact arithmetic and does not address the implementation of numerically stable recursions.

More specifically, we derive two main results for the problems of Definition 4. Both results leverage the gap amplifying polynomials $\phi(x)$ of degree $2q+1$ with odd powers (see, Section 6.1), a starting guess $\mathbf{X} \in \mathbb{R}^{n\times s}$ (see, Section 6.1), and the crucial assumption that rank $\left(\mathbf{V}_{\mathrm{k}}^\top \mathbf{X}\right) = \mathrm{k}$, where $\mathbf{V}_k$ are the top-k right singular vectors of $\mathbf{A}$. Let $\mathbf{\Sigma}_k$, $\mathbf{\Sigma}_{k,\perp}$ be the diagonal matrices of the polynomial $\phi(x)$ applied on the top-k and the bottom-$(n-k)$ singular values of $\mathbf{A}$, respectively. Then

1. Our **first result** (see, Section 6.2), bounds the distance between the block Krylov subspace, $\mathcal{K}_q$ and the space spanned by the top-k left singular vectors of $\mathbf{A}$, $\mathbf{U}_k$.

$$\|\sin\mathbf{\Theta}(\mathcal{K}_q, \mathbf{U}_k)\|_{2,F} \quad \leq \quad \|\phi(\mathbf{\Sigma}_{k,\perp})\|_2 \, \|\phi(\mathbf{\Sigma}_k)^{-1}\|_2 \, \|\mathbf{V}_{k,\perp}^\top \mathbf{X}(\mathbf{V}_k^\top \mathbf{X})^\dagger\|_{2,F}.$$

   The dependence on the singular gap between the k-th and the $(k+1)$-st singular value of $\mathbf{A}$ is clear by the appearance of the term $\|\phi(\mathbf{\Sigma}_{k,\perp})\|_2 \, \|\phi(\mathbf{\Sigma}_k)^{-1}\|_2$ which is the actual singular gap amplified.

2. Our **second result** (see, Section 6.3), bounds the quality of the approximation of $\mathbf{U}_k$ from the block Krylov space, $\mathcal{K}_q$. Let $\hat{\mathbf{U}}_k$ denote the approximation to the top-k left singular vectors of $\mathbf{A}$. Then,

$$\|\mathbf{A} - \hat{\mathbf{U}}_k\hat{\mathbf{U}}_k^\top\mathbf{A}\|_{2,F} \quad \leq \quad \|\mathbf{A} - \mathbf{U}_k\mathbf{U}_k^\top\mathbf{A}\|_{2,F} + \|\phi(\mathbf{\Sigma}_{k,\perp})\|_2 \, \|\mathbf{V}_{k,\perp}^\top \mathbf{X}(\mathbf{V}_k^\top \mathbf{X})^\dagger\|_{2,F}.$$

In this case it is apparent, by the term $\|\phi(\mathbf{\Sigma}_{k,\perp})\|_2$, that the bound depends on the $k+1$-st singular value and not on the singular gap as observed at the previous result.

Both bounds have a dependency on the starting guess $\mathbf{X}$ (see, Section 6.5). This is obvious by the appearance of the term $\|\mathbf{V}_{k,\perp}^\top \mathbf{X}(\mathbf{V}_k^\top \mathbf{X})^\dagger\|_{2,F}$ in both bounds, a term that measures the quality of the starting guess.

2 PRELIMINARIES

This chapter introduces basic notation used through out the dissertation. It further includes summary of the necessary background on linear algebra (see, Section 2.2), polynomial approximation theory (see, Section 2.3) probability theory (see, Section 2.4) and RandNLA tools (see, Section 2.5) that is required by the algorithms and technical proofs presented later.

2.1 Basic notation

We will be using bold uppercase Latin letters, $\mathbf{A}, \mathbf{B}, \ldots$, and bold lowercase Latin letters, $\mathbf{a}, \mathbf{b}, \ldots$ to represent matrices and vectors respectively. Scalars will be denoted by either lowercase Latin letters $a, b, \ldots$ or Greek letters, $\alpha, \beta, \ldots$. $\mathbf{I}_n$ will represent the identity matrix of size $n \times n$, while $\mathbf{1}_n$, $\mathbf{0}_n$ will represent the all ones and all zeros vectors of size n, respectively. Similarly, $\mathbf{1}_{n\times n}$, $\mathbf{0}_{n\times n}$ will represent the all zeros and all ones matrices of size $n \times n$, respectively. The (i, j)-th element of a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ will be denoted as $\mathbf{A}_{i,j}$ where $i = 1, \ldots, m$ and $j = 1, \ldots, n$.

Given a square matrix, $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{Tr}(\mathbf{A})$, $\mathbf{det}(\mathbf{A})$ will denote the trace and the determinant of $\mathbf{A}$ respectively. λ_i, σ_i will denote the i-th eigenvalue and i-th singular value of $\mathbf{A}$, respectively. The rank of $\mathbf{A}$ will be denoted by $\operatorname{rank}(\mathbf{A})$ and its condition number by $\kappa(\mathbf{A})$.

Unless defined as a space, we will use blackboard bold uppercase Latin letters to denote random variables, e.g. $\mathbb{X}$, will denote the random variable X. We will use the notation $\mathbf{E}[\cdot]$ and $\mathbf{Var}[\cdot]$ to denote the expectation and the variance of a random variable, respectively. Given an event $\mathcal{E}$ in the sample space Ω, $\mathbf{Pr}[\mathcal{E}]$ will denote the probability of $\mathcal{E}$ to happen.

Finally, we will define the k-th derivative of a function f, with $0 \neq k > 2$ as $f^{(k)}(\cdot)$. The first and second derivatives of f will be denoted as $f'(\cdot)$ and $f''(\cdot)$, respectively.

2.2 Linear algebra basics

This section summarizes basic Linear Algebra background that is useful for the following chapters.

2.2.1 Vector and matrix norms

A norm can be seen as the generalization of the absolute value in vector spaces by providing a sense of distance between points in the vector space. Mathematically a norm is briefly defined as the function that maps an object from a vector space to the non-negative real space.

Definition 5 (Vector Norm [GV96])**.** *A vector norm on* $\mathbb{R}^n$ *is a function* $f : \mathbb{R}^n \to \mathbb{R}$ *that satisfies the following properties:*

Positive Semi-Definiteness	$f(\mathbf{x}) \geq 0,\ \mathbf{x} \in \mathbb{R}^n$ *(i.e.* $f(\mathbf{x}) = 0$ *iff* $\mathbf{x} = \mathbf{0}$*)*
Subadditivity	$f(\mathbf{x}+\mathbf{y}) \leq f(\mathbf{x}) + f(\mathbf{y}),\ \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$
Absolute Homogeneity	$f(\alpha\mathbf{x}) = \lvert\alpha\rvert\, f(\mathbf{x}),\ \alpha \in \mathbb{R},\ \mathbf{x} \in \mathbb{R}^n$

A matrix norm is defined similarly[1], by substituting $\mathbb{R}^n$ with $\mathbb{R}^{m\times n}$ and vectors with matrices. This manuscript, unless stated otherwise, considers the class of *p-norms*. The p-norm of a vector $\mathbf{x} \in \mathbb{R}^n$ is defined as:

$$\|\mathbf{x}\|_p = (|x_1|^p + |x_2|^p + \cdots + |x_n|^p)^{\frac{1}{p}}\ ,\ p > 0^2 \tag{2.1}$$

where x_i represents the i-th component of $\mathbf{x}$. The most common vector p-norms are:

1-norm:

$$\|\mathbf{x}\|_1 = |x_1| + |x_2| + \cdots + |x_n|$$

[1]Indeed $\mathbb{R}^{m\times n}$ is isomorphic to $\mathbb{R}^{mn}$.

[2]The zero norm $\|\cdot\|_0$ although referred as a norm it is not an p-norm. That is because it does not satisfy the positive semi-definite property of the vector norms.

2-norm or Eucledian norm:

$$\|\mathbf{x}\|_2 = \left(|x_1|^2 + |x_2|^2 + \cdots + |x_n|^2\right)^{\frac{1}{2}} = \left(\mathbf{x}^\top \mathbf{x}\right)^{\frac{1}{2}}$$

∞-norm:

$$\|\mathbf{x}\|_\infty = \max_{i=1}^{n} |x_i|$$

The p-norm of a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ is defined as:

$$\|\mathbf{A}\|_p = \sup_{\mathbf{x}\neq\mathbf{0}} \frac{\|\mathbf{Ax}\|_p}{\|\mathbf{x}\|_p} \tag{2.2}$$

where $\mathbf{x} \in \mathbb{R}^n$. Common matrix p-norms include:

1-norm:

$$\|\mathbf{A}\|_1 = \max_{1\leq j\leq n} \sum_{i=1}^{m} |a_{ij}|$$

2-norm:

$$\|\mathbf{A}\|_2 = \left(\max_{1\leq j\leq n} \lambda_j(\mathbf{A}^\top\mathbf{A})\right)^{\frac{1}{2}} = \sigma_{\max}$$

∞-norm:

$$\|\mathbf{A}\|_\infty = \max_{1\leq i\leq m} \sum_{j=1}^{n} |a_{ij}|$$

Besides the matrix p-norms a large class of matrix norms that is ubiquitously used throughout the manuscript is the family of Schatten p-norms of a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$. The Schatten p-norm is defined as:

$$\|\mathbf{A}\|_p = \left(\mathbf{Tr}\left(|\mathbf{A}|^p\right)\right)^{\frac{1}{p}} = \left(\sum_{i=1}^{\min\{m,n\}} \sigma_i(\mathbf{A})^p\right)^{\frac{1}{p}}, \tag{2.3}$$

where $\sigma_i(\mathbf{A}),\ i = 1,\ldots,\min\{m,n\}$ is the i-th singular value of $\mathbf{A}$. Commonly used Schatten p-norms are:

[3]Also known as the Nuclear Norm.

Schatten 1-norm (or Trace Norm[3]):

$$\|\mathbf{A}\|_1 = \mathbf{Tr}\left(\mathbf{A}\right) = \sum_{i=1}^{\min\{m,n\}} \sigma_i(\mathbf{A})$$

Schatten 2-norm (or Frobenius Norm)

$$\|\mathbf{A}\|_F = \left(\mathbf{Tr}\left(\mathbf{A}^\top\mathbf{A}\right)\right)^{\frac{1}{2}} = \left(\sum_{i=1}^{\min\{m,n\}} \sigma_i(\mathbf{A})^2\right)^{\frac{1}{2}} = \left(\sum_{i=1}^{m}\sum_{j=1}^{n} |A_{i,j}|^2\right)^{\frac{1}{2}}$$

∞-norm (or Spectral Norm)

$$\|\mathbf{A}\|_\infty = \sigma_{\max}(\mathbf{A})$$

2.2.2 The singular value decomposition (SVD)

Any matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ can be decomposed in the form

$$\mathbf{A} = \mathbf{U}\begin{pmatrix} \mathbf{\Sigma} \\ \mathbf{0}_{(m-n)\times n} \end{pmatrix}\mathbf{V}^\top, \text{ if } m \geq n \tag{2.4}$$

or

$$\mathbf{A} = \mathbf{U}\begin{pmatrix} \mathbf{\Sigma} & \mathbf{0}_{m\times(n-m)} \end{pmatrix}\mathbf{V}^\top, \text{ if } m < n \tag{2.5}$$

where $\mathbf{U} \in \mathbb{R}^{m\times m}$ and $\mathbf{V} \in \mathbb{R}^{n\times n}$ are the orthogonal matrices of the left and right singular vectors of $\mathbf{A}$ respectively. $\mathbf{\Sigma} \in \mathbb{R}^{q\times q}$, with $q = \min\{m, n\}$ is the diagonal matrix of the singular values of $\mathbf{A}$. Eqn. (2.4) and eqn. (2.5) define the **full SVD** of $\mathbf{A}$. Given $r = \text{rank}\left(\mathbf{A}\right)$ we define the **thin SVD** of $\mathbf{A}$ as:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top, \tag{2.6}$$

where $\mathbf{U} \in \mathbb{R}^{m\times r}$ and $\mathbf{V} \in \mathbb{R}^{n\times r}$ are the orthonormal matrices of the left and right singular vectors associated with the r non-zero singular values residing in the diagonal matrix $\mathbf{\Sigma} \in \mathbb{R}^{r\times r}$.

Let $\mathbf{A} = \mathbf{U\Sigma V}^\top$ be the full SVD of an $m \times n$ matrix $\mathbf{A}$, so that $\mathbf{U}$ and $\mathbf{V}$ are the orthogonal matrices of the singular vectors and $\mathbf{\Sigma}$ is the diagonal matrix of the singular values. Then given a rank parameter $k \leq \min\{m, n\}$, **the best rank-k** approximation of $\mathbf{A}$, $\mathbf{A}_k$, is defined as:

$$\mathbf{A}_k = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^\top,$$

where $\mathbf{U}_k$ and $\mathbf{V}_k$ are the orthonormal matrices of the top-k left and right singular vectors respectively and $\mathbf{\Sigma}_k$ is the diagonal matrix of the top-k singular values.

Theorem 1 (Eckart–Young–Mirsky Theorem). *Given a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ and rank $r > k$. The solution of the approximation problem:*

$$\min_{\text{rank}(\mathbf{Z})=\text{k}} \|\mathbf{A} - \mathbf{Z}\|_{2,F} \tag{2.7}$$

is given by

$$\mathbf{Z} = \mathbf{A}_k = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^\top$$

The minimum value of (2.7)

$$\|\mathbf{A} - \mathbf{A}_k\|_{2,F} = \sigma_{k+1}$$

2.2.3 Generalized Moore-Penrose pseudo-inverse

Given a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ we define the generalized Moore-Penrose Pseudoinverse, denoted by $\mathbf{A}^\dagger$, as the unique matrix that satisfies the following properties:

1. $\mathbf{A}\mathbf{A}^\dagger\mathbf{A} = \mathbf{A}$

2. $\mathbf{A}^\dagger\mathbf{A}\mathbf{A}^\dagger = \mathbf{A}^\dagger$

3. $(\mathbf{A}\mathbf{A}^{\dagger})^{\top} = \mathbf{A}\mathbf{A}^{\dagger}$

4. $(\mathbf{A}^{\dagger}\mathbf{A})^{\top} = \mathbf{A}^{\dagger}\mathbf{A}$

Algebraically, the Moore-Penrose Pseudoinverse is computed using the Singular Value Decomposition:

$$\mathbf{A}^{\dagger} = \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^{\top}$$

where $\mathbf{V} \in \mathbb{R}^{n\times n}$ and $\mathbf{U} \in \mathbb{R}^{m\times m}$ are the orthogonal matrices of the right and left singular vectors of $\mathbf{A}$ respectively, and $\boldsymbol{\Sigma} \in \mathbb{R}^{m\times n}$ is the diagonal matrix of the singular values of $\mathbf{A}$.

Theorem 2 (**[Bjö15, Theorem 2.2.3]).** *Given two matrices* $\mathbf{A} \in \mathbb{R}^{m\times k}$ *and* $\mathbf{B} \in \mathbb{R}^{k\times n}$*, if* $\operatorname{rank}(\mathbf{A}) = \operatorname{rank}(\mathbf{B}) = \mathrm{k}$*, then*

$$(\mathbf{A}\mathbf{B})^{\dagger} = \mathbf{B}^{\dagger}\mathbf{A}^{\dagger}. \tag{2.8}$$

The principal component analysis (PCA) The principal component analysis is a popular method in data mining and machine learning. It provides information about the direction towards which the variance is large. PCA is a simple yet powerful tool that reveals the underlying structure of the data by emerging the SVD of the covariance matrix. In this dissertation we are more interested in expressing PCA as an optimization problem (see, Problem 6).

Definition 6 (The PCA Problem)**.** *Given an input matrix* $\mathbf{X} \in \mathbb{R}^{m\times n}$ *with covariance matrix* $\mathbf{A} = \mathbf{X}^{\top}\mathbf{X} \in \mathbb{R}^{n\times n}$*, we seek to find a vector* $\mathbf{x}$ *that is the solution to:*

$$\mathcal{Z}_{opt} = \max_{\mathbf{x} \in \mathbb{R}^n} \quad \mathbf{x}^{\top}\mathbf{A}\,\mathbf{x} \tag{2.9a}$$

$$\text{s.t.} \quad \|\mathbf{x}\|_2 = 1 \tag{2.9b}$$

In general we assume that the input matrix $\mathbf{X}$ is mean-centered as each column is assumed to be a random variable with zero mean. The solution of problem (2.9) is the dominant eigenvectors of the covariance matrix $\mathbf{A}$ and the corresponding eigenvalue is the value of $\mathcal{Z}_{opt}$. The top eigenvector of $\mathbf{A}$ is further called the first principal component. Principal components of decreasing order can be found using **deflation**; e.g. the k-th principal component can be found, by first computing the deflated matrix

$$\mathbf{A}_k = \mathbf{A} - \sum_{m=1}^{k-1} \mathcal{Z}_{opt_m} \mathbf{x}_m \mathbf{x}_m^\top,$$

where $\mathbf{x}_m$ is the m-th principal component and $\mathcal{Z}_{opt_m}$ the corresponding eigenvalue and then solving the problem (2.10):

$$\mathcal{Z}_{opt_k} = \max_{\mathbf{x}_k \in \mathbb{R}^n} \quad \mathbf{x}_k^\top \mathbf{A}_k \, \mathbf{x}_k \tag{2.10a}$$

$$\text{s.t.} \qquad \|\mathbf{x}_k\|_2 = 1 \tag{2.10b}$$

The final step is to enforce orthogonality between the k principal components $\mathbf{x}_1, \ldots, \mathbf{x}_{k-1}, \mathbf{x}_k$. This can be done by invoking a procedure like Gram-Schmidt.

2.2.4 Matrix inequalities

In this section we review important matrix inequalities that will appear in many of our technical proofs.

Triangle inequality The triangle inequality bounds the distance between two vectors. Given two vectors $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^n$ then for any invariant norm $\|\cdot\|$

$$\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|.$$

Strong sub-multiplicativity The strong sub-multiplicativity of the Frobenius norm [HJ91, page 211] associates the Frobenius norm of the product of two matrices with the Frobenius norm of the one matrix and the spectral norm of the other. Given the matrices $\mathbf{A} \in \mathbb{R}^{m \times k}$ and $\mathbf{B} \in \mathbb{R}^{k \times n}$,

$$\begin{aligned} \|\mathbf{AB}\|_F &\leq \|\mathbf{A}\|_2 \|\mathbf{B}\|_F \\ \|\mathbf{AB}\|_F &\leq \|\mathbf{A}\|_F \|\mathbf{B}\|_2. \end{aligned}$$

Matrix Pythagoras Lemma 1 is the matrix analog of the well known Pythagorean theorem.

Lemma 1 (Matrix Pythagoras)**.** *Given the matrices* $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{m \times n}$*. If* $\mathbf{A}^\top \mathbf{B} = \mathbf{0}_{n \times n}$ *then*

$$\|\mathbf{A} + \mathbf{B}\|_F^2 = \|\mathbf{A}\|_F^2 + \|\mathbf{B}\|_F^2.$$

2.2.5 Positive (semi-)definiteness of matrices

Symmetric or Hermitian positive (semi-)definite matrices appear ubiquitously in linear algebra. They are structured matrices with special eigenvalue properties that are frequently desired in many applications. Recall, that a symmetric (or Hermitian) matrix has only real eigenvalues.

Definition 7 (Symmetric Positive Definite Matrix (SPD))**.** *A symmetric matrix* $\mathbf{A} \in \mathbb{R}^{n \times n}$ *is also positive definite if for all* ***non-zero*** *vectors* $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x}^\top \mathbf{A} \mathbf{x} > 0$.

Definition 8 (Symmetric Positive semi-Definite Matrix (SPSD))**.** *A symmetric matrix* $\mathbf{A} \in \mathbb{R}^{n \times n}$ *is also positive semi-definite if for all vectors* $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x}^\top \mathbf{A} \mathbf{x} \geq 0$.

The analogs of Definitions 7 and 8 in the complex space are given in Definitions 9 and 10.

Definition 9 (Hermitian Positive Definite Matrix (HPD))**.** *A Hermitian matrix* $\mathbf{A} \in \mathbb{C}^{n \times n}$ *is also positive definite if for all* ***non-zero*** *vectors* $\mathbf{x} \in \mathbb{C}^n$, $\mathbf{x}^* \mathbf{A} \mathbf{x} > 0$.

Definition 10 (Hermitian Positive semi-Definite Matrix (HPSD))**.** *A Hermitian matrix* $\mathbf{A} \in \mathbb{C}^{n\times n}$ *is also positive semi-definite if for all vectors* $\mathbf{x} \in \mathbb{C}^n$*,* $\mathbf{x}^*\mathbf{A}\mathbf{x} \geq 0$.

2.2.6 Matrix functions

Given a matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$ and a scalar function f we can define the $n \times n$ matrix $f(\mathbf{A})$ as the generalization of f in a multidimensional space. There exist multiple definitions of $f(\mathbf{A})$; we refer the interested reader to [Hig08, Chapter 1]. In this dissertation we will use a definition that is closely connected with the spectrum of $\mathbf{A}$.

Definition 11 (The Jordan Canonical Form [Hig08, Section, 1.2.1])**.** *Any matrix* $\mathbf{A} \in \mathbb{C}^{n\times n}$ *can be expressed in the Jordan canonical form:*

$$\mathbf{Z}^{-1}\mathbf{A}\mathbf{Z} = \mathbf{J} = \operatorname{diag}(\mathbf{J}_1,\ \mathbf{J}_2,\ \ldots,\ \mathbf{J}_p) \tag{2.11}$$

where, $\mathbf{J} \in n \times n$ *is the unique Jordan matrix*[4] *and* $\mathbf{Z}$ *is a nonsingular matrix; the submatrix* $\mathbf{J}_k$ *is referred as the* k*-th Jordan block and is defined as:*

$$\mathbf{J}_k = \mathbf{J}_k(\lambda_k) = \begin{bmatrix} \lambda_k & 1 & & \\ & \lambda_k & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_k \end{bmatrix} \in \mathbb{C}^{m_k \times m_k}, \tag{2.12}$$

where λ_k *is the* k*-th eigenvalue of* $\mathbf{A}$ *and* $m_1 + m_2 + \cdots + m_p = n$.

Now, if $\mathbf{A}$ has s distinct eigenvalues, $\lambda_1, \lambda_2, \ldots, \lambda_s$ and the order of the largest Jordan block in which λ_i appears is n_i, then a matrix function is said to be defined on the spectrum of $\mathbf{A}$ if the values $f^{(j)}(\lambda_i),\ j = 0, \ldots, n_i - 1\ \ i = 1, \ldots, s$ exist.

[4]The Jordan matrix is unique up to the ordering of the Jordan blocks [Hig08].

Definition 12 (Matrix Function via Jordan Canonical Form [Hig08, Definition 1.2])**.** *Let f be defined on the spectrum of $\mathbf{A} \in \mathbb{C}^{n\times n}$ and let $\mathbf{A}$ have the Jordan canonical form of eqn.* (2.11)*. Then*

$$f(\mathbf{A}) = \mathbf{Z}f(\mathbf{J})\mathbf{Z}^{-1} = \mathbf{Z}\,\mathrm{diag}\left(f(\mathbf{J}_k)\right)\,\mathbf{Z}^{-1}, \tag{2.13}$$

where

$$f(\mathbf{J}_k) = \begin{bmatrix} f(\lambda_k) & f'(\lambda_k) & \dots & \frac{f^{(m_k-1)}(\lambda_k)}{(m_k-1)!} \\ & f(\lambda_k) & \ddots & \vdots \\ & & \ddots & f'(\lambda_k) \\ & & & f(\lambda_k) \end{bmatrix} \tag{2.14}$$

If $\mathbf{A}$ is diagonalizable then the Jordan canonical form reduces to the eigendecomposition $\mathbf{A} = \mathbf{Z}\mathbf{D}\mathbf{Z}^{-1}$ where $\mathbf{D} \in \mathbb{C}^{n\times n}$ is the diagonal matrix of the eigenvalues of $\mathbf{A}$, and $\mathbf{Z} \in \mathbb{C}^{n\times n}$ is the unitary matrix of the eigenvectors of $\mathbf{A}$. Then eqn. (2.13) yields

$$f(\mathbf{A}) = \mathbf{Z}f(\mathbf{D})\mathbf{Z}^{-1} = \mathbf{Z}\,\mathrm{diag}\left(f(\lambda_i)\right)\mathbf{Z}^{-1}$$

In this dissertation we will frequently refer to logarithms of positive definitive matrices. Recall that a symmetric (or Hermitian) matrix, $\mathbf{A}$, has all its eigenvalues in the real space. If in addition, $\mathbf{A}$ is positive definite then it is always diagonalizable with eigenvalues strictly greater than zero. Thus $f(x) = \log(x)$ can be defined on the spectrum of $\mathbf{A}$ and $\log[\mathbf{A}]$[5] is defined as in Definition 13.

Definition 13 (Logarithm of a Positive Definite Matrix)**.** *Given an SPD matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, we define the logarithm of $\mathbf{A}$, denoted by the $n \times n$ matrix $\log[\mathbf{A}]$, as :*

$$\log[\mathbf{A}] = \mathbf{U}\mathbf{D}\mathbf{U}^{\top},$$

[5]Since there are not negative eigenvalues, then $\mathbf{A}$ has a **unique** principal logarithm (see, [Hig08, Theorem 1.31]).

where $\mathbf{U} \in \mathbb{R}^{n\times n}$ *is the orthogonal matrix of the eigenvectors of* $\mathbf{A}$ *and* $\mathbf{D} \in \mathbb{R}^{n\times n}$ *is the diagonal matrix*

$$\operatorname{diag}\left(\log(\lambda_1(\mathbf{A})), \quad \log(\lambda_2(\mathbf{A})), \quad \ldots, \quad \log(\lambda_n(\mathbf{A})\right).$$

Similarly, given an HPD $\mathbf{A} \in \mathbb{C}^{n\times n}$*, we define the logarithm of* $\mathbf{A}$*, denoted by the* $n \times n$ *matrix* $\log[\mathbf{A}]$*, as :*

$$\log[\mathbf{A}] = \mathbf{U}\mathbf{D}\mathbf{U}^*,$$

where $\mathbf{U} \in \mathbb{C}^{n\times n}$ *is the unitary matrix of the eigenvectors of* $\mathbf{A}$ *and* $\mathbf{D} \in \mathbb{R}^{n\times n}$ *is the diagonal matrix*

$$\operatorname{diag}\left(\log(\lambda_1(\mathbf{A}), \quad \log(\lambda_2(\mathbf{A})), \quad \ldots, \quad \log(\lambda_n(\mathbf{A})\right).$$

We should note that the logarithm of a positive semi-definite matrix can only be defined under certain assumptions e.g. if we assume that $\log(0) = 0$.

2.3 Function approximation via polynomials

In this section we describe two of the most popular polynomials that are broadly used for the approximation of continuous functions. The Taylor polynomials are described in Section 2.3.1 and the Chebyshev polynomials are described in Section 2.3.2.

2.3.1 Taylor series

The Taylor series first formulated by James Gregory but later introduced by Brook Taylor in 1715 is a powerful mathematical tool and is broadly used in approximation theory. The Taylor series represent a function as the infinite combination of terms that are associated with the derivatives of the function on a certain point.

Definition 14 (Taylor Series [HHW18, Section 10.8]). *Let f be a function with derivatives of all orders throughout some interval containing α as an interior point. Then the Taylor series generated by f at $x = \alpha$ is*

$$\sum_{k=0}^{\infty} \frac{f^{(k)}(\alpha)}{k!}(x-\alpha) = f(\alpha) + f'(\alpha)(x-\alpha) + \frac{f''(\alpha)}{2}(x-\alpha)^2 + \cdots + \frac{f^{(n)}(\alpha)}{n!}(x-\alpha)^n + \dots$$

The easiest way to approximate a function f that is at least m times differentiable at a point α is by using the Taylor polynomial of degree n which consists of the first n terms of the Taylor series of f on $x = \alpha$:

Definition 15 (Taylor polynomials [HHW18, Section 10.8]). *Let f be a function with derivatives of order up to n in some interval containing α as an interior point. Then for any integer from $1, \dots, m$ the Taylor polynomial of order n generated by f at $x = \alpha$ is*

$$\sum_{k=0}^{n} \frac{f^{(k)}(\alpha)}{k!}(x-\alpha) = f(\alpha) + f'(\alpha)(x-\alpha) + \frac{f''(\alpha)}{2}(x-\alpha)^2 + \cdots + \frac{f^{(n)}(\alpha)}{n!}(x-\alpha)^n$$

Using Definition 15 and assuming that $m \leq n$ the m-th order approximation of f follows from:

$$f(x) \approx \sum_{k=0}^{m} \frac{f^{(k)}(\alpha)}{k!}(x-\alpha).$$

The approximation error is proportional to the reminder of the series

$$R_m(x) = \mathcal{O}\left(|x-\alpha|^{m+1}\right).$$

We further define the Taylor expansion of $\log(1-x)$ at zero[6] and its matrix analog. Both definitions will be useful in the chapters that follow.

[6]This is also called the McLaurin expansion.

Definition 16 (**Taylor expansion of** $\log(\mathbf{1} - \mathbf{x})$)**.** *Let x be a scalar variable that satisfies $|x| < 1$. Then:*

$$\log(1 - x) = -\sum_{k=1}^{\infty} \frac{x^k}{k}.$$

A matrix generalization of Definition 16 is given in Lemma 2.

Lemma 2 (**Taylor expansion of** $\log[\mathbf{I_n} - \mathbf{A}]$)**.** *Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be a symmetric matrix whose eigenvalues all lie in the interval $(-1, 1)$. Then,*

$$\log[\mathbf{I}_n - \mathbf{A}] = -\sum_{k=1}^{\infty} \frac{\mathbf{A}^k}{k}.$$

2.3.2 Chebyshev polynomials

The Chebyshev polynomials were introduced in 1854 by Pafnuty Chebyshev and frequently appear in numerical analysis and approximation theory. The Chebyshev polynomials is a family of orthogonal polynomials with the largest possible leading coefficients whose absolute values for inputs from $[-1, 1]$ are bounded by 1. In approximation theory the Chebyshev polynomials are widely used in polynomial interpolation as the computed interpolants (built utilizing the Chebyshev nodes) minimize the Runge phenomenon and provide an excellent approximation to the actual polynomial. Additionally, the Chebyshev polynomials provide the best, in terms of the maximum norm, polynomial approximation to a continuous function.

Definition 17 (Chebyshev Polynomials [Tre12, eqn. (3.1)])**.** *Given a variable x, a function f defined in $[-1, 1]$ and a variable z ranging over the unit circle in the complex plane,*

$$f(x) \approx \sum_{k=0}^{n} \alpha_k T_k(x), \tag{2.15}$$

where α_k is the coefficient of T_k which is the k-th Chebyshev polynomial defined as the real part of the function z^k on the unit circle

$$x = \frac{1}{2}(z + z^{-1}) = \cos\theta, \quad \theta = \cos^{-1} x$$

$$T_k(x) = \frac{1}{2}(z^k + z^{-k}) = \cos(k\theta).$$

The construction of Chebyshev polynomials is a recursive procedure distinguished in two kinds:

1. **Chebyshev polynomials of the first kind:**

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_{k+1} &= 2xT_k(x) - T_{k-1}(x) \end{aligned}$$

2. **Chebyshev polynomials of the second kind:**

$$\begin{aligned} U_0(x) &= 1 \\ U_1(x) &= 2x \\ U_{k+1} &= 2xU_k(x) - U_{k-1}(x) \end{aligned}$$

In this dissertations, unless stated otherwise, we will refer to the Chebyshev polynomials of the first kind. Finally, Theorem 3 describes the Chebyshev series of a function $f(x)$ with special characteristics.

Theorem 3 (Chebyshev series [Tre12, Theorem (3.1)]). *If f is a Lipschitz continuous function on $[-1, 1]$, then it has a unique representation as a Chebyshev series,*

$$f(x) = \sum_{k=0}^{\infty} \alpha_k T_k(x),$$

which is absolutely and uniformly convergent. The coefficients are given for $k \geq 1$ by the formula

$$\alpha_k = \frac{2}{\pi} \int_{-1}^{1} \frac{f(x)T_k(x)}{\sqrt{1-x^2}} dx,$$

and for $k = 0$ by the formula

$$\alpha_0 = \frac{1}{\pi} \int_{-1}^{1} \frac{f(x)T_k(x)}{\sqrt{1-x^2}} dx.$$

The Clenshaw algorithm The Clenshaw algorithm implements a recursive formula that fastly computes linear combinations of Chebyshev polynomials like the one of eqn. (2.15). In this section we briefly sketch Clenshaw's algorithm to evaluate linear combinations of Chebyshev polynomials with matrix inputs. Clenshaw's algorithm is a recursive approach with base cases $b_{m+2}(x) = b_{m+1}(x) = 0$ and the recursive step (for $k = m, m-1, \ldots, 0$):

$$b_k(x) = \alpha_k + 2x b_{k+1}(x) - b_{k+2}(x), \tag{2.16}$$

where α_k is the k-th coefficient of the linear combination. Then, the linear combination of Chebyshev polynomials, $f_m(x)$ can be computed using the formula:

$$f_m(x) = \frac{1}{2} \left(\alpha_0 + b_0(x) - b_2(x) \right). \tag{2.17}$$

Using the mapping $x \to 2(x/u) - 1$, eqn. (2.16) becomes

$$b_k(x) = \alpha_k + 2 \left(\frac{2}{u} x - 1 \right) b_{k+1}(x) - b_{k+2}(x). \tag{2.18}$$

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

$$\mathbf{B}_k(\mathbf{R}) = \alpha_k \mathbf{I}_n + 2 \left(\frac{2}{u} \mathbf{R} - \mathbf{I}_n \right) \mathbf{B}_{k+1}(\mathbf{R}) - \mathbf{B}_{k+2}(\mathbf{R}) \tag{2.19}$$

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

$$f_m(\mathbf{R}) = \frac{1}{2} \left(\alpha_0 \mathbf{I}_n + \mathbf{B}_0(\mathbf{R}) - \mathbf{B}_2(\mathbf{R}) \right). \tag{2.20}$$

Using the matrix version of Clenshaw's algorithm, we can now provide a modified Clenshaw scheme to compute $\mathbf{g}^\top f_m(\mathbf{R})\mathbf{g}$, where $\mathbf{g} \in \mathbb{R}^n$. First, we right multiply eqn. (2.19) by $\mathbf{g}$,

$$\begin{aligned} \mathbf{B}_k(\mathbf{R})\mathbf{g} &= \alpha_k \mathbf{I}_n \mathbf{g} + 2\left(\frac{2}{u}\mathbf{R} - \mathbf{I}_n\right)\mathbf{B}_{k+1}(\mathbf{R})\mathbf{g} - \mathbf{B}_{k+2}(\mathbf{R})\mathbf{g}, \\ \mathbf{y}_k &= \alpha_k \mathbf{g} + 2\left(\frac{2}{u}\mathbf{R} - \mathbf{I}_n\right)\mathbf{y}_{k+1} - \mathbf{y}_{k+2}. \end{aligned} \tag{2.21}$$

Eqn. (2.21) follows by substituting $\mathbf{y}_i = \mathbf{B}_i(\mathbf{R})\mathbf{g}$. Multiplying the base cases by $\mathbf{g}$, we get $\mathbf{y}_{m+2} = \mathbf{y}_{m+1} = \mathbf{0}$ and the final sum becomes

$$\mathbf{g}^\top f_m(\mathbf{R})\mathbf{g} = \frac{1}{2}\left(\alpha_0(\mathbf{g}^\top\mathbf{g}) + \mathbf{g}^\top(\mathbf{y}_0 - \mathbf{y}_2)\right). \tag{2.22}$$

Algorithm 1 summarizes all the above.

Algorithm 1 Clenshaw's algorithm to compute $\mathbf{g}^\top f_m(\mathbf{R})\mathbf{g}$.

```
Input: $\alpha_i$, $i = 0, \ldots, m$, $\mathbf{R} \in \mathbb{R}^{n \times n}$, $\mathbf{g} \in \mathbb{R}^n$
Output: $\mathbf{g}^\top f_m(\mathbf{R})\mathbf{g}$.
1: Set $\mathbf{y}_{m+2} = \mathbf{y}_{m+1} = \mathbf{0}$
2: for $k = m, m-1, \ldots, 0$ do
3:     $\mathbf{y}_k = \alpha_k \mathbf{g} + \frac{4}{u}\mathbf{R}\mathbf{y}_{k+1} - 2\mathbf{y}_{k+1} - \mathbf{y}_{k+2}$
4: end for
5: return $\mathbf{g}^\top f_m(\mathbf{R})\mathbf{g} = \frac{1}{2}\left(\alpha_0(\mathbf{g}^\top\mathbf{g}) + \mathbf{g}^\top(\mathbf{y}_0 - \mathbf{y}_2)\right)$
```

2.4 Probability theory basics

This section summarizes the basics of probability theory that will frequently appear in our probabilistic analyses.

Definition 18 (Union Bound). *Given n events $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_n$ with $\mathcal{E}_1 \subseteq \mathcal{E}_2 \subseteq \cdots \subseteq \mathcal{E}_n$ then*

$$\mathbf{Pr}\left[\mathcal{E}_1 \cup \mathcal{E}_2 \cup \cdots \cup \mathcal{E}_n\right] = \mathbf{Pr}\left[\mathcal{E}_1\right] + \mathbf{Pr}\left[\mathcal{E}_2\right] + \cdots + \mathbf{Pr}\left[\mathcal{E}_n\right].$$

Definition 19 (Linearity of Expectation). *Given the random variables* $(\mathbb{X}_1, \mathbb{X}_2, \ldots, \mathbb{X}_n)$ *over the sample space* $\Omega_1 \times \Omega_2 \times \cdots \times \Omega_n \subseteq \mathbb{R}$ *then*

$$\mathbf{E}\left[\mathbb{X}_1 + \mathbb{X}_2 + \cdots + \mathbb{X}_n\right] = \mathbf{E}\left[\mathbb{X}_1\right] + \mathbf{E}\left[\mathbb{X}_2\right] + \cdots + \mathbf{E}\left[\mathbb{X}_n\right].$$

Definition 20 (Markov's Inequality). *Given the non-negative random variable* $\mathbb{X}$ *over the sample space* $\Omega \subseteq \mathbb{R}^7$ *with expected value* $\mathbf{E}[\mathbb{X}] = \mu$ *and a scalar* $\lambda > 0$, *it holds:*

$$\mathbf{Pr}\left[\mathbb{X} \geq \lambda\mu\right] \leq \frac{1}{\lambda}$$

or given $\tau = \lambda\mu > 0$,

$$\mathbf{Pr}\left[\mathbb{X} \geq \tau\right] \leq \frac{\mu}{\tau}.$$

2.5 Randomized numerical linear algebra (RandNLA) tools

This section presents randomized numerical linear algebra tools and the corresponding probabilistic bound that will be useful in the design and analysis of our algorithms.

2.5.1 Power method with provable bounds

The power-method is an iterative method used to obtain an accurate estimate of the largest eigenvalue of a matrix $\mathbf{A}$. Briefly, $\mathbf{A}$ is repeatedly post-multiplied with a vector (see, step 4 of Algorithm 2) that after a sufficient number of iterations converges to the dominant eigenvector of $\mathbf{A}$ (i.e. the eigenvector associated with the largest eigenvalue of $\mathbf{A}$). Then, the Rayleigh quotient is used to obtain the approximation to the largest eigenvalue, $\lambda_1(\mathbf{A})$ (see, step 6 of Algorithm 2). An important part of each iterative method, is its initialization. In the case of power method, it is important to initialize $\mathbf{0}_n$, the initial approximation of the dominant eigenvector, $\mathbf{1}_n$, as good as possible. Multiple initializations have been suggested (e.g.

7Mathematically denoted as: $\mathbb{X} \in \Omega \subseteq \mathbb{R}_{\geq 0}$.

0, **1**, Gaussian vectors, e.t.c.), but in our case we will be using Rademacher vectors (see, step 2 of Algorithm 2). This is a choice that has been used in [Tre11] to enable further theoretical analysis of the power method on SPSD matrices that results in probabilistic guarantees useful for our results in Chapters 3 and 4.

Algorithm 2 Power method, repeated q times.

```
Input: SPSD matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, integers $q,\ t > 0$
Output: $\tilde{\lambda}_1(\mathbf{A})$, the approximation to the largest eigenvalue, $\lambda_1(\mathbf{A})$ and $\tilde{\mathbf{u}}_1$, the
    approximation to the dominant eigenvector $\mathbf{u}_1$ of $\mathbf{A}$.
1: for $j = 1, \ldots, q$ do
2:     Pick uniformly at random a vector $\mathbf{x}_0^j \in \{+1, -1\}^n$
3:     for $i = 1, \ldots, t$ do
4:         $\mathbf{x}_i^j = \mathbf{A} \cdot \mathbf{x}_{i-1}^j$
5:     end for
6:     $\tilde{\lambda}_1^j(\mathbf{A}) = \frac{{\mathbf{x}_t^j}^\top \mathbf{A} \mathbf{x}_t^j}{{\mathbf{x}_t^j}^\top \mathbf{x}_t^j}$
7: end for
8: return $\tilde{\lambda}_1(\mathbf{A}) = \max_{j=1\ldots q} \tilde{\lambda}_1^j(\mathbf{A})$ and $\tilde{\mathbf{u}}_1 = \mathbf{x}_t^j$
```

Algorithm 2 is a randomized algorithm specifically for SPSD matrices, and a slight modification of *Algorithm Power* of [Tre11] to (i) boost the success probability (by repeating the algorithm q times) and (ii) to return the approximation to the dominant eigenvalue of $\mathbf{A}$. Lemma 3 (see, [Tre11] for a proof) argues that any $\tilde{\lambda}_1^j(\mathbf{A})$ is close to $\lambda_1(\mathbf{A})$:

Lemma 3 (**[Tre11, Theorem 1]**). *For any fixed $j = 1 \ldots q$, and for any $t > 0$, $\varepsilon > 0$, with probability at least $3/16$,*

$$\frac{(1-\varepsilon)}{1 + 4n(1-\varepsilon)^{2t}} \lambda_1(\mathbf{A}) \leq \frac{{\mathbf{x}_t^j}^\top \mathbf{A} \mathbf{x}_t^j}{{\mathbf{x}_t^j}^\top \mathbf{x}_t^j} = \tilde{\lambda}_1^j(\mathbf{A}).$$

Algorithm 2 runs in time proportional to

$$\mathcal{O}\left(q \cdot t \left(n + \operatorname{nnz}\left(\mathbf{A}\right)\right)\right).$$

We continue to bound the number of repetitions, q and the number of inner iterations, t, of Algorithm 2. Let $e = 2.718\ldots$ and let $\varepsilon = 1 - (1/e)$ and $t = \lceil \log \sqrt{4n} \rceil$; then, with probability at least 3/16, for any fixed $j = 1 \ldots q$,

$$\frac{1}{6}\lambda_1(\mathbf{A}) \leq \frac{1}{2e}\lambda_1(\mathbf{A}) \leq \tilde{\lambda}_1^j(\mathbf{A}).$$

It is now easy to see that the largest value $\tilde{\lambda}_1(\mathbf{A})$ (and the corresponding vector $\mathbf{x}_t$) fails to satisfy the inequality $(1/6)\lambda_1(\mathbf{A}) \leq \tilde{\lambda}_1(\mathbf{A})$ with probability at most

$$\left(1 - \frac{3}{16}\right)^q = \left(\frac{13}{16}\right)^q \leq \delta,$$

where the last inequality follows by setting $q = \lceil 4.82 \log(1/\delta) \rceil \geq \log(1/\delta)/\log(16/13)$. Finally, we note that, from the min-max principle, $\tilde{\lambda}_1(\mathbf{A}) \leq \lambda_1(\mathbf{A})$. Lemma 4 summarizes the above.

Lemma 4. *Let $\tilde{\lambda}_1(\mathbf{A})$ be the output of Algorithm 2 with $q = \lceil 4.82 \log(1/\delta) \rceil$ and $t = \lceil \log \sqrt{4n} \rceil$. Then, with probability at least $1 - \delta$,*

$$\frac{1}{6}\lambda_1(\mathbf{A}) \leq \tilde{\lambda}_1(\mathbf{A}) \leq \lambda_1(\mathbf{A}).$$

The running time of Algorithm 2 is

$$\mathcal{O}\left(n + \mathrm{nnz}\left(\mathbf{A}\right) \log(\mathrm{n}) \log\left(\frac{1}{\delta}\right)\right).$$

2.5.2 The Gaussian trace estimator

Even though computing the trace of a square $n \times n$ matrix requires only $\mathcal{O}(n)$ arithmetic operations, the situation is more complicated when $\mathbf{A}$ is implicitly known through a matrix function, e.g., $\mathbf{A} = \mathbf{X}^2$, for some matrix $\mathbf{X}$ and the user only observes $\mathbf{X}$. For situations such as these, Avron and Toledo [AT11] analyzed several algorithms to estimate the trace of a SPSD matrix $\mathbf{A}$. Algorithm 3 sketches the

Gaussian trace estimator, while Lemma 5 guarantees that Algorithm 3 returns an (ϵ, δ)-estimator to $\mathbf{Tr}(\mathbf{A})$.

Algorithm 3 Gaussian Trace Estimation

Input: SPSD matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, $0 < \varepsilon < 1$, and $0 < \delta < 1$.
Output: γ, the approximation to $\mathbf{Tr}(\mathbf{A})$.

```
1: $p = \lceil 20 \log(2/\delta)/\varepsilon^2 \rceil$
2: Generate $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_p$ i.i.d. random Gaussian vectors in $\mathbb{R}^n$
3: $\gamma = 0$
4: for $i = 1, \ldots, p$ do
5:     $\gamma = \gamma + \mathbf{g}_i^\top \mathbf{A} \mathbf{g}_i$
6: end for
7: return $\gamma = \gamma/p$
```

Lemma 5 ([AT11, Theorem 5.2]). *Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be an SPSD matrix, let $0 < \varepsilon < 1$ be an accuracy parameter, and let $0 < \delta < 1$ be a failure probability. If $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_p \in \mathbb{R}^n$ are independent random standard Gaussian vectors, then, for $p = \lceil 20 \log(2/\delta)/\varepsilon^2 \rceil$, with probability at least $1 - \delta$,*

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

An improved bound for the Gaussian trace estimator can be found in [RKA15]. Roosta-Khorasani and Ascher provide the “sufficient” bound, $p \leq 8 \cdot c(\varepsilon, \delta)$ (where $c(\varepsilon, \delta)$ is a function of ε and δ), for the number of Gaussian vectors required to achieve an (ε, δ)-estimator to $\mathbf{Tr}(\mathbf{A})$. Instead, lemma 5 requires $p \leq 20 \cdot c(\varepsilon, \delta)$ random Gaussian vectors, which is 2.5 times worse than the bound of [RKA15].

2.5.3 Random projections

Random projections is a powerful tool in RandNLA. It is used as a dimensionality reduction technique and is usually preferred for its simplicity; it requires only a cheap multiplication of the input matrix with a random projection matrix of special structure.

The main idea behind random projection is the Johnson–Lindenstrauss (see, Lemma 6) which in words states that a set of high-dimensional points can be projected to a lower-dimensional space nearly preserving the distances between them.

Lemma 6 (Johnson–Lindenstrauss). *Given a set S of d points in $\mathbb{R}^n$ and an integer $k << n$, there exists a map function $f : \mathbb{R}^n \to \mathbb{R}^k$, mapping the points in S to the much lower dimensional space $\mathbb{R}^k$. Setting $k = \mathcal{O}\left(\log(d)/\varepsilon^2\right)$ the guarantee is that with probability at least $1 - 1/d$,*

$$(1-\varepsilon)\|x-y\|_2^2 \leq \|f(x)-f(y)\|_2^2 \leq (1+\varepsilon)\|x-y\|_2^2 \; 0 \leq \varepsilon \leq 1$$

for all pairs of points $x, y \in S$.

The matrix multiplication between the input matrix and the random projection matrix, requires $\mathcal{O}\left(d \cdot n \cdot k\right)$ operations. In [AC09], Ailon and Chazelle claim that this multiplication can be performed in $\mathcal{O}\left(d \cdot n \cdot \log(k)\right)$, using the **fast Johnson-Lindenstrauss transform** (see, [DMMS11, Tro11] for more details).

Many constructions of the random projection matrix have been proposed. A trivial random projection matrix is the r**andom Gaussian matrix**, i.e the matrix whose values are drawn i.i.d. from the normal distribution $\mathcal{N} \sim (0,1)$. In this section we will describe two random projection matrix constructions; the subsampled randomized Hadamard transform (see, Section 2.5.3) and the input sparsity transform (see, Section 2.5.3). We refer the reader to [Woo14] for more details on various constructions of random projection matrices.

The randomized subsampled Hadamard transform The *randomized Hadamard transform* is a crucial step in the design of the fast Johnson-Lindenstrauss transform (see, [AC09]). Its construction is based on the Hadamard matrix, $\tilde{\mathbf{H}}_n$, that is recursively defined as:

$$\tilde{\mathbf{H}}_n = \begin{bmatrix} \tilde{\mathbf{H}}_{n/2} & \tilde{\mathbf{H}}_{n/2} \\ \tilde{\mathbf{H}}_{n/2} & -\tilde{\mathbf{H}}_{n/2} \end{bmatrix} \quad \tilde{\mathbf{H}}_2 = \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix}.$$

The *normalized Hadamard transform* can then be defined as:

$$\mathbf{H}_n = \frac{1}{\sqrt{n}}\tilde{\mathbf{H}}_n. \tag{2.23}$$

Let $\mathbf{D} \in \mathbb{R}^{d\times d}$ be the diagonal matrix with entries that get values according to eqn. (2.24).

$$\mathbf{D}_{ii} = \begin{cases} +1 & \text{, with probability } 1/2 \\ -1 & \text{, with probability } 1/2 \end{cases} \tag{2.24}$$

The *randomized Hadamard transform* is then defined to be the product **HD**. If we further define the matrix $\mathbf{S} \in \mathbb{R}^{n\times k}$ to be the matrix of k rows of $\mathbf{I}_n$ that are picked uniformly at random with replacement we can define the product **DHS** as the *randomized subsampled Hadamard transform*. Algorithm 4 summarizes the steps descibed for the construction of the randomized subsampled Hadamard transform. Lemma 7 has appeared in [DMMS11, Tro11] and [Woo14] and describes the quality

Algorithm 4 The Subsampled Randomized Hadamard Transform

```
Input: Integers $n, k > 0$ with $k \ll n$.
Output: $\mathbf{\Pi} \in \mathbb{R}^{n\times k}$, the random projection matrix.
1: Let $\mathbf{S}$ be an empty matrix.                    ▷ it will finally be in $\mathbb{R}^{n\times k}$
2: for $t = 1, \ldots, k$ do
3:     Pick uniformly at random an integer, $i$, from $\{1, 2, \ldots, n\}$.   ▷ i.i.d. trials with replacement
4:     Append $\mathbf{e}_i$ to $\mathbf{S}$.         ▷ $\mathbf{e}_i \in \mathbb{R}^n$ is the $i$-th canonical vector.
5: end for
6: Let $\mathbf{H} \in \mathbb{R}^{n\times n}$ be the normalized Hadamard transform matrix.   ▷ see eqn. (2.23).
7: Let $\mathbf{D} \in \mathbb{R}^{n\times n}$ be a diagonal matrix with
       $\mathbf{D}_{ii} = \begin{cases} +1 & \text{, with probability } 1/2 \\ -1 & \text{, with probability } 1/2 \end{cases}$
8: return $\mathbf{\Pi} = \mathbf{DHS}$.
```

of the random projection using the subsampled randomized Hadamard transform.

Lemma 7. *Let* $\mathbf{U} \in \mathbb{R}^{n\times r}$ *such that* $\mathbf{U}^\top\mathbf{U} = \mathbf{I}_r$ *and let* $\mathbf{\Pi} \in \mathbb{R}^{n\times k}$ *be constructed by Algorithm 4. Then, with probability at least 0.9,*

$$\left\| \frac{n}{r}\mathbf{U}^\top\mathbf{\Pi}\mathbf{\Pi}^\top\mathbf{U} - \mathbf{I}_r \right\|_2 \le \varepsilon,$$

by setting

$$s = \mathcal{O}\left((r + \log(n)) \cdot \frac{\log(r)}{\varepsilon^2} \right).$$

Input sparsity transform The input sparsity transform of [CW13] is a major breakthrough that uses a special construction for the random projection matrix[8]. It's novelty, lies in the fact that the matrix multiplication between the input matrix, $\mathbf{A}$ and the random projection matrix, can be performed in time proportional to the sparsity of $\mathbf{A}$, $\mathcal{O}(\text{nnz}(\mathbf{A}))$. The input sparsity transform was further analyzed in [MM13] and [NN13]. Algorithm 5 sketches the construction of the transform. Lemma 8 has appeared in [MM13].

Algorithm 5 Input-Sparsity Transform

```
Input: Integers $n, k > 0$ with $k \ll n$.
Output: $\mathbf{\Pi} \in \mathbb{R}^{n\times k}$, the random projection matrix.
1: Let $\mathbf{S}$ be an empty matrix.
2: for $t = 1, \ldots, n$ do
3:     Pick uniformly at random an integer, $i$, from $\{1, 2, \ldots, k\}$.    ▷ ii.i.d. trials with replacement
4:     Append $\mathbf{e}_i^\top$ to $\mathbf{S}$.    ▷ $\mathbf{e}_i \in \mathbb{R}^k$ is the $i$-th canonical vector.
5: end for
6: Let $\mathbf{D} \in \mathbb{R}^{n\times n}$ be a diagonal matrix with
       $$\mathbf{D}_{ii} = \begin{cases} +1 & \text{, with probability } 1/2 \\ -1 & \text{, with probability } 1/2 \end{cases}$$
7: return $\mathbf{\Pi} = \mathbf{DS}$.
```

[8]Also, known as `CountSketch` matrix, especially in data stream literature; see, [CCFC04]

Lemma 8 ([MM13, Appendix A1]). *Let* $\mathbf{U} \in \mathbb{R}^{n \times r}$ *such that* $\mathbf{U}^\top \mathbf{U} = \mathbf{I}_r$ *and let* $\mathbf{\Pi} \in \mathbb{R}^{n \times k}$ *be constructed by Algorithm 5. Then, with probability at least 0.9,*

$$\|\mathbf{U}^\top \mathbf{\Pi} \mathbf{\Pi}^\top \mathbf{U} - \mathbf{I}_r\|_2 \leq \varepsilon,$$

by setting

$$k = \mathcal{O}\left(r^2/\varepsilon^2\right).$$

We refer the interested reader to [NN13] for improved analyses of Algorithm 5 and its variants.

We do note that even though it appears that Algorithm 5 is always better than Algorithm 4 (at least in terms of their respective *theoretical* running times), both algorithms are worth evaluating experimentally: in particular, prior work [PBMID13] has reported that Algorithm 4 often outperforms Algorithm 5 in terms of empirical accuracy and running time when the input matrix is dense.

3 APPROXIMATION OF THE LOGARITHM DETERMINANT OF A SYMMETRIC POSITIVE DEFINITE MATRIX

We present our algorithm for the approximation of the logarithm determinant of a symmetric positive definite matrix. Throughout the chapter we will use the notation $\operatorname{logdet}(\mathbf{A})$ to refer to the logarithm of the determinant of a SPD matrix $\mathbf{A}$ and the notation $\widehat{\operatorname{logdet}}(\mathbf{A})$ to refer to its approximation. Our algorithm achieves, with high probability, additive error guarantee for general SPD matrices, and relative error guarantee for SPD matrices with spectrum in the interval $(0, 1)$.

The chapter is organized as follows: in Section 3.1 we provide the mathematical setting under which we will work and any essential information that will be useful through out the chapter. In Section 3.2 we describe the main algorithm that achieves additive error approximation to $\operatorname{logdet}(\mathbf{A})$. In Section 3.3 we describe the modified main algorithm that achieves relative error approximation to $\operatorname{logdet}(\mathbf{A})$. Finally, Section 3.4 provides a comprehensive empirical evaluation of our main algorithm on a variety of datasets.

Sections of chapter 3 have been published in [BDK$^+$17]

A Randomized Algorithm for Approximating the Log Determinant of a Symmetric Positive Definite Matrix C. Boutsidis, P. Drineas, P. Kambadur, E-M. Kontopoulou, A. Zouzias in Linear Algebra and its Applications (2017), Vol. 533, pp.95-117

3.1 Setting

We focus on SPD matrices with a full set of eigenvectors. Given an SPD matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ our goal is to compute $\operatorname{logdet}(\mathbf{A})$. Towards this end, we define the SPD matrix $\mathbf{B} = \mathbf{A}/\alpha$ where $0 < \alpha < \|\mathbf{A}\|_2$. Obviously $\|\mathbf{B}\|_2 < 1$. Now, consider the eigendecomposition of $\mathbf{B}$, $\mathbf{B} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$, where $\mathbf{U} \in \mathbb{R}^{n\times n}$ is the orthogonal matrix

whose columns correspond to the eigenvectors of $\mathbf{B}$ and $\mathbf{\Lambda}$ is the diagonal matrix with the eigenvalues , $\lambda_i(\mathbf{B}),\ i = 1, \ldots, n$, of $\mathbf{B}$ in decreasing order.
It follows that

$$\begin{aligned}
\operatorname{logdet}(\mathbf{A}) &= \log(\mathbf{det}\,(\alpha \cdot \mathbf{A}/\alpha)) \\
&= \log(\alpha^n \mathbf{det}\,(\mathbf{A}/\alpha)) && (3.1) \\
&= \log(\alpha^n) + \log(\mathbf{det}\,(\mathbf{B})) && (3.2) \\
&= n\log(\alpha) + \log\left(\prod_{i=1}^{n} \lambda_i(\mathbf{B})\right) && (3.3) \\
&= n\log(\alpha) + \sum_{i=1}^{n} \log(\lambda_i(\mathbf{B})) && (3.4) \\
&= n\log(\alpha) + \mathbf{Tr}\,(\log[\mathbf{B}])\,. && (3.5)
\end{aligned}$$

Eqn. (3.1) and eqn. (3.3) follow from standard properties of the determinant. Eqn. (3.2) and eqn. (3.4) follow from standard properties of the logarithm function. Finally, eqn. (3.5) follows from the fact that,

$$\begin{aligned}
\mathbf{Tr}\,(\log[\mathbf{B}]) &= \sum_{i=1}^{n} \lambda_i(\log[\mathbf{B}]) \\
&= \sum_{i=1}^{n} \log(\lambda_i(\mathbf{B})). && (3.6)
\end{aligned}$$

Eqn. (3.6) follows from the definition of matrix functions (see, Section 2.2.6). More precisely, let $h(x) = \log x$ for any $x > 0$ and let $h(0) = 0$. Then,

$$\begin{aligned}
\sum_{i,\lambda_i>0} \log \lambda_i(\mathbf{B}) &= \mathbf{Tr}\left(h(\mathbf{\Lambda})\right) \\
&= \mathbf{Tr}\left(h(\mathbf{\Lambda})\mathbf{U}^\top\mathbf{U}\right) \\
&= \mathbf{Tr}\left(\mathbf{U}h(\mathbf{\Lambda})\mathbf{U}^\top\right) && (3.7) \\
&= \mathbf{Tr}\left(h(\mathbf{B})\right) \\
&= \mathbf{Tr}\left(\log\left[\mathbf{B}\right]\right), && (3.8)
\end{aligned}$$

where eqn. (3.7) follows from the circular property of the trace operator and eqn. (3.8) follows from the definition of $h(x)$.

3.2 Additive error approximation for general SPD matrices

Lemma 9 is the starting point of our main algorithm for approximating the determinant of a SPD matrix. In words the lemma states that the computation of the logarithm determinant of an SPD matrix $\mathbf{A}$ can be reduced to a two-fold task; first, computing the largest eigenvalue of $\mathbf{A}$ followed by computing the trace of all the powers of a matrix $\mathbf{C}$ related to $\mathbf{A}$.

Lemma 9. *Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be an SPD matrix. For any α with $\lambda_1(\mathbf{A}) < \alpha$, define $\mathbf{B} := \mathbf{A}/\alpha$ and $\mathbf{C} := \mathbf{I}_n - \mathbf{B}$. Then,*

$$\text{logdet}\left(\mathbf{A}\right) = n\log(\alpha) - \sum_{k=1}^{\infty} \frac{\mathbf{Tr}\left(\mathbf{C}^k\right)}{k}. \qquad (3.9)$$

Proof. In eqn. (3.5) we proved that given SPD matrices $\mathbf{A} \in \mathbb{R}^{n\times n}$ and $\mathbf{B} = \mathbf{A}/\alpha$ for $0 < \alpha \le \|\mathbf{A}\|_2$, $\text{logdet}\left(\mathbf{A}\right) = n\log(\alpha) + \mathbf{Tr}\left(\log\left[\mathbf{B}\right]\right)$. The goal is to compute $\mathbf{Tr}\left(\log\left[\mathbf{B}\right]\right)$ as, given α the computation of $n\log(\alpha)$ is trivial.

Algorithm 6 High Level Procedure for Log Determinant Computation

```
 1: function LOGDET($\mathbf{A}$)
 2:     Compute $\alpha$ such that $\lambda_1(\mathbf{A}) < \alpha$.
 3:     $\mathbf{C} \leftarrow \mathbf{I}_n - \mathbf{A}/\alpha$.
 4:     $Sum \leftarrow 0$.
 5:     for $k \leftarrow 1, \ldots$ do
 6:         $Sum \leftarrow Sum + \mathbf{Tr}\left(\mathbf{C}^k\right)/k$.
 7:     end for
 8:     $\text{logdet}(\mathbf{A}) \leftarrow n\log(\alpha) - Sum$.
 9:     return $\text{logdet}(\mathbf{A})$.
10: end function
```

$$\begin{aligned} \mathbf{Tr}\left(\log\left[\mathbf{B}\right]\right) &= \mathbf{Tr}\left(\log\left[\mathbf{I}_n - (\mathbf{I}_n - \mathbf{B})\right]\right) \\ &= \mathbf{Tr}\left(-\sum_{k=1}^{\infty}\frac{(\mathbf{I}_n - \mathbf{B})^k}{k}\right) \qquad (3.10) \\ &= -\sum_{k=1}^{\infty}\frac{\mathbf{Tr}\left(\mathbf{C}^k\right)}{k}. \qquad (3.11) \end{aligned}$$

Eqn. (3.10) follows from the Taylor expansion of Lemma 2 because all the eigenvalues of $\mathbf{C} = \mathbf{I}_n - \mathbf{B}$ are contained[1] in $(0, 1)$ and Eqn. (3.11) follows from the linearity of the trace operator. □

3.2.1 Algorithm

Lemma 9 indicates a high-level function for computing the logarithm determinant of an SPD matrix $\mathbf{A}$ that is described in Algorithm 6.

To implement step 2 we use the power method from numerical linear algebra literature (see, Section 2.5.1). Step 3 is straightforward. To implement step 5, we truncate the expansion $\sum_{k=1}^{\infty}\mathbf{Tr}\left(\mathbf{C}^k\right)$ keeping few of the summands. This step is important

[1]Indeed, if $\mathbf{x}$ is an eigenvector of $\mathbf{B}$ then $(\mathbf{I}_n - \mathbf{B})\mathbf{x} = (1 - \lambda_x(\mathbf{B}))\mathbf{x}$, which means that $\lambda_i(\mathbf{C}) = 1 - \lambda_i(\mathbf{B})$. Given that $0 < \lambda_i(\mathbf{B}) < 1$ for all $i = 1 \ldots n$, we conclude that $0 < \lambda_i(\mathbf{C}) < 1$, for all $i = 1 \ldots n$.

Algorithm 7 Additive Error Randomized Log Determinant Estimation

```
Input: $\mathbf{A} \in \mathbb{R}^{n\times n}$, accuracy parameter $\varepsilon > 0$, and integer $m > 0$.
Output: $\widehat{\operatorname{logdet}}(\mathbf{A})$ the approximation to $\operatorname{logdet}(\mathbf{A})$.
1:  Compute $\tilde{\lambda}_1(\mathbf{A})$ using Algorithm 2    ▷ iwith $t = \mathcal{O}(\log n)$ and $q = \mathcal{O}(\log(1/\delta))$.
2:  $\alpha = 7\tilde{\lambda}_1(\mathbf{A})$
3:  $\mathbf{C} = \mathbf{I}_n - \mathbf{A}/\alpha$
4:  $p = \lceil 20\log(2/\delta)/\varepsilon^2 \rceil$
5:  Generate $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_p \in \mathbb{R}^n$ be i.i.d. random Gaussian vectors.
6:  for $i = 1, 2 \ldots, p$ do
7:      $\mathbf{v}_1^{(i)} = \mathbf{C}\mathbf{g}_i$
8:      $\boldsymbol{\gamma}_1^{(i)} = \mathbf{g}_i^\top \mathbf{v}_1^{(i)}$
9:      for $k = 2, \ldots, m$ do
10:         $\mathbf{v}_k^{(i)} := \mathbf{C}\mathbf{v}_{k-1}^{(i)}$.
11:         $\boldsymbol{\gamma}_k^{(i)} = \mathbf{g}_i^\top \mathbf{v}_k^{(i)}$    ▷ Inductively $\boldsymbol{\gamma}_k^{(i)} = \mathbf{g}_i^\top \mathbf{C}^k \mathbf{g}_i$)
12:     end for
13: end for
14: return $\widehat{\operatorname{logdet}}(\mathbf{A}) = n\log(\alpha) - \sum_{k=1}^{m}\left(\frac{1}{p}\sum_{i=1}^{p}\boldsymbol{\gamma}_k^{(i)}\right)/k$
```

since the quality of the approximation, both theoretically and empirically, depends on the number of summands (denoted by m) that will be kept. On the other hand, the running time of the algorithm increases with m.

Finally, to estimate the traces of the powers of $\mathbf{C}$, we use the randomized algorithm of Section 2.5.2. Our approach is described in detail in Algorithm 7; notice that step 8 in Algorithm 7 is an efficient way of computing

$$\widehat{\operatorname{logdet}}(\mathbf{A}) := n\log(\alpha) - \sum_{k=1}^{m}\frac{1}{k}\left(\frac{1}{p}\sum_{i=1}^{p}\mathbf{g}_i^\top \mathbf{C}^k \mathbf{g}_i\right). \tag{3.12}$$

3.2.2 Error bound

Lemma 10 that follows, proves that Algorithm 7 returns an accurate approximation to $\operatorname{logdet}(\mathbf{A})$.

Lemma 10. *Let* $\widehat{\text{logdet}}\left(\mathbf{A}\right)$ *be the output of Algorithm 7 on inputs* $\mathbf{A}$, m, *and* ε. *Then, with probability at least* $1-2\delta$,

$$\left|\widehat{\text{logdet}}\left(\mathbf{A}\right)-\text{logdet}\left(\mathbf{A}\right)\right| \leq \left(\varepsilon+\left(1-\frac{1}{7\kappa\left(\mathbf{A}\right)}\right)^{m}\right)\cdot\boldsymbol{\gamma}, \tag{3.13}$$

where $\boldsymbol{\gamma}=\sum_{i=1}^{n}\log\left(7\cdot\frac{\lambda_1(\mathbf{A})}{\lambda_i(\mathbf{A})}\right)$. *If* $m\geq\left\lceil 7\kappa\left(\mathbf{A}\right)\log\left(\frac{1}{\varepsilon}\right)\right\rceil$, *then*

$$\left|\widehat{\text{logdet}}\left(\mathbf{A}\right)-\text{logdet}\left(\mathbf{A}\right)\right|\leq 2\gamma\varepsilon. \tag{3.14}$$

Proof. First, note that using our choice for α in Step 2 of Algorithm 7 and applying Lemma 4, we get that, with probability at least $1-\delta$,

$$\lambda_1(\mathbf{A})<\frac{7}{6}\lambda_1(\mathbf{A})\leq\alpha\leq 7\lambda_1(\mathbf{A}), \tag{3.15}$$

The strict inequality at the leftmost side of the above equation follows since all eigenvalues of $\mathbf{A}$ are strictly positive. Let's call the event that the above inequality holds $\mathcal{E}_1$; obviously, $\mathbf{Pr}\left[\mathcal{E}_1\right]\geq 1-\delta$ (and thus $\mathbf{Pr}\left[\bar{\mathcal{E}_1}\right]\leq\delta$). We condition all further derivations on $\mathcal{E}_1$ holding and we manipulate $\Delta=\left|\widehat{\text{logdet}}\left(\mathbf{A}\right)-\text{logdet}\left(\mathbf{A}\right)\right|$ as follows:

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

Below, we bound Δ_1 and Δ_2 separately. We start with Δ_1; the idea is to apply Lemma 5 on the matrix $\sum_{k=1}^{m}\mathbf{C}^k/k$ with $p=\lceil 20\log(2/\delta)/\varepsilon^2\rceil$. Let $\mathcal{E}_2$ denote the probability that Lemma 5 holds; obviously, $\mathbf{Pr}\left[\mathcal{E}_2\right]\geq 1-\delta$ (and thus $\mathbf{Pr}\left[\bar{\mathcal{E}_2}\right]\leq\delta$)

given our choice of p. We condition all further derivations on $\mathcal{E}_2$ holding as well to get

$$\begin{aligned}\Delta_1 &\leq \varepsilon \cdot \mathbf{Tr}\left(\sum_{k=1}^{m} \mathbf{C}^k/k\right) \\ &\leq \varepsilon \cdot \mathbf{Tr}\left(\sum_{k=1}^{\infty} \mathbf{C}^k/k\right). \qquad (3.16)\end{aligned}$$

In the inequalityof eqn. (3.16) we used the fact that $\mathbf{C}$ is a positive matrix, hence for all k, $\mathbf{Tr}\left(\mathbf{C}^k\right) > 0$. The second term Δ_2 is bounded as follows:

$$\begin{aligned}\Delta_2 &= \left|\sum_{k=m+1}^{\infty} \mathbf{Tr}\left(\mathbf{C}^k\right)/k\right| \leq \sum_{k=m+1}^{\infty} \mathbf{Tr}\left(\mathbf{C}^k\right)/k \qquad (3.17)\\ &= \sum_{k=m+1}^{\infty} \mathbf{Tr}\left(\mathbf{C}^m \cdot \mathbf{C}^{k-m}\right)/k \leq \sum_{k=m+1}^{\infty} \|\mathbf{C}^m\|_2 \cdot \mathbf{Tr}\left(\mathbf{C}^{k-m}\right)/k \qquad (3.18)\\ &= \|\mathbf{C}^m\|_2 \cdot \sum_{k=m+1}^{\infty} \mathbf{Tr}\left(\mathbf{C}^{k-m}\right)/k \leq \|\mathbf{C}^m\|_2 \cdot \sum_{k=1}^{\infty} \mathbf{Tr}\left(\mathbf{C}^k\right)/k \\ &\leq \left(1 - \frac{\lambda_n\left(\mathbf{A}\right)}{\alpha}\right)^m \cdot \sum_{k=1}^{\infty} \mathbf{Tr}\left(\mathbf{C}^k\right)/k. \qquad (3.19)\end{aligned}$$

In the inequality of eqn. (3.17), we used the triangle inequality and the fact that $\mathbf{C}$ is a positive matrix. In the inequality of eqn. (3.18), we used the following fact[2]: given two positive semidefinite matrices $\mathbf{A}, \mathbf{B}$ of the same size, $\mathbf{Tr}\left(\mathbf{AB}\right) \leq \|\mathbf{A}\|_2 \cdot \mathbf{Tr}\left(\mathbf{B}\right)$. Finally, in the inequality of eqn. (3.19), we used the fact that

$$\lambda_1(\mathbf{C}) = 1 - \lambda_n(\mathbf{B}) = 1 - \lambda_n(\mathbf{A})/\alpha.$$

Combining the bounds for Δ_1 and Δ_2 gives

$$\left|\widehat{\text{logdet}}\left(\mathbf{A}\right) - \text{logdet}\left(\mathbf{A}\right)\right| \leq \left(\epsilon + \left(1 - \frac{\lambda_n\left(\mathbf{A}\right)}{\alpha}\right)^m\right) \cdot \sum_{k=1}^{\infty} \frac{\mathbf{Tr}\left(\mathbf{C}^k\right)}{k}.$$

[2]This follows from Von Neumann's trace inequality.

We have already proven in Lemma 9 that

$$\sum_{k=1}^{\infty} \frac{\mathbf{Tr}\left(\mathbf{C}^k\right)}{k} = -\mathbf{Tr}\left(\log\left[\mathbf{B}\right]\right) = n\log(a) - \operatorname{logdet}\left(\mathbf{A}\right).$$

Notice that the assumption of Lemma 9 ($\lambda_1(\mathbf{A}) < \alpha$) is satisfied from the inequality of eqn. (3.15). We further manipulate the last term as follows:

$$\begin{aligned} n\log(a) - \operatorname{logdet}\left(\mathbf{A}\right) &= n\log(\alpha) - \log\left(\prod_{i=1}^{n} \lambda_i(\mathbf{A})\right) \\ &= n\log(\alpha) - \sum_{i=1}^{n} \log\left(\lambda_i(\mathbf{A})\right) \\ &= \sum_{i=1}^{n} \left(\log\left(\alpha\right) - \log\left(\lambda_i(\mathbf{A})\right)\right) \\ &= \sum_{i=1}^{n} \log\left(\frac{\alpha}{\lambda_i(\mathbf{A})}\right). \end{aligned}$$

Collecting our results together, we get:

$$\left|\widehat{\operatorname{logdet}}\left(\mathbf{A}\right) - \operatorname{logdet}\left(\mathbf{A}\right)\right| \leq \left(\epsilon + \left(1 - \frac{\lambda_n\left(\mathbf{A}\right)}{\alpha}\right)^m\right) \cdot \sum_{i=1}^{n} \log\left(\frac{\alpha}{\lambda_i(\mathbf{A})}\right).$$

Using the inequality of eqn. (3.15) (only the upper bound on α is needed here) proves the inequality of eqn. (3.13) of Lemma 9 getting:

$$\begin{aligned} \left|\widehat{\operatorname{logdet}}\left(\mathbf{A}\right) - \operatorname{logdet}\left(\mathbf{A}\right)\right| &\leq \left(\epsilon + \left(1 - \frac{\lambda_n\left(\mathbf{A}\right)}{7\lambda_1\left(\mathbf{A}\right)}\right)^m\right) \cdot \underbrace{\sum_{i=1}^{n} \log\left(\frac{7\lambda_1\left(\mathbf{A}\right)}{\lambda_i(\mathbf{A})}\right)}_{\gamma} \\ &\leq \left(\epsilon + \left(1 - \frac{\lambda_n\left(\mathbf{A}\right)}{7\lambda_1\left(\mathbf{A}\right)}\right)^m\right) \cdot \gamma \qquad (3.20) \end{aligned}$$

To prove the inequality of eqn. (3.14), we use the well-known property:

$$\left(1 - \xi^{-1}\right)^{\rho} \leq e^{-\rho/\xi}, \qquad (3.21)$$

where $e = 2.718\ldots$, is the Euler's number, and $\rho, \xi > 0$. Then observe that:

$$\left(1 - \frac{\lambda_n\left(\mathbf{A}\right)}{7\lambda_1\left(\mathbf{A}\right)}\right)^m = \left(1 - \frac{1}{7\kappa\left(\mathbf{A}\right)}\right)^m .$$

To get relative error we need to find the value of m such that $\left(1 - \frac{1}{7\kappa(\mathbf{A})}\right)^m \leq \varepsilon$, then from the inequality of eqn. (3.21) we get:

$$\left(1 - \frac{1}{7\kappa\left(\mathbf{A}\right)}\right)^m \leq e^{\frac{-m}{7\kappa(\mathbf{A})}}, \tag{3.22}$$

and we restrict the rightmost part to be at most ε:

$$\begin{aligned} e^{\frac{-m}{7\kappa(\mathbf{A})}} &\leq \varepsilon \\ \frac{-m}{7\kappa\left(\mathbf{A}\right)} &\leq \log \varepsilon \\ m &\geq 7\kappa\left(\mathbf{A}\right) \log\left(\frac{1}{\varepsilon}\right). \end{aligned}$$

Using the derived lower bound for m, the bound of eqn. (3.20) transforms to a relative error one:

$$\left|\widehat{\text{logdet}}\left(\mathbf{A}\right) - \text{logdet}\left(\mathbf{A}\right)\right| \leq 2\gamma\varepsilon.$$

Finally, recall that we conditioned all derivations on events $\mathcal{E}_1$ and $\mathcal{E}_2$ both holding, which can be bounded as follows:

$$\mathbf{Pr}\left[\mathcal{E}_1 \cap \mathcal{E}_2\right] = 1 - \mathbf{Pr}\left[\bar{\mathcal{E}}_1 \cup \bar{\mathcal{E}}_2\right],$$

using the union bound we get:

$$1 - \mathbf{Pr}\left[\bar{\mathcal{E}}_1 \cup \bar{\mathcal{E}}_2\right] \geq 1 - \mathbf{Pr}\left[\bar{\mathcal{E}}_1\right] - \mathbf{Pr}\left[\bar{\mathcal{E}}_1\right] \geq 1 - 2\delta.$$

This concludes our proof. □

3.2.3 Running time

Step 1 takes $\mathcal{O}\left(\mathrm{nnz}\left(\mathbf{A}\right)\log(\mathrm{n})\log(1/\delta)\right)$ time; we assume that $\mathrm{nnz}\left(\mathbf{A}\right) \geq \mathrm{n}$, since otherwise the determinant of $\mathbf{A}$ would be trivially equal to zero. The algorithm inductively computes $\mathbf{v}_k$ and $\mathbf{g}_i^\top \mathbf{C}^k \mathbf{g}_i = \mathbf{g}_i^\top \mathbf{v}_k$ for all $k = 1, 2, \ldots, m$. Given $\mathbf{v}_{k-1}$, $\mathbf{v}_k$ and $\mathbf{g}_i^\top \mathbf{C}^k \mathbf{g}_i$ can be computed in $\mathrm{nnz}\left(\mathbf{C}\right)$ and $\mathcal{O}\left(n\right)$ time, respectively. Notice that $\mathrm{nnz}\left(\mathbf{C}\right) \leq \mathrm{n} + \mathrm{nnz}\left(\mathbf{A}\right)$. Therefore, step 5 requires $\mathcal{O}\left(p \cdot m \cdot \mathrm{nnz}\left(\mathbf{A}\right)\right)$ time. Since $p = \mathcal{O}\left(\varepsilon^{-2}\log(1/\delta)\right)$, the total cost of Algorithm 7 is

$$\mathcal{O}\left(\mathrm{nnz}\left(\mathbf{A}\right) \cdot \left(\frac{\mathrm{m}}{\varepsilon^2} + \log \mathrm{n}\right) \cdot \log\left(\frac{1}{\delta}\right)\right).$$

3.3 Relative error approximation for SPD matrices with bounded eigenvalues

In this section, we argue that a simplified version of Algorithm 7 achieves a relative error approximation to the $\mathrm{logdet}\left(\mathbf{A}\right)$, under the assumption that all the eigenvalues of the SPD matrix $\mathbf{A}$ lie in the interval $(\theta_1, 1)$, where $0 < \theta_1 < 1$. This is a mild generalization of the setting introduced in [HMS15].

Given the upper bound on the largest eigenvalue of $\mathbf{A}$, the proof of the following lemma (which is the analog of Lemma 9) is straightforward.

Lemma 11. *Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be an SPD matrix whose eigenvalues lie in the interval $(\theta_1, 1)$, for some $0 < \theta_1 < 1$. Let $\mathbf{C} := \mathbf{I}_n - \mathbf{A}$; then,*

$$\mathrm{logdet}\left(\mathbf{A}\right) = -\sum_{k=1}^{\infty} \frac{\mathbf{Tr}\left(\mathbf{C}^k\right)}{k}. \tag{3.23}$$

Proof. Similarly to the proof of Lemma 9,

$$\mathrm{logdet}\left(\mathbf{A}\right) = \log\left(\prod_{i=1}^{n} \lambda_i(\mathbf{A})\right) = \sum_{i=1}^{n} \log(\lambda_i(\mathbf{A})) = \mathbf{Tr}\left(\log\left[\mathbf{A}\right]\right).$$

Now,

$$\begin{aligned}\mathbf{Tr}\left(\log\left[\mathbf{A}\right]\right) &= \mathbf{Tr}\left(\log\left[\mathbf{I}_n - \left(\mathbf{I}_n - \mathbf{A}\right)\right]\right) \\ &= \mathbf{Tr}\left(-\sum_{k=1}^{\infty}\frac{\left(\mathbf{I}_n - \mathbf{A}\right)^k}{k}\right) \qquad (3.24) \\ &= -\sum_{k=1}^{\infty}\frac{\mathbf{Tr}\left(\mathbf{C}^k\right)}{k}. \qquad (3.25)\end{aligned}$$

Eqn. (3.24) follows from the Taylor expansion of Lemma 2 because all the eigenvalues of $\mathbf{C} = \mathbf{I}_n - \mathbf{A}$ are contained[3] in the interval $(0, 1)$. Eqn. (3.25) follows from the linearity of the trace operator. □

Then eqn. (3.23) suggests the approximation scheme that is the analog of the one defined in eqn. (3.12) and emerges using the trace estimator of Lemma (5) and truncating the infinite series to m:

$$\widehat{\operatorname{logdet}}\left(\mathbf{A}\right) := -\sum_{k=1}^{m}\frac{1}{k}\left(\frac{1}{p}\sum_{i=1}^{p}\mathbf{g}_i^{\top}\mathbf{C}^k\mathbf{g}_i\right). \qquad (3.26)$$

3.3.1 Algorithm

We simplify Algorithm 7 as follows: we skip steps 1 and 2 and in step 3 we set $\mathbf{C} = \mathbf{I}_n - \mathbf{A}$. Finally the algorithm returns the approximation to the $\operatorname{logdet}(\mathbf{A})$ using eqn. (3.26). Algorithm 8 sketches the aforementioned changes.

3.3.2 Error bound

Lemma (12) proves that in the special case when all the eigenvalues of the SPD matrix $\mathbf{A}$ lie in the interval $(\theta_1, 1)$, with $0 < \theta_1 < 1$, Algorithm 8 returns a relative error approximation to $\operatorname{logdet}(\mathbf{A})$.

[3]Indeed, if $\mathbf{x}$ is an eigenvector of $\mathbf{A}$ then $(\mathbf{I}_n - \mathbf{A})\mathbf{x} = (1 - \lambda_x(\mathbf{A}))\mathbf{x}$, which means that $\lambda_i(\mathbf{C}) = 1 - \lambda_i(\mathbf{A})$. Given that $\theta_1 < \lambda_i(\mathbf{A}) < 1$ for all $i = 1 \ldots n$ and that $0 < \theta_1 < 1$, we conclude that $0 < \lambda_i(\mathbf{C}) < 1 - \theta_1$, for all $i = 1 \ldots n$.

Algorithm 8 Relative Error Randomized Log Determinant Estimation

```
Input: $\mathbf{A} \in \mathbb{R}^{n \times n}$, accuracy parameter $\varepsilon > 0$, and integer $m > 0$.
Output: $\widehat{\text{logdet}}(\mathbf{A})$ the approximation to $\text{logdet}(\mathbf{A})$.
1: $\mathbf{C} = \mathbf{I}_n - \mathbf{A}$
2: $p = \lceil 20 \log(2/\delta)/\varepsilon^2 \rceil$
3: Generate $\mathbf{g}_1, \mathbf{g}_2, \dots, \mathbf{g}_p \in \mathbb{R}^n$ be i.i.d. random Gaussian vectors.
4: for $i = 1, 2 \dots, p$ do
5:     $\mathbf{v}_1^{(i)} = \mathbf{C}\mathbf{g}_i$
6:     $\boldsymbol{\gamma}_1^{(i)} = \mathbf{g}_i^\top \mathbf{v}_1^{(i)}$
7:     for $k = 2, \dots, m$ do
8:         $\mathbf{v}_k^{(i)} := \mathbf{C}\mathbf{v}_{k-1}^{(i)}$.
9:         $\boldsymbol{\gamma}_k^{(i)} = \mathbf{g}_i^\top \mathbf{v}_k^{(i)}$        ▷ Inductively $\boldsymbol{\gamma}_k^{(i)} = \mathbf{g}_i^\top \mathbf{C}^k \mathbf{g}_i$
10:    end for
11: end for
12: return $\widehat{\text{logdet}}(\mathbf{A}) = -\sum_{k=1}^m \left(\frac{1}{p}\sum_{i=1}^p \boldsymbol{\gamma}_k^{(i)}\right)/k$
```

Lemma 12. *Let* $\widehat{\text{logdet}}(\mathbf{A})$ *be the output of Algorithm 8 on inputs* $\mathbf{A}$ *and* ε. *Then, with probability at least* $1-\delta$,

$$\left|\widehat{\text{logdet}}(\mathbf{A}) - \text{logdet}(\mathbf{A})\right| \leq 2\varepsilon \cdot |\text{logdet}(\mathbf{A})|.$$

Proof. The proof follows in lines the proof of Lemma 10. We first manipulate the absolute error $\Delta = \left|\widehat{\text{logdet}}(\mathbf{A}) - \text{logdet}(\mathbf{A})\right|$ as follows:

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

We continue by separately bounding Δ_1 and Δ_2 . We start with Δ_1; the idea is to apply Lemma 5 to approximate the trace of the matrix $\sum_{k=1}^m \mathbf{C}^k/k$ using $p =$

$\lceil 20\log(2/\delta)/\varepsilon^2 \rceil$, random Gaussian vectors. Hence, with probability at least $1-\delta$ (this is the only probabilistic event in this lemma and hence $1-\delta$ is a lower bound on the success probability of the lemma):

$$\begin{aligned}\Delta_1 \leq & \varepsilon \cdot \mathbf{Tr}\left(\sum_{k=1}^{m} \mathbf{C}^k/k\right) \\ \leq & \varepsilon \cdot \mathbf{Tr}\left(\sum_{k=1}^{\infty} \mathbf{C}^k/k\right). \end{aligned} \tag{3.27}$$

The inequality of eqn. (3.27) follows if we observe that $\mathbf{C}$ is positive definite, hence for all k, $\mathbf{Tr}\left(\mathbf{C}^k\right) > 0$. Bounding Δ_2 follows the lines of the proof of Lemma 10:

$$\begin{aligned}\Delta_2 &= \left|\sum_{k=m+1}^{\infty} \mathbf{Tr}\left(\mathbf{C}^k\right)/k\right| = \left|\sum_{k=m+1}^{\infty} \mathbf{Tr}\left(\mathbf{C}^m \cdot \mathbf{C}^{k-m}\right)/k\right| \\ &\leq \left|\sum_{k=m+1}^{\infty} \|\mathbf{C}^m\|_2 \cdot \mathbf{Tr}\left(\mathbf{C}^{k-m}\right)/k\right| = \|\mathbf{C}^m\|_2 \cdot \left|\sum_{k=m+1}^{\infty} \mathbf{Tr}\left(\mathbf{C}^{k-m}\right)/k\right| \\ &\leq \|\mathbf{C}^m\|_2 \cdot \left|\sum_{k=1}^{\infty} \mathbf{Tr}\left(\mathbf{C}^k\right)/k\right| \\ &\leq \left(1-\lambda_n\left(\mathbf{A}\right)\right)^m \left|\sum_{k=1}^{\infty} \mathbf{Tr}\left(\mathbf{C}^k\right)/k\right|. \end{aligned} \tag{3.28}$$

In the inequality of eqn. (3.27), we used the fact that $\lambda_1(\mathbf{C}) = 1-\lambda_n(\mathbf{A})$. Combining the bounds for Δ_1 and Δ_2 gives:

$$\left|\widehat{\text{logdet}}\left(\mathbf{A}\right) - \text{logdet}\left(\mathbf{A}\right)\right| \leq \left(\epsilon + \left(1-\lambda_n\left(\mathbf{A}\right)\right)^m\right) \cdot \sum_{k=1}^{\infty} \frac{\mathbf{Tr}\left(\mathbf{C}^k\right)}{k}.$$

We have already proven in Lemma 11 that

$$\sum_{k=1}^{\infty} \frac{\mathbf{Tr}\left(\mathbf{C}^k\right)}{k} = -\mathbf{Tr}\left(\log\left[\mathbf{A}\right]\right) = -\text{logdet}\left(\mathbf{A}\right).$$

Collecting our results, we get:

$$\left|\widehat{\text{logdet}}\left(\mathbf{A}\right) - \text{logdet}\left(\mathbf{A}\right)\right| \leq \left(\varepsilon + \left(1 - \lambda_n\left(\mathbf{A}\right)\right)^m\right) \cdot \left|\text{logdet}\left(\mathbf{A}\right)\right|.$$

Using $1 - \lambda_n(\mathbf{A}) < 1 - \theta_1$, we conclude that

$$\left|\widehat{\text{logdet}}\left(\mathbf{A}\right) - \text{logdet}\left(\mathbf{A}\right)\right| \leq \left(\varepsilon + \left(1 - \theta_1\right)^m\right) \cdot \left|\text{logdet}\left(\mathbf{A}\right)\right|.$$

Using an argument similar to the one used in eqn. (3.22) we find that by setting:

$$m = \left\lceil \frac{1}{\theta_1} \cdot \log\left(\frac{1}{\varepsilon}\right) \right\rceil$$

guarantees that $(1 - \theta_1)^m \leq \varepsilon$ and therefore:

$$\left|\widehat{\text{logdet}}\left(\mathbf{A}\right) - \text{logdet}\left(\mathbf{A}\right)\right| \leq 2\varepsilon \cdot \left|\text{logdet}\left(\mathbf{A}\right)\right|.$$

This concludes the proof. □

3.3.3 Running time

The running time of Algorithm 8, is equal to $\mathcal{O}\left(p \cdot m \cdot \text{nnz}\left(\mathbf{A}\right)\right)$. Given that $p = \mathcal{O}\left(\frac{\log(1/\delta)}{\varepsilon^2}\right)$ and $m = \mathcal{O}\left(\frac{\log(1/\varepsilon)}{\theta_1}\right)$, the running time of Algorithm 8 becomes

$$\mathcal{O}\left(\frac{\log(1/\varepsilon)\log(1/\delta)}{\varepsilon^2 \cdot \theta_1} \cdot \text{nnz}\left(\mathbf{A}\right)\right).$$

3.4 Empirical evaluation

The goal of our experimental section is to establish that our approximation to the logdet $(\mathbf{A})$ (as computed by Algorithm 7) is both accurate and fast for both dense and sparse matrices. The accuracy of Algorithm 7 is measured by comparing its result against the exact logdet $(\mathbf{A})$ computed via the Cholesky factorization.

We developed high-quality, shared- and distributed-memory parallel C++ code for the algorithms listed in this paper. All of the code that was developed for this paper is hosted in `http://web.ics.purdue.edu/~ekontopo/software.html`. In it's current state, our software supports: (1) ingesting dense (binary and text format) and sparse (binary, text, and matrix market format) matrices, (2) generating large random SPD matrices, (3) computing both approximate and exact spectral norms of matrices, (4) computing both approximate and exact traces of matrices, and (5) computing both approximate and exact log determinants of matrices. Currently, we support both Eigen [GB+10] and Elemental [PMVdG+13] matrices. As we wanted the random SPD generation to be fast, we have used parallel random number generators from Random123 [SMDS11] in conjunction with Boost.Random.

All our experiments ran on "Nadal", a 60-core machine, where each core is an Intel® Xeon® E7-4890 machine running at 2.8 Ghz. Nadal has 1 TB of RAM and runs Linux kernel version 2.6-32. For compilation, we used GCC 4.9.2. We used Eigen 3.2.4, OpenMPI 1.8.4, Boost 1.55.7, and the latest version of Elemental at `https://github.com/elemental`. For experiments with Elemental, we used OpenBlas, which is an extension of GotoBlas [GVDG08], for its parallel prowess; Eigen has built-in the BLAS and LAPACK packages.

3.4.1 Empirical results for dense matrices

In our experiments, we used two types of synthetic SPD matrices. The first type were diagonally dominant SPD matrices and were generated as follows. First, we created $\mathbf{X} \in \mathbb{R}^{n \times n}$ by drawing n^2 entries from a uniform sphere with center 0.5 and radius 0.25. Then, we generated a symmetric matrix $\mathbf{Y}$ by setting

$$\mathbf{Y} = 0.5 * (\mathbf{X} + \mathbf{X}^\top).$$

Finally, we ensured that the desired matrix $\mathbf{A}$ is positive definite by adding the value n to each diagonal entry [Cur09] of $\mathbf{Y}$: $\mathbf{A} = \mathbf{Y} + n\mathbf{I}_n$. We will call this method `randSPDDenseDD`.

The second approach generates SPD matrices that are not diagonally dominant. We created $\mathbf{X}, \mathbf{D} \in \mathbb{R}^{n \times n}$ by drawing n^2 and n entries, respectively, from a uniform sphere with center 0.5 and radius 0.25; $\mathbf{D}$ is a diagonal matrix with small entries. Next, we generated an orthogonal random matrix $\mathbf{Q}$ using the QR decomposition. Thus, $\mathbf{Q}$ is an orthonormal basis for $\operatorname{range}(\mathbf{X})$. Finally, we generated $\mathbf{A} = \mathbf{QDQ}^\top$. We call this method `randSPDDense`. `randSPDDense` is more expensive than `randSPDDenseDD`, as it requires an additional $\mathcal{O}(n^3)$ computations for the QR factorization and the matrix-matrix product.

To evaluate the runtime of Algorithm 7 against a baseline, we used the Cholesky decomposition to compute the $\operatorname{logdet}(\mathbf{A})$. More specifically, we computed $\mathbf{A} = \mathbf{LL}^\top$ and returned $\operatorname{logdet}(\mathbf{A}) = 2 \cdot \operatorname{logdet}(\mathbf{L})$. Since Elemental provides distributed and shared memory parallelism, we restricted ourselves to experiments with Elemental matrices throughout this section. Note that we measured the accuracy of the approximate algorithm in terms of the relative error to ensure that we have numbers of the same scale for matrices with vastly different values for $\operatorname{logdet}(\mathbf{A})$; we defined the relative error e as $e = 100(x - \tilde{x})/x$, where x is the true value and $\tilde{x}$ is the approximation. Similarly, we defined the speedup s as $s = t_x/t_{\tilde{x}}$, where t_x is the time needed to compute x and $t_{\tilde{x}}$ is the time needed to compute the approximation $\tilde{x}$.

We first tested dense synthetic matrices, generated using `randSPDDense`; these are relatively ill-conditioned matrices. We experimented with various matrix sizes. More specifically n, the number of rows and columns of $\mathbf{A}$, was set to get values from the set $\{5,000,\ 7,500,\ 10,000,\ 12,500,\ 15,000\}$.

Algorithm 7 has two profound sources of accuracy loss. The first one comes from the truncation of the Taylor series, while the second from the trace estimation. We are particularly interested in observing how the number of Taylor terms, m and the number of random Gaussian vectors, p are influencing the quality fo the approximation.

First, we discuss the effect of m, the number of Taylor terms; Figure 3.1 depicts our results for the sequential case (the number of processors was set to one) when using $p = 60$ random Gaussian vectors and $t = 2 \log \sqrt{4n}$ power method steps. On the y-axis, we observe the relative error, which is measured against the exact $\operatorname{logdet}(\mathbf{A})$ as computed via the Cholesky factorization. We observe that for these ill-conditioned matrices and small values of m (typically less than four) the relative error is high, indicating the dependence of the approximation to the condition number of input matrix as well as to the number of Taylor terms. However, for all values of $m \geq 4$, we observe that the error drops significantly and stabilizes showing that as m grows the effect of the condition number on the accuracy of the approximation is diminishing. This however, comes with an increase in the running time as observed in Figure 3.2. We note that in each iteration, all random processes were re-seeded with new values; we have plotted the error bars Figure 3.1. The standard deviation for both the accuracy and the running time was consistently small; indeed, it is not visible to the naked eye at scale. To see the benefit of approximation, we compare Figure 3.2 with Figure 3.1. For example, at $m = 4$, for all matrices, we get at least a factor of two speedup. As n gets larger, the speedup also increases. For example, for $n = 15,000$, the speedup at $m = 4$ is nearly six-fold. In terms of accuracy, Figure 3.1 shows that at $m = 4$, the relative error is approximately 4%. This speedup is expected as the Cholesky factorization requires $\mathcal{O}(n^3)$ operations; Algorithm 7 only relies on matrix-vector products, the number of which is independent of n.

Finally, we discuss the parallel speedup in Figure 3.3, which shows the relative speedup of the approximate algorithm with respect to the baseline Cholesky algorithm. For this evaluation, we set $m = 4$ and varied the number of processes, denoted by np, from one to 60. The main take away message from Figure 3.3 is that the approximate algorithm provides nearly the same or increasingly better speedups when compared to a parallelized version of the exact (Cholesky) algorithm. For example,

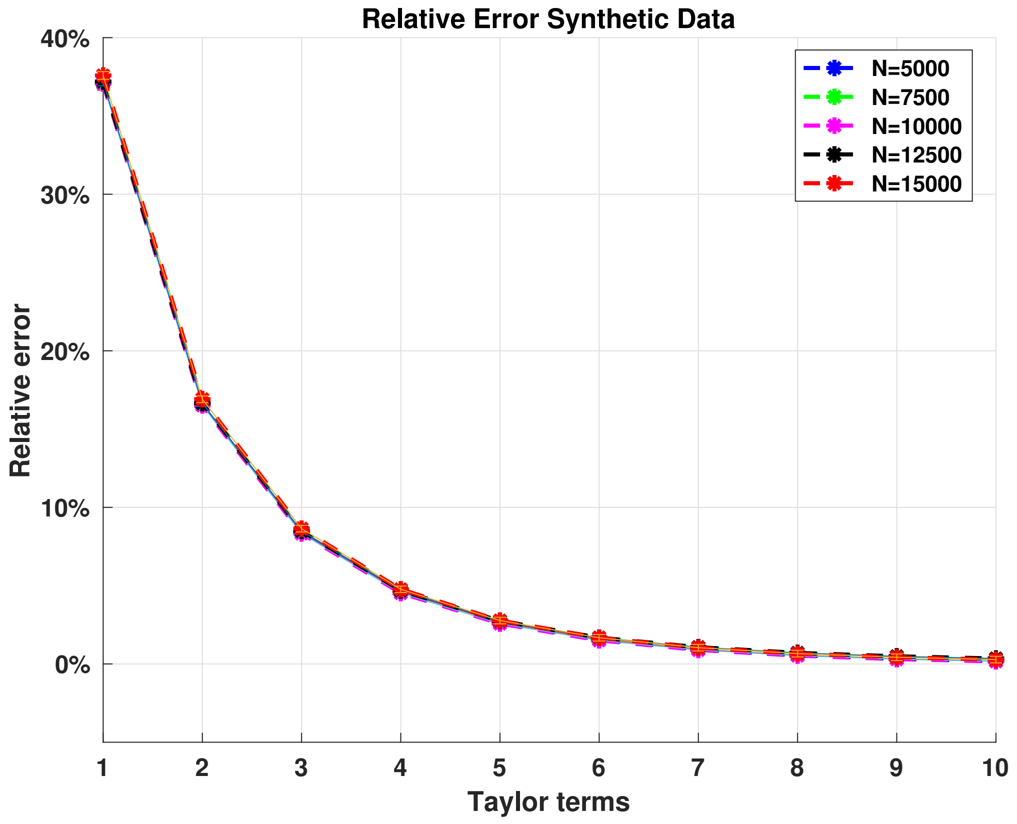


Figure 3.1.: Relative error for $N \times N$ synthetic dense SPD matrices generated by `randSPDDense`, using $p = 60$ random Gaussian vectors and $t = 2\log\sqrt{4n}$ power method steps. The number of processors was set to one.

for $n = 15,000$, the speedup for using the approximate algorithm is consistently around $6.5x$. Figure 3.3 further indicates that our method with dense inputs can be fast without consuming a significant amount of computational resources. In Table 3.1 we report the values of $\operatorname{logdet}(\mathbf{A})$ and the corresponding run time when running the implementation of Algorithm 7, along with the the baseline as computed using the Cholesky factorization. The reported numbers are the ones observed when $m = 4$ Taylor terms.

For the second set of dense experiments, we generated diagonally dominant matrices using `randSPDDenseDD`; we were able to quickly generate and run benchmarks on matrices of sizes $N \times N$ with N in the set $\{10,000,\ 20,000,\ 30,000,\ 40,000\}$ due to

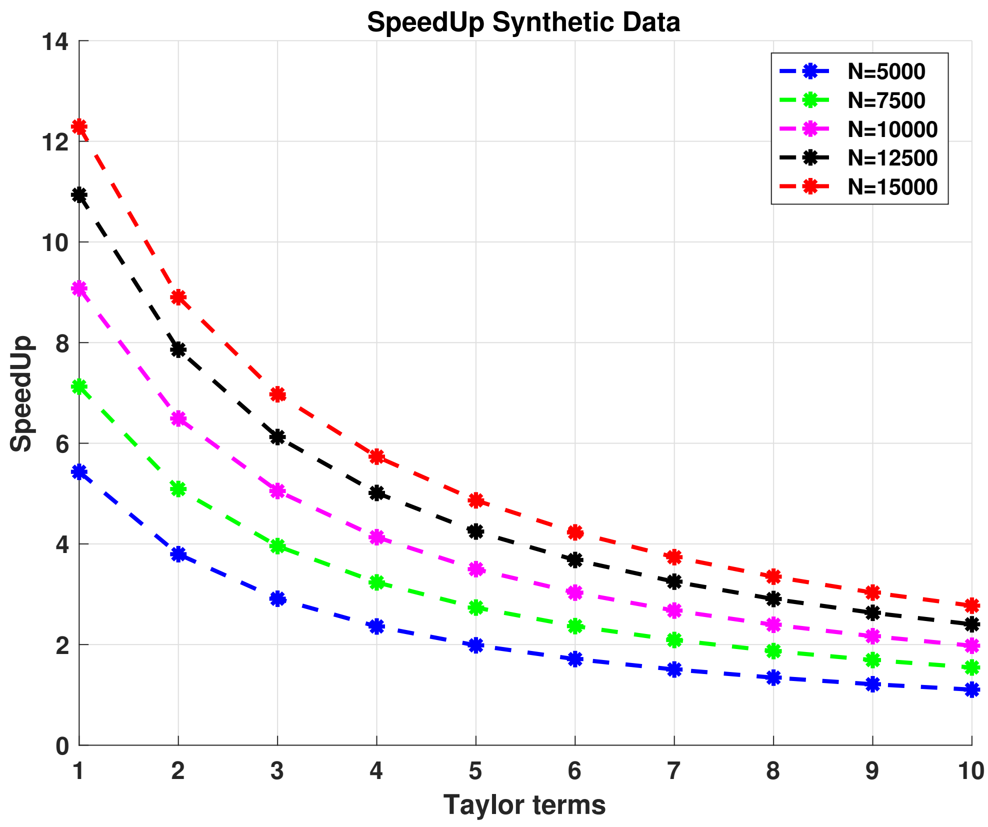


Figure 3.2.: Speedup for $N \times N$ synthetic dense SPD matrices generated by `randSPDDense`, using $p = 60$ random Gaussian vectors and $t = 2\log\sqrt{4n}$ power method steps. The number of processors was set to one.

Table 3.1.: Results and sequential running times when $p = 60$, $m = 4$ and $t = \log\sqrt{4n}$ for synthetic SPD matrices generated using `randSPDDense`. The mean and standard deviation are reported over ten iterations.

n	logdet $(\mathbf{A})$			time (secs)		
	exact	mean	std	exact	mean	std
5000	-3717.89	-3546.920	8.10	2.56	1.15	0.0005
7500	-5474.49	-5225.152	8.73	7.98	2.53	0.0015
10000	-7347.33	-7003.086	7.79	18.07	4.47	0.0006
12500	-9167.47	-8734.956	17.43	34.39	7.00	0.0030
15000	-11100.9	-10575.16	15.09	58.28	10.39	0.0102

the relatively simpler procedure involved in matrix generation. In this set of experiments, due to the diagonal dominance, all matrices were well-conditioned. The results

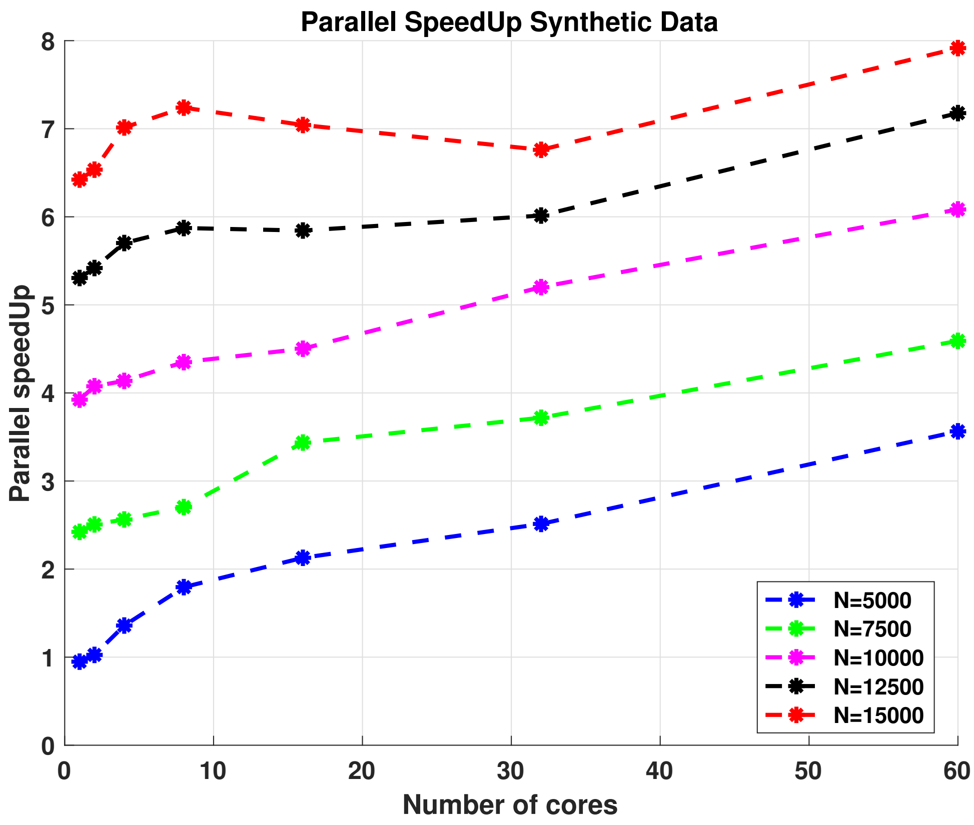


Figure 3.3.: Parallel speedup for $N \times N$ synthetic dense SPD matrices generated by `randSPDDense`, using $m = 4$ Taylor terms, $p = 60$ random Gaussian vectors and $t = 2\log\sqrt{4n}$ power method steps.

of our experiments on the well-conditioned matrices are presented in Figures 3.4, 3.5 and 3.6. It's clear that there is a noticeable improvement over the results presented in Figures 3.1, 3.2 and 3.3. First, it is obvious from Figure 3.4 that very few terms of the Taylor series (i.e., small m) are sufficient to get highly accurate approximations. In fact, we see that even at $m = 2$, we are approaching 0% and at $m = 3$, for most of the matrices, we have near-zero percent relative error. Recall that, according to our theoretical analysis, both the error bound as well as the bound for m depend on the condition number of the matrix. This experimental result, combined with Figure 3.5 is particularly encouraging; for example at $m = 2$, we seem to not only have a nearly lossless approximation of $\operatorname{logdet}(\mathbf{A})$, but also have at least a $5x$ speedup. Similarly to Figure 3.2, the speedup improves as the size of the matrix increases. For example, for $N = 40,000$, the speedup at $m = 2$ is nearly $20x$.

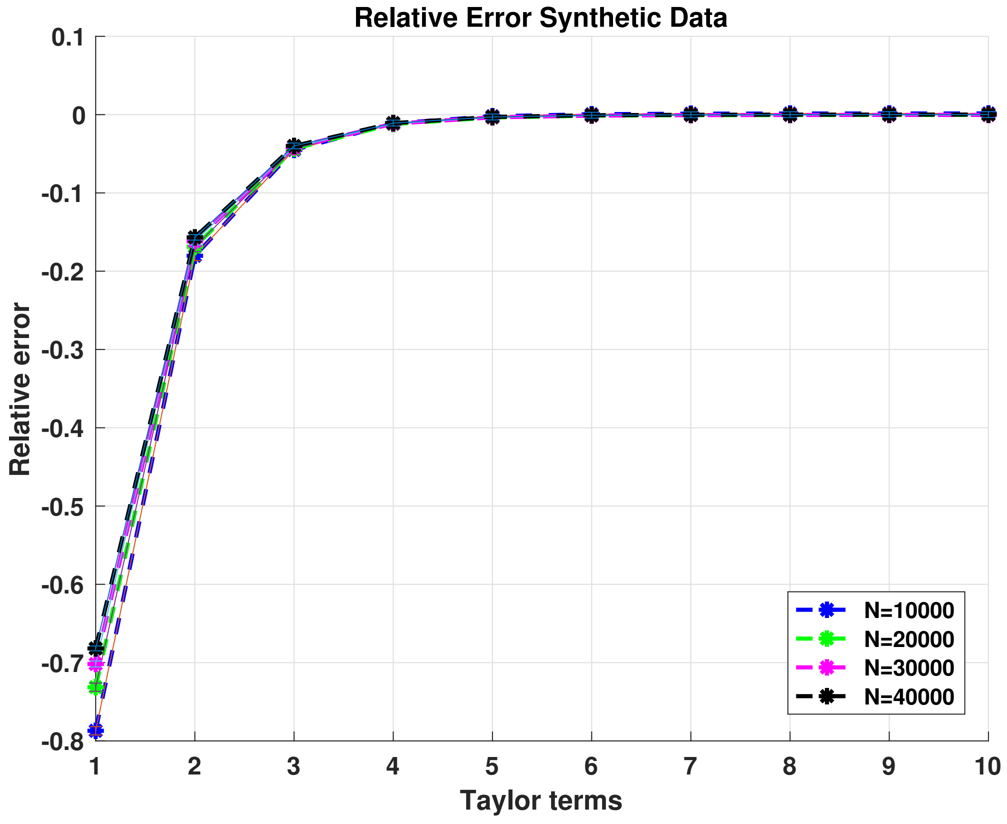


Figure 3.4.: Relative error for $N \times N$ synthetic dense diagonally dominant SPD matrices generated by `randSPDDenseDD`, using $p = 60$ random Gaussian vectors and $t = 2\log\sqrt{4n}$ power method steps. The number of processors was set to one.

We conclude our analysis by presenting Figure 3.6, which similarly to Figure 3.3, points out that at any level of parallelism, Algorithm 7 maintains its relative performance over the exact (Cholesky) factorization.

In Table 3.2 we report the values of $\operatorname{logdet}(\mathbf{A})$ and the corresponding running time when running the implementation of Algorithm 7, along with the the baseline as computed using the Cholesky factorization. The reported numbers are the ones observed when $m = 2$ Taylor terms.

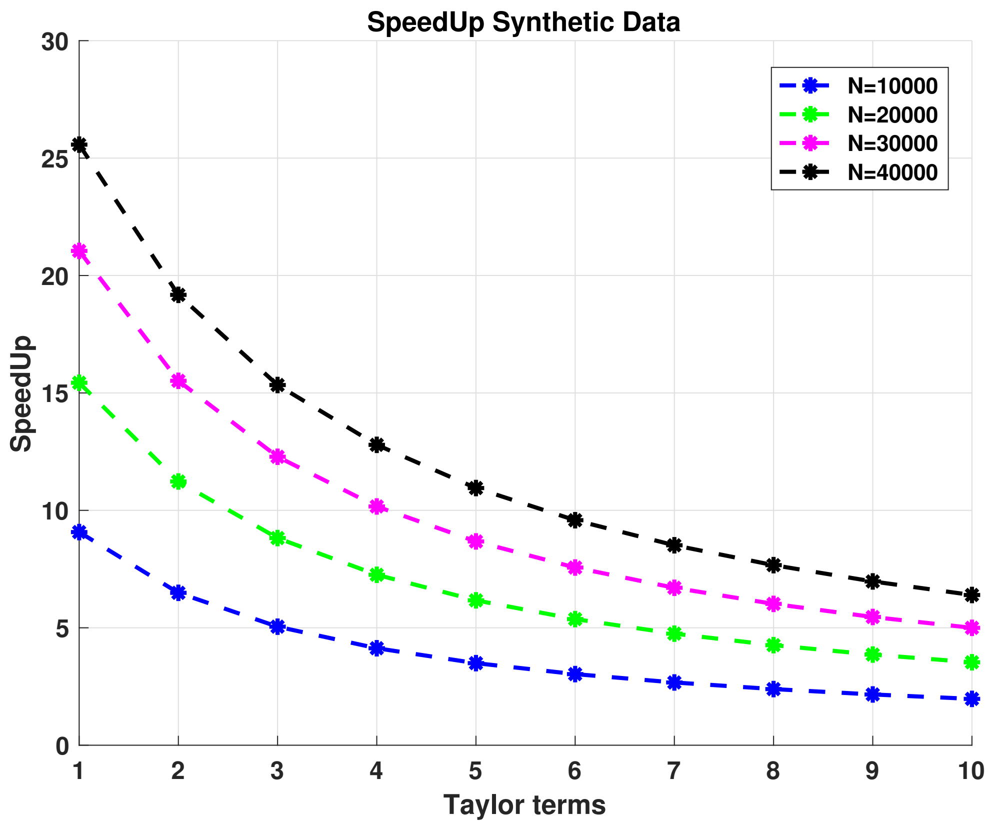


Figure 3.5.: Speedup for $N \times N$ synthetic diagonally dominant dense SPD matrices generated by `randSPDDenseDD`, using $p = 60$ random Gaussian vectors and $t = 2\log\sqrt{4n}$ power method steps. The number of processors was set to one.

Table 3.2.: Results and sequential running times when $p = 60$, $m = 2$, and $t = 2\log\sqrt{4n}$ for diagonally dominant dense random matrices generated using `randSPDDenseDD`. The mean and standard deviation are reported over ten iterations.

n	logdet $(\mathbf{A})$			time (secs)		
	exact	mean	std	exact	mean	std
10000	92103.1	92269.5	5.51	18.09	2.87	0.01
20000	198069.0	198397.4	9.60	135.92	12.41	0.02
30000	309268.0	309763.8	20.04	448.02	30.00	0.12
40000	423865.0	424522.4	14.80	1043.74	58.05	0.05

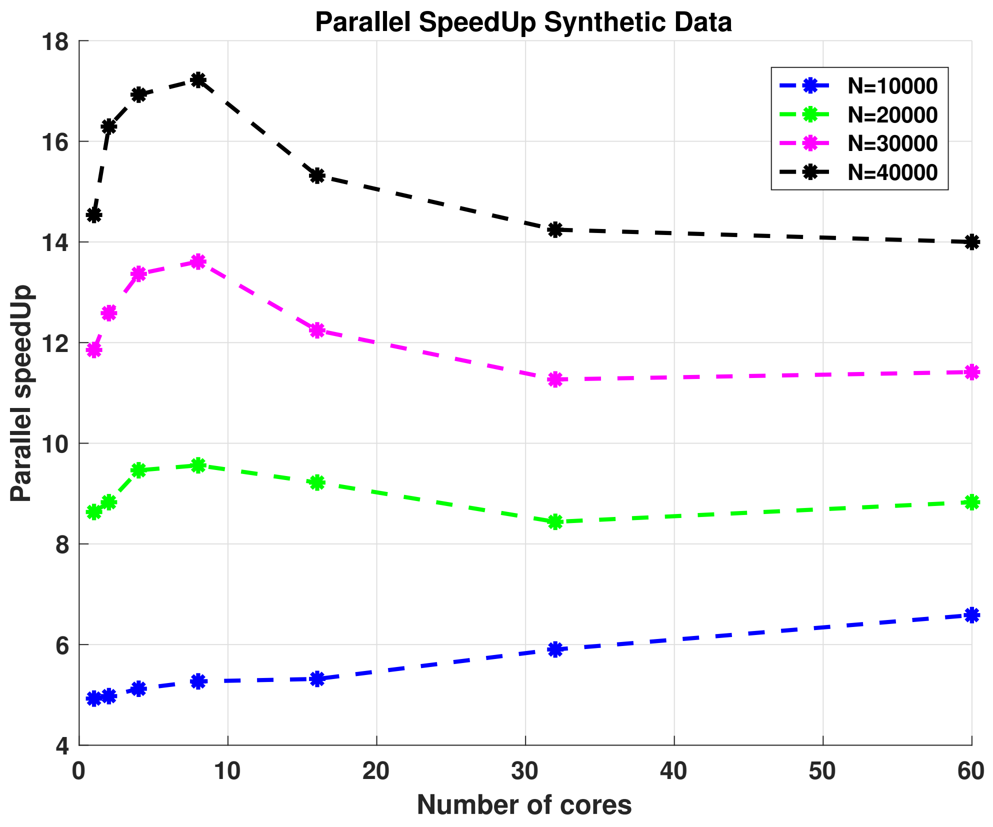


Figure 3.6.: Parallel speedup for $N \times N$ synthetic dense diagonally dominant SPD matrices generated by `randSPDDenseDD`, using $m = 2$ Taylor terms, $p = 60$ random Gaussian vectors and $t = 2\log\sqrt{4n}$ power method steps.

3.4.2 Empirical results for sparse matrices

To generate a sparse, synthetic matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, with $\mathrm{nnz}\left(\mathbf{A}\right)$ non-zeros, we use a randomized rounding technique based on a Bernoulli distribution to determine the location of the non-zero entries and a uniform distribution to generate the values. First, we completely fill the n principle diagonal entries. Next, we generate index positions equal to $\mathrm{pca}\left(\mathrm{nnz}\left(\mathbf{A}\right) - \mathrm{n}\right)/2$ in the upper triangle for the non-zero entries by sampling from a Bernoulli distribution with probability $(\mathrm{nnz}\left(\mathbf{A}\right) - \mathrm{n})/(\mathrm{n}^2 - \mathrm{n})$. We reflect each entry across the principle diagonal to ensure that $\mathbf{A}$ is symmetric and we add n to each diagonal entry to ensure that $\mathbf{A}$ is SPD (actually, $\mathbf{A}$ is also diagonally dominant).

To demonstrate the prowess of Algorithm 7 on real-world data, we used SPD matrices from the University of Florida's sparse matrix collection [DH11]. The complete list of matrices from this collection used in our experiments, as well as a brief description of each matrix, is given in columns one through four of Table 3.3. It is tricky to pick any single method as the "exact method" to compute the $\operatorname{logdet}(\mathbf{A})$ for a sparse SPD matrix $\mathbf{A}$. One approach would be to use direct methods such as Cholesky decomposition of $\mathbf{A}$ [Dav06, Gup00]. For direct methods, it is difficult to derive an analytical solution for the number of operations required for the factorization as a function of the number of non-zero entries of the matrix, as this is highly dependent on the structure of the matrix [GKK97]. In the distributed setting, one also needs to consider the volume of communication involved, which is often the bottleneck. Alternately, we can use iterative methods to compute the eigenvalues of $\mathbf{A}$ [Dav75] and use the eigenvalues to compute $\operatorname{logdet}(\mathbf{A})$. It is clear that the worst case performance of both the direct and iterative methods is $\mathcal{O}(n^3)$. However, iterative methods are typically used to compute a few eigenvalues and eigenvectors: therefore, we chose to use the Cholesky factorization based on matrix reordering to compute the exact value of $\operatorname{logdet}(\mathbf{A})$. It is important to note that both the direct and iterative methods are notoriously hard to implement, which comes to stark contrast with the almost trivial implementation of Algorithm 7, which is also embarrassingly parallelizable.

The true power of Algorithm 7 lies in its ability to approximate $\operatorname{logdet}(\mathbf{A})$ for sparse $\mathbf{A}$. The Cholesky factorization, as most direct methods for factorizing a matrix, is prone to fill-in and thus for many problems, there is insufficient memory to factorize a large, sparse matrix. In our first set of experiments, we wanted to show the effect of m on: (i) the convergence of $\operatorname{logdet}(\mathbf{A})$, and (ii) the cost of the solution. To this end, we generated sparse, diagonally dominant SPD matrices of size $n = 10^5$ and varied the sparsity from 0.1% to 1% in increments of 0.25%. We did not compute the exact $\operatorname{logdet}(\mathbf{A})$ for these synthetic matrices merely because the aim of this experiment is to study the speedup with m for various sparsity ratios. The values of t, the number of power method steps and p, the number of random Gaussian vectors, were held constant at $2\log\sqrt{4n}$ and 60 respectively. Figure 3.7 depicts the convergence of $\operatorname{logdet}(\mathbf{A})$ measured as a relative error of the current estimate over the final estimate. As can be seen — for well conditioned matrices – convergence is fast. Figure 3.8 shows the relative cost of increasing m; using $m = 1$ as the baseline.

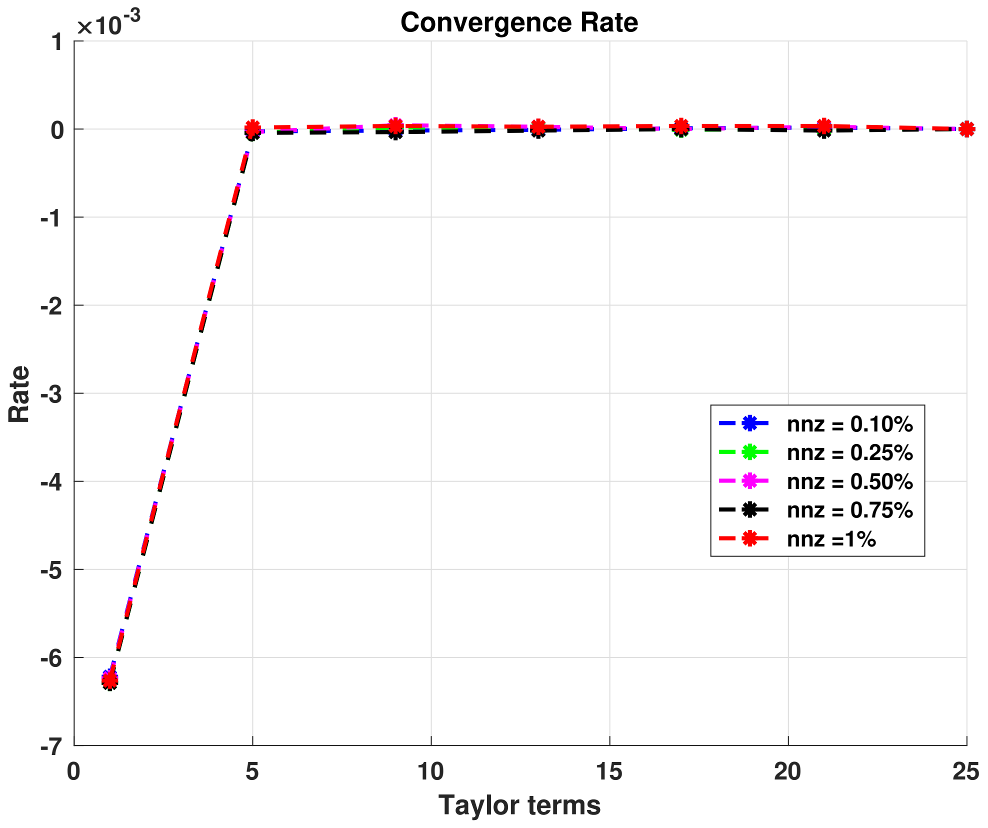


Figure 3.7.: Convergence rate of Algorithm 7, on synthetic sparse inputs, to the final value of $\operatorname{logdet}(\mathbf{A})$ as observed at $m = 25$ Taylor terms. The number of random Gaussian vectors was fixed to $p = 60$, and the number of power method steps was fixed to $t = 2\log\sqrt{4n}$. The number of processors was set to one.

It seems that the additional cost incurred by increasing m is linear when all other parameters are held constant.

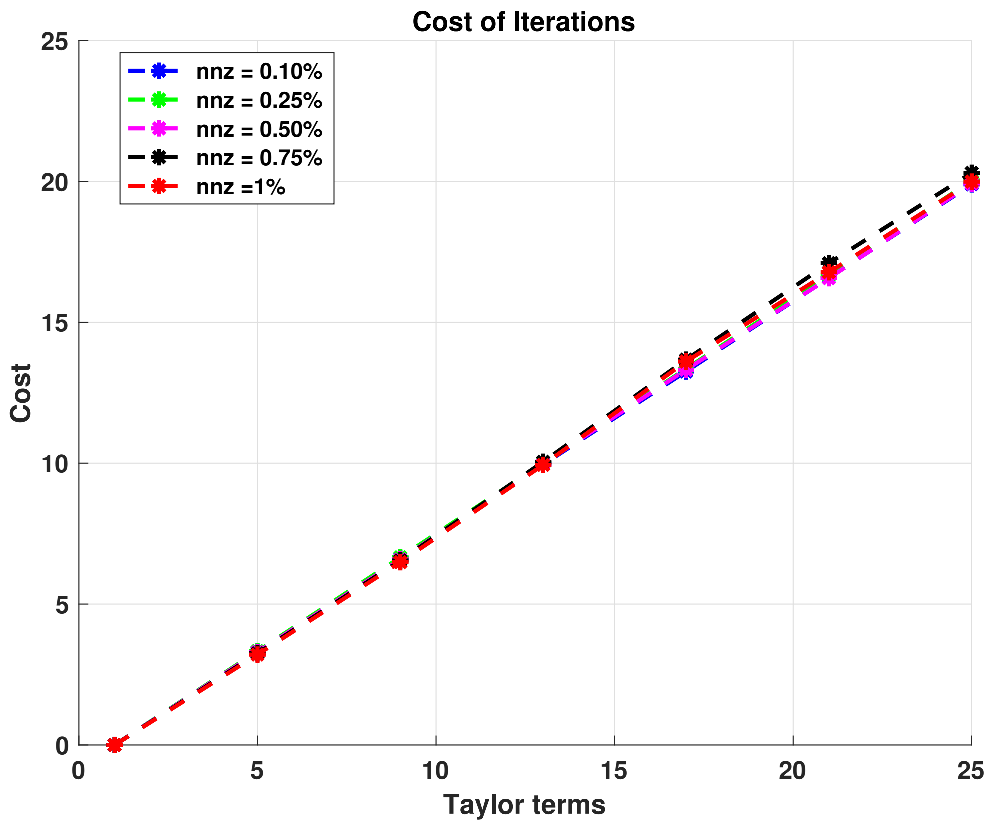


Figure 3.8.: Relative cost of iterations of Algorithm 7, on synthetic sparse inputs with $p = 60$, random Gaussian vectors and $t = 2\log\sqrt{4n}$, power method steps. The baseline was set to $m = 1$ Taylor terms.The number of processors was set to one.

The results of running Algorithm 7 on the UFL matrices are shown in Table 3.3. The numbers reported for the approximation are the mean and standard deviation over ten iterations, for $t = 5$ power method steps, and $p = 5$ random Gaussian vectors[4]. The value of m varied between one and 150 in increments of five, to select the best average accuracy. The matrices shown in Table 3.3 have a nice structure, which results in easy reorderings and consequently an efficient computation of the Cholesky factorization. We see that even in such cases, the performance of Algorithm 7 is

[4]We experimented with various values of p and t and picked the smallest values that did not result in accuracy loss.

commendable due to its lower algorithmic complexity. In the case of `thermomech_TC`, we achieve good accuracy while achieving a $22x$ speedup.

Table 3.3.: Results and running time of real world SPD matrices from the University of Florida sparse matrix collection [DH11]. The mean and the standard deviation were taken over ten iterations at using $t = 5$ and $p = 5$; we only report the mean for the running times, since the standard deviation is negligible.

name	n	nnz	area of origin	logdet ($\mathbf{A}$)			time (sec)		m
				exact	approx		exact	approx	
					mean	std		mean	
thermal2	1228045	8580313	Thermal	1.3869e6	1.3928e6	964.79	31.28	31.24	149
ecology2	999999	4995991	2D/3D	3.3943e6	3.403e6	1212.8	18.5	10.47	125
ldoor	952203	42493817	Structural	1.4429e7	1.4445e7	1683.5	117.91	17.60	33
thermomech_TC	102158	711558	Thermal	-546787	-546829.4	553.12	57.84	2.58	77
boneS01	127224	5516602	Model reduction	1.1093e6	1.106e6	247.14	130.4	8.48	125

4 ESTIMATION OF THE VON NEUMANN ENTROPY OF DENSITY MATRICES

This chapter presents our algorithms for the approximation of Von Neumann Entropy of a density matrix. Throughout the chapter we will use the notation $\mathcal{H}(\mathbf{A})$ to refer to the actual Von Neumann entropy of the density matrix $\mathbf{A}$ and the notation $\widehat{\mathcal{H}}(\mathbf{A})$ to refer to its approximation.

Our first two algorithms, that will also be addressed as polynomial-based algorithms achieve with high probability relative error guarantees for the approximation of the Von Neumann entropy of **real and complex** density matrices with a full set of pure states. Our third set of algorithms, that will be also be addressed as random-projection-based algorithms, achieve additive-relative error guarantees for the approximation of the Von Neumann entropy of **real** density matrices that do not necessarily have a full set of pure states.

The chapter is organized as follows: in Section 4.1 we provide the mathematical setting under which we will work and any essential information that will be useful through out the chapter. The polynomial-based algorithms are described in Sections 4.2 and 4.3. The random-projection-based algorithms are described in Section 4.5. Finally, Section 4.6 provides the empirical evaluation of our algorithms on a variety of datasets.

Sections of chapter 4 have been published in [KDS+20]

Randomized Linear Algebra Approaches to Estimate the Von Neumann Entropy of Density Matrices E-M. Kontopoulou, G. Dexter, A. Grama, W. Szpankowski, P. Drineas in IEEE Transactions on Information Theory (2020), to appear

4.1 Setting

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

$$\mathbf{R} = \sum_{i=1}^{k} p_i \boldsymbol{\psi}_i \boldsymbol{\psi}_i^\top \in \mathbb{R}^{n \times n}. \tag{4.1}$$

The vectors $\boldsymbol{\psi}_i \in \mathbb{R}^n$ for $i = 1 \dots k$ represent the $k \leq n$ pure states and can be assumed to be pairwise orthogonal and normal, while p_i's correspond to the probability of each state and satisfy $p_i > 0$ and $\sum_{i=1}^{k} p_i = 1$. From a linear algebraic perspective, eqn. (4.1) can be rewritten as

$$\mathbf{R} = \boldsymbol{\Psi} \boldsymbol{\Sigma}_p \boldsymbol{\Psi}^\top \in \mathbb{R}^{n \times n}, \tag{4.2}$$

where $\boldsymbol{\Psi} \in \mathbb{R}^{n \times k}$ is the orthonormal matrix of the vectors $\boldsymbol{\psi}_i$ and $\boldsymbol{\Sigma}_p \in \mathbb{R}^{k \times k}$ is a diagonal matrix whose entries are the (positive) p_i's. Given our assumptions for $\boldsymbol{\psi}_i$, $\boldsymbol{\Psi}^\top \boldsymbol{\Psi} = \mathbf{I}$; also $\mathbf{R}$ is symmetric positive semi-definite with its eigenvalues equal to p_i and corresponding left/ right singular vectors equal to $\boldsymbol{\psi}_i$'s; and $\mathbf{Tr}(\mathbf{R}) = \sum_{i=1}^{k} p_i = 1$. Notice that eqn. (4.2) essentially reveals the (thin) Singular Value Decomposition (SVD) [GV96] of $\mathbf{R}$. The Von Neumann entropy of $\mathbf{R}$, denoted by $\mathcal{H}(\mathbf{R})$ is equal to (see, also eqn. (1.8))

$$\begin{aligned} \mathcal{H}(\mathbf{R}) &= -\sum_{i:p_i>0} p_i \log p_i \\ &= -\mathbf{Tr}(\mathbf{R} \log [\mathbf{R}]) . \end{aligned} \tag{4.3}$$

Eqn. (4.3) follows from the definition of matrix functions (see, Section 2.2.6). More precisely, we overload notation and consider the full SVD of $\mathbf{R}$, namely $\mathbf{R} = \boldsymbol{\Psi} \boldsymbol{\Sigma}_p \boldsymbol{\Psi}^\top$, where $\boldsymbol{\Psi} \in \mathbb{R}^{n \times n}$ is an orthogonal matrix whose top k columns correspond to the k pure states and the bottom $n - k$ columns are chosen so that $\boldsymbol{\Psi} \boldsymbol{\Psi}^\top = \boldsymbol{\Psi}^\top \boldsymbol{\Psi} = \mathbf{I}_n$.

Here $\mathbf{\Sigma}_p$ is a diagonal matrix whose bottom $n-k$ diagonal entries are set to zero. Let $h(x) = x \log x$ for any $x > 0$ and let $h(0) = 0$. Then,

$$\begin{aligned}
-\sum_{i,p_i>0} p_i \log p_i &= -\,\mathbf{Tr}\left(h(\mathbf{\Sigma}_p)\right) \\
&= -\,\mathbf{Tr}\left(h(\mathbf{\Sigma}_p)\mathbf{\Psi}^\top\mathbf{\Psi}\right) \\
&= -\,\mathbf{Tr}\left(\mathbf{\Psi} h(\mathbf{\Sigma}_p)\mathbf{\Psi}^\top\right) \qquad (4.4) \\
&= -\,\mathbf{Tr}\left(h(\mathbf{R})\right) \\
&= -\,\mathbf{Tr}\left(\mathbf{R}\log\left[\mathbf{R}\right]\right), \qquad (4.5)
\end{aligned}$$

where eqn. (4.4) follows from the circular property of the trace operator and eqn. (4.5) follows from the definition of $h(x)$.

4.2 An approach via Taylor series

Our first approach to approximate the Von Neumann entropy of a density matrix uses a Taylor series expansion to approximate the logarithm of a matrix, combined with a relative-error trace estimator for symmetric positive semi-definite matrices and the power method to upper bound the largest singular value of a matrix. Lemma 13 is the starting point of our main algorithm for approximating the Von Neumann entropy of a density matrix. In words Lemma 13 states that the computation of the Von Neumann entropy of a density matrix $\mathbf{R}$ with a full set of pure states, boils down to a two-fold task; first, computing the largest eigenvalue of $\mathbf{R}$ and second, computing the trace all the powers of a matrix that is related to $\mathbf{R}$.

Lemma 13. *Let $\mathbf{R} \in \mathbb{R}^{n\times n}$ be a symmetric positive definite matrix with unit trace and whose eigenvalues lie in the interval $[\ell, u]$, for some $0 < \ell \leq u < 1$. Then,*

$$\mathcal{H}\left(\mathbf{R}\right) = \log u^{-1} + \sum_{k=1}^{\infty} \frac{\mathbf{Tr}\left(\mathbf{R}(\mathbf{I}_n - u^{-1}\mathbf{R})^k\right)}{k}. \qquad (4.6)$$

Algorithm 9 High Level Procedure for Von Neumann Entropy Computation

```
1: function VNEntropy($\mathbf{R}$)
2:     Compute $u$ such that $p_1(\mathbf{R}) \leq u$.
3:     $\mathbf{C} \leftarrow (\mathbf{I}_n - \mathbf{R}/u)$.
4:     $Sum \leftarrow 0$.
5:     for $k \leftarrow 1, \ldots$ do
6:         $Sum \leftarrow Sum + \mathbf{Tr}\left(\mathbf{RC}^k\right)/k$.
7:     end for
8:     $\mathcal{H}(\mathbf{R}) \leftarrow \log(1/u) + Sum$.
9:     return $\mathcal{H}(\mathbf{R})$.
10: end function
```

Proof. From the definition of the Von Neumann entropy and a Taylor expansion,

$$\begin{aligned} \mathcal{H}(\mathbf{R}) &= -\mathbf{Tr}\left(\mathbf{R}\log\left(uu^{-1}\mathbf{R}\right)\right) \\ &= -\mathbf{Tr}\left((\log u)\mathbf{R}\right) - \mathbf{Tr}\left(\mathbf{R}\log(\mathbf{I}_n - (\mathbf{I}_n - u^{-1}\mathbf{R}))\right) \\ &= \log u^{-1} - \mathbf{Tr}\left(-\mathbf{R}\sum_{k=1}^{\infty}\frac{(\mathbf{I}_n - u^{-1}\mathbf{R})^k}{k}\right) \qquad (4.7) \\ &= \log u^{-1} + \sum_{k=1}^{\infty}\frac{\mathbf{Tr}\left(\mathbf{R}(\mathbf{I}_n - u^{-1}\mathbf{R})^k\right)}{k}. \end{aligned}$$

The first term of eqn. (4.7) follows since $\mathbf{R}$ has unit trace while the second term of eqn. (4.7) comes from the Taylor expansion of $\log[\mathbf{I}_n - \mathbf{A}]$ as expressed in Definition 16. It is then the case that the eigenvalues of $\mathbf{I}_n - u^{-1}\mathbf{R}$ are all in the interval $(0, 1-(\ell/u)]$, whose upper bound is strictly less than one since, by our assumptions, $\ell/u > 0$. □

4.2.1 Algorithm

Lemma 13 indicates a high-level function for computing the Von Neumann entropy of a density matrix $\mathbf{R}$ that is described in Algorithm 9.

To implement step 2 we use the power method from the numerical linear algebra literature (see, Section 2.5.1). Step 3 is straightforward. To implement step 5, we

Algorithm 10 A Taylor series approach to estimate the Von Neumann entropy.

```
Input: $\mathbf{R} \in \mathbb{R}^{n\times n}$, accuracy parameter $\varepsilon > 0$, failure probability $\delta$, and integer
    $m > 0$.
Output: $\widehat{\mathcal{H}}(\mathbf{R})$, the approximation of $\mathcal{H}(\mathbf{R})$.
1: Compute $\tilde{p}_1$, the estimate of the largest eigenvalue of $\mathbf{R}$, $p_1$, using Algorithm 2
   with $t = \mathcal{O}(\log n)$ and $q = \mathcal{O}(\log(1/\delta))$.
2: $u = \min\{1, 6\tilde{p}_1\}$.
3: $s = \lceil 20\log(2/\delta)/\varepsilon^2 \rceil$.
4: Generate $\mathbf{g}_1, \mathbf{g}_2, \dots, \mathbf{g}_s \in \mathbb{R}^n$ i.i.d. random Gaussian vectors.
5: return
   $$\widehat{\mathcal{H}}(\mathbf{R}) = \log u^{-1} + \frac{1}{s}\sum_{i=1}^{s}\sum_{k=1}^{m}\frac{\mathbf{g}_i^\top \mathbf{R}(\mathbf{I}_n - u^{-1}\mathbf{R})^k \mathbf{g}_i}{k}.$$
```

truncate the expansion $\sum_{k=1}^{\infty} \mathbf{Tr}\left(\mathbf{R}\mathbf{C}^k\right)$ keeping few of the summands. This step is important since the quality of the approximation, both theoretically and empirically, depends on the number of summands (denoted by m) that will be kept. On the other hand, the running time of the algorithm increases with m. Finally, to estimate the traces of the powers of $\mathbf{C}$ times $\mathbf{R}$, we use the randomized algorithm of Section 2.5.2. Our approach is described in detail in Algorithm 10. We note that step 4 of Algorithm 10 can be efficiently implemented as it was indicated in step 5 of Algorithm 7 in Section 3.2.1.

4.2.2 Error bound

Theorem 4 is our main quality-of-approximation result for Algorithm 10. It essentially proves that Algorithm 10 returns an accurate approximation to $\mathcal{H}(\mathbf{R})$.

Theorem 4. *Let* $\mathbf{R}$ *be a density matrix such that all probabilities* p_i, $i = 1 \dots n$ *satisfy* $0 < \ell \leq p_i$. *Let* u *be computed as in Algorithm 10 and let* $\widehat{\mathcal{H}}(\mathbf{R})$ *be the output of Algorithm 10 on inputs* $\mathbf{R}$, m, *and* $\varepsilon < 1$; *Then, with probability at least* $1 - 2\delta$,

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

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

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

Proof. We begin by making the assumption that our analysis on Algorithm 2 is successful, which happens with probability at least $1-\delta$. In this case, $u = \min\{1, 6\tilde{p}_1\}$ is an upper bound for all probabilities p_i. For notational convenience, set $\mathbf{C} = \mathbf{I}_n - u^{-1}\mathbf{R}$. We start by manipulating $\Delta = \left|\widehat{\mathcal{H}}(\mathbf{R}) - \mathcal{H}(\mathbf{R})\right|$ as follows:

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

We now bound the two terms Δ_1 and Δ_2 separately. We start with Δ_1: the idea is to apply Lemma 5 on the matrix $\sum_{k=1}^{m} \mathbf{R}\mathbf{C}^k/k$ with $s = \lceil 20\log(2/\delta)/\varepsilon^2 \rceil$. Hence, with probability at least $1-\delta$:

$$\begin{aligned}
\Delta_1 &\leq \varepsilon \cdot \mathbf{Tr}\left(\sum_{k=1}^{m} \mathbf{R}\mathbf{C}^k/k\right) && (4.8) \\
&\leq \varepsilon \cdot \mathbf{Tr}\left(\sum_{k=1}^{\infty} \mathbf{R}\mathbf{C}^k/k\right). && (4.9)
\end{aligned}$$

A subtle point in applying Lemma 5 is that the matrix $\sum_{k=1}^{m} \mathbf{R}\mathbf{C}^k/k$ must be SPSD To prove that $\sum_{k=1}^{m} \mathbf{R}\mathbf{C}^k/k$ is SPSD we first define the SVD of $\mathbf{R}$ be $\mathbf{R} = \mathbf{\Psi}\mathbf{\Sigma}_p\mathbf{\Psi}^\top$, where all three matrices are in $\mathbb{R}^{n\times n}$ and the diagonal entries of $\mathbf{\Sigma}_p$ are in the interval $[\ell, u]$. Then, it is easy to see that

$$\mathbf{C} = \mathbf{I}_n - u^{-1}\mathbf{R} = \mathbf{\Psi}(\mathbf{I}_n - u^{-1}\mathbf{\Sigma}_p)\mathbf{\Psi}^\top$$

and

$$\mathbf{RC}^k = \mathbf{\Psi}\mathbf{\Sigma}_p(\mathbf{I}_n - u^{-1}\mathbf{\Sigma}_p)^k\mathbf{\Psi}^\top.$$

This proves that the matrix $\mathbf{RC}^k$ is SPSD for any k, a fact which will be useful throughout the proof. Now,

$$\sum_{k=1}^{m}\mathbf{RC}^k/k = \mathbf{\Psi}\underbrace{\left(\mathbf{\Sigma}_p\sum_{k=1}^{m}(\mathbf{I}_n - u^{-1}\mathbf{\Sigma}_p)^k/k\right)}_{\mathbf{W}}\mathbf{\Psi}^\top,$$

which shows that $\mathbf{W}$ is also a SPSD matrix which in turn validates the inequality of eqn. (4.8). Additionally, since $\mathbf{RC}^k$ is SPSD, its trace is non-negative, which proves the inequality of eqn. (4.9). We proceed to bound Δ_2 as follows:

$$\begin{aligned}
\Delta_2 &= \left|\sum_{k=m+1}^{\infty}\mathbf{Tr}\left(\mathbf{RC}^k\right)/k\right| = \left|\sum_{k=m+1}^{\infty}\mathbf{Tr}\left(\mathbf{RC}^m\mathbf{C}^{k-m}\right)/k\right| \\
&= \left|\sum_{k=m+1}^{\infty}\mathbf{Tr}\left(\mathbf{C}^m\mathbf{C}^{k-m}\mathbf{R}\right)/k\right| \leq \left|\sum_{k=m+1}^{\infty}\|\mathbf{C}^m\|_2\cdot\mathbf{Tr}\left(\mathbf{C}^{k-m}\mathbf{R}\right)/k\right| \qquad (4.10) \\
&= \|\mathbf{C}^m\|_2\cdot\left|\sum_{k=m+1}^{\infty}\mathbf{Tr}\left(\mathbf{RC}^{k-m}\right)/k\right| \leq \|\mathbf{C}^m\|_2\cdot\left|\sum_{k=1}^{\infty}\mathbf{Tr}\left(\mathbf{RC}^k\right)/k\right| \qquad (4.11) \\
&\leq \left(1-\frac{\ell}{u}\right)^m\sum_{k=1}^{\infty}\mathbf{Tr}\left(\mathbf{RC}^k\right)/k. \qquad (4.12)
\end{aligned}$$

To prove the inequality of eqn. (4.10), we used Von Neumann's trace inequality[1]. The inequality of eqn. (4.10) follows since $\mathbf{C}^{k-m}\mathbf{R}$ is SPSD[2]. To prove the inequality of eqn. (4.11), we used the fact that $\mathbf{Tr}\left(\mathbf{RC}^k\right)/k \geq 0$ for any $k \geq 1$. Finally, to prove eqn. (4.12), we used the fact that $\|\mathbf{C}\|_2 = \|\mathbf{I}_n - u^{-1}\mathbf{\Sigma}_p\|_2 \leq 1 - \ell/u$ since, by our assumptions, the smallest entry in Σ_p is at least ℓ. We also removed unnecessary ab-

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

[2]This can be proven using an argument similar to the one used to prove eqn. (4.8).

solute values since $\mathbf{Tr}\left(\mathbf{RC}^k\right)/k$ is non-negative for any positive integer k. Combining the bounds for Δ_1 and Δ_2 gives

$$\left|\widehat{\mathcal{H}}\left(\mathbf{R}\right) - \mathcal{H}\left(\mathbf{R}\right)\right| \leq \left(\varepsilon + \left(1 - \frac{\ell}{u}\right)^m\right) \sum_{k=1}^{\infty} \frac{\mathbf{Tr}\left(\mathbf{RC}^k\right)}{k}.$$

We have already proven in Lemma 13 that

$$\sum_{k=1}^{\infty} \frac{\mathbf{Tr}\left(\mathbf{RC}^k\right)}{k} \leq \mathcal{H}\left(\mathbf{R}\right) - \log u^{-1} \leq \mathcal{H}\left(\mathbf{R}\right),$$

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

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

Setting

$$m = \left\lceil \frac{u}{\ell} \log \frac{1}{\varepsilon} \right\rceil$$

and using Euler's identity as defined in eqn. (3.21) of Section 3.2.2 we guarantee that $(1 - \ell/u)^m \leq \varepsilon$. The failure probability of Algorithm 10 is at most 2δ; the sum of the failure probabilities of Algorithm 2 and Algrorithm 3. This concludes the proof. $\square$

A few remarks are necessary to better understand Theorem 4. First, ℓ could be set to p_n, the smallest of the probabilities corresponding to the n pure states of the density matrix $\mathbf{R}$. Second, it should be obvious that u in Algorithm 10 could be simply set to one and thus we could avoid calling Algorithm 2[3] to estimate p_1 by $\tilde{p}_1$ and thus compute u. However, if p_1 is small, then u could be significantly smaller than one, thus reducing the running time of Algorithm 10, which depends on the ratio u/ℓ (see, eqn. (4.13)). Third, ideally, if both p_1 and p_n were used instead of u and ℓ, respectively, the running time of the algorithm would scale with the ratio p_1/p_n, which is equal to the condition number of $\mathbf{R}$.

[3]Actually we will be using $u = 1$ for our Hermitian case Taylor-based approach.

4.2.3 Running time

Finally, we discuss the running time of Algorithm 10, which is equal to

$$\mathcal{O}\left(s \cdot m \cdot \operatorname{nnz}\left(\mathbf{R}\right)\right).$$

Since $s = \mathcal{O}\left(\frac{\log(1/\delta)}{\varepsilon^2}\right)$ and $m = \mathcal{O}\left(\frac{u \log(1/\varepsilon)}{\ell}\right)$, the running time becomes (after accounting for the running time of Algorithm 2)

$$\mathcal{O}\left(\left(\frac{u}{\ell} \cdot \frac{\log(1/\varepsilon)}{\varepsilon^2} + \log(n)\right) \log(1/\delta) \cdot \operatorname{nnz}\left(\mathbf{R}\right)\right). \tag{4.13}$$

4.3 An approach via Chebyschev polynomials

Our second approach is a Chebyschev polynomial-based approximation scheme to estimate the Von Neumann entropy of a density matrix. Our approach follows the work of [WBS14], but our analysis uses the trace estimators of [AT11] and Algorithm 2 and its analysis. Importantly, we present conditions under which the proposed approach is competitive with the approach of Section 4.2.

The proposed algorithm leverages the fact that the Von Neumann entropy of a density matrix $\mathbf{R}$ is equal to the (negative) trace of the matrix function $f\left(\mathbf{R}\right) = \mathbf{R}\log\left[\mathbf{R}\right]$ and approximates $f\left(\mathbf{R}\right)$ using the weighted sum of Chebyschev polynomials of the first kind (see, Section 2.3.2); then, the trace of the resulting matrix is estimated using Algorithm 3.

First we define the Chebyshev polynomial approximation for the scalar function $f(x) = x\log(x)$, with $x \in [0, u]$.

Definition 21. *The analytical function $f(x) = x\log(x)$ on $[0, u]$ can be approximated by the polynomial $f_m(x)$ of degree $m \geq 0$, defined as:*

$$f_m(x) = \sum_{w=0}^{m} \alpha_w \mathcal{T}_w(x), \tag{4.14}$$

where,

$$\mathcal{T}_w(x) = \cos\left(w \cdot \arccos\left(\frac{2}{u}x - 1\right)\right)$$

is the Chebyschev polynomial of the first kind of degree $w > 0$ and

$$\alpha_0 = \frac{u}{2}\left(\log\frac{u}{4} + 1\right), \quad \alpha_1 = \frac{u}{4}\left(2\log\frac{u}{4} + 3\right), \quad \alpha_w = \frac{(-1)^w u}{w^3 - w},$$

are the coefficients of $f_m(x)$.

Definition 22. *Let $\mathbf{R} \in \mathbb{R}^{n \times n}$ be a symmetric positive definite matrix with unit trace and whose eigenvalues lie in the interval $[\ell, u]$, for some $0 < \ell \leq u < 1$. Then,*

$$\mathcal{H}(\mathbf{R}) \approx -\mathbf{Tr}\left(f_m(\mathbf{R})\right) = -\mathbf{Tr}\left(\sum_{w=0}^{m} \alpha_w \mathcal{T}_w(\mathbf{R})\right) \tag{4.15}$$

4.3.1 Algorithm

Given a density matrix $\mathbf{R} \in \mathbb{R}^{n \times n}$ with eigenvalues $u \geq p_1 \geq \cdots \geq p_n$, Definition 22 reveals a high-level function for the approximation of $\mathcal{H}(\mathbf{R})$ that is sketched in Algorithm 11.

Algorithm 11 High Level Procedure for the Von Neumann Entropy Estimation

```
1: function VNEntropy($\mathbf{R}$)
2:     Compute $u$ such that $p_1(\mathbf{R}) \leq u$.
3:     return $\widehat{\mathcal{H}}(\mathbf{R}) = -\mathbf{Tr}(f_m(\mathbf{R}))$.
4: end function
```

To implement step 2 we use the power method from the numerical linear algebra literature (see, Section 2.5.1). To estimate the trace of $f_m(\mathbf{R})$, we use the randomized algorithm of Section 2.5.2. Our approach is depicted in detail in Algorithm 12.

We note that the computation in step 5 can be efficiently done using Clenshaw's algorithm (see, Algorithm 1).

Algorithm 12 A Chebyschev polynomial approach to estimate the Von Neumann entropy.

```
Input: $\mathbf{R} \in \mathbb{R}^{n\times n}$, accuracy parameter $\varepsilon > 0$, failure probability $\delta$, and integer $m > 0$.
Output: $\widehat{\mathcal{H}}(\mathbf{R})$, the approximation to $\mathcal{H}(\mathbf{R})$.
1: Compute $\tilde{p}_1$, the estimate of the largest eigenvalue of $\mathbf{R}$, $p_1$, using Algorithm 2 with $t = \mathcal{O}(\log(n))$ and $q = \mathcal{O}(\log(1/\delta)$.
2: $u = \min\{1, 6\tilde{p}_1\}$.
3: $s = \lceil 20\log(2/\delta)/\varepsilon^2 \rceil$.
4: Generate $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_s \in \mathbb{R}^n$ i.i.d. random Gaussian vectors.
5: return
       $$\widehat{\mathcal{H}}(\mathbf{R}) = -\frac{1}{s}\sum_{i=1}^{s} \mathbf{g}_i^\top f_m(\mathbf{R})\mathbf{g}_i.$$
```

4.3.2 Error bound

Theorem 5 is our main quality-of-approximation result for Algorithm 12. It essentially proves that Algorithm 12 returns an accurate approximation to $\mathcal{H}(\mathbf{R})$.

Theorem 5. *Let $\mathbf{R}$ be a density matrix such that all probabilities p_i, $i = 1 \ldots n$ satisfy $0 < \ell \leq p_i$. Let u be computed as in Algorithm 10 and let $\widehat{\mathcal{H}}(\mathbf{R})$ be the output of Algorithm 12 on inputs $\mathbf{R}$, m, and $\varepsilon < 1$; Then, with probability at least $1 - 2\delta$,*

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

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

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

Proof. We will condition our analysis on Algorithm 2 being successful, which happens with probability at least $1-\delta$. In this case, $u = \min\{1, 6\tilde{p}_1\}$ is an upper bound for all probabilities p_i. Recall that in the introduction of the chapter we defined the function $h(x) = x\log x$ for any real $x \in (0, 1]$, with $h(0) = 0$. Let $\mathbf{R} = \mathbf{\Psi}\mathbf{\Sigma}_p\mathbf{\Psi}^\top \in \mathbb{R}^{n\times n}$ be the density matrix, where both $\mathbf{\Sigma}_p$ and $\mathbf{\Psi}$ are both matrices in $\mathbb{R}^{n\times n}$. Notice that the diagonal entries of $\mathbf{\Sigma}_p$ are the p_i's and they satisfy $0 < \ell \leq p_i \leq u < 1$ for all $i = 1 \ldots n$.

Using the definitions of matrix functions (see, Section 2.2.6), we can now define $h(\mathbf{R}) = \mathbf{\Psi} h(\mathbf{\Sigma}_p)\mathbf{\Psi}^\top$, where $h(\mathbf{\Sigma}_p)$ is a diagonal matrix in $\mathbb{R}^{n\times n}$ with entries equal to $h(p_i)$ for all $i = 1 \ldots n$. We now restate Proposition 3.1 from [WBS14] in the context of our work, using our notation.

Lemma 14. *The function $h(x)$ in the interval $[0, u]$ can be approximated by*

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

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

$$\begin{aligned} |h(x) - f_m(x)| &\leq \frac{u}{2m(m+1)} \\ &\leq \frac{u}{2m^2}, \end{aligned} \tag{4.16}$$

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

In Lemma 14, $\mathcal{T}_w(x) = \cos(w \cdot \arccos((2/u)x - 1))$ for any integer $w \geq 0$ and $x \in [0, u]$. Notice that the function $(2/u)x - 1$ essentially maps the interval $[0, u]$, which is the interval of interest for the function $h(x)$, to $[-1, 1]$, which is the interval over which Chebyschev polynomials are commonly defined. Lemam 14 exploits the fact that the Chebyschev polynomials form an orthonormal basis for the space of functions over the interval $[-1, 1]$.

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

$$\begin{aligned}
-\mathbf{Tr}\left(f_m(\mathbf{R})\right) =& -\mathbf{Tr}\left(\sum_{w=0}^{m}\alpha_w\mathcal{T}_w(\mathbf{R})\right)\\
=& -\mathbf{Tr}\left(\sum_{w=0}^{m}\alpha_w\mathbf{\Psi}\mathcal{T}_w(\mathbf{\Sigma}_p)\mathbf{\Psi}^\top\right)\\
=& -\sum_{w=0}^{m}\alpha_w\mathbf{Tr}\left(\mathcal{T}_w(\mathbf{\Sigma}_p)\right)\\
=& -\sum_{w=0}^{m}\alpha_w\sum_{i=1}^{n}\mathcal{T}_w(p_i)\\
=& -\sum_{i=1}^{n}\sum_{w=0}^{m}\alpha_w\mathcal{T}_w(p_i).
\end{aligned} \tag{4.17}$$

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

$$\begin{aligned}
\left|\mathcal{H}(\mathbf{R}) - \mathbf{Tr}\left(-f_m(\mathbf{R})\right)\right| =& \left|-\sum_{i=1}^{n}h(p_i) + \sum_{i=1}^{n}\sum_{w=0}^{m}\alpha_w\mathcal{T}_w(p_i)\right|\\
\leq& \sum_{i=1}^{n}\left|h(p_i) - \sum_{w=0}^{m}\alpha_w\mathcal{T}_w(p_i)\right|\\
\leq& \frac{nu}{2m^2}.
\end{aligned} \tag{4.18}$$

The inequality of eqn. (4.18) follows by the bound of eqn. (4.16) in Lemma 14, since all p_i's are in the interval $[0, u]$.

Recall that we further assumed that all p_i's are lower-bounded by $\ell > 0$ and thus

$$\mathcal{H}(\mathbf{R}) = \sum_{i=1}^{n} p_i \log\frac{1}{p_i} \geq n\ell\log\frac{1}{1-\ell}. \tag{4.19}$$

We note that the upper bound on the p_i's follows since the smallest p_i is at least $\ell > 0$ and thus the largest p_i cannot exceed $1-\ell < 1$. We note that we cannot use the upper bound u in formula of eqn. (4.19), since u could be equal to one; $1-\ell$ is always strictly less than one but it cannot be a-priori computed (and thus cannot be used in Algorithm 12), since ℓ is not a-priori known.

We can now restate the bound of eqn. (4.18) as follows:

$$\begin{aligned}\left|\mathcal{H}\left(\mathbf{R}\right) - \mathbf{Tr}\left(-f_m(\mathbf{R})\right)\right| \leq & \frac{u}{2m^2\ell\log(1/(1-\ell))}\mathcal{H}\left(\mathbf{R}\right)\\ \leq & \varepsilon\mathcal{H}\left(\mathbf{R}\right), \end{aligned} \tag{4.20}$$

where the last inequality follows by setting

$$m = \sqrt{\frac{u}{2\cdot\varepsilon\cdot\ell\cdot\log(1/(1-\ell))}}. \tag{4.21}$$

Next, we argue that the matrix $-f_m(\mathbf{R})$ is symmetric positive semi-definite (under our assumptions) and thus one can apply Lemma 5 to estimate its trace. We note that

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

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

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

where the inequalities follow from Lemma 14 and our choice for m from eqn. (4.21). This inequality holds for all $i = 1\dots n$ and implies that

$$-f_m(p_i) \geq p_i\log\frac{1}{p_i} - \varepsilon\ell\log\frac{1}{1-\ell} \geq (1-\varepsilon)\ell\log\frac{1}{1-\ell},$$

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

We can now apply the trace estimator of Lemma 5 to get

$$\left|\mathbf{Tr}\left(-f_m(\mathbf{R})\right)-\left(-\frac{1}{s}\sum_{i=1}^{s}\mathbf{g}_i^\top f_m(\mathbf{R})\mathbf{g}_i\right)\right|\leq\varepsilon\cdot\mathbf{Tr}\left(-f_m(\mathbf{R})\right). \tag{4.22}$$

For the above bound to hold, we need to set

$$s=\left\lceil 20\log(2/\delta)/\varepsilon^2\right\rceil. \tag{4.23}$$

We now conclude as follows:

$$\left|\mathcal{H}\left(\mathbf{R}\right)-\widehat{\mathcal{H}}\left(\mathbf{R}\right)\right|\leq\left|\mathcal{H}\left(\mathbf{R}\right)-\mathbf{Tr}\left(-f_m(\mathbf{R})\right)\right|+\left|\mathbf{Tr}\left(-f_m(\mathbf{R})\right)-\left(-\frac{1}{s}\sum_{i=1}^{s}\mathbf{g}_i^\top f_m(\mathbf{R})\mathbf{g}_i\right)\right| \tag{4.24}$$

$$\leq\varepsilon\mathcal{H}\left(\mathbf{R}\right)+\varepsilon\mathbf{Tr}\left(-f_m(\mathbf{R})\right) \tag{4.25}$$

$$\leq\varepsilon\mathcal{H}\left(\mathbf{R}\right)+\varepsilon(1+\varepsilon)\mathcal{H}\left(\mathbf{R}\right) \tag{4.26}$$

$$\leq 3\varepsilon\mathcal{H}\left(\mathbf{R}\right). \tag{4.27}$$

The inequality of eqn. (4.24) follows by adding and subtracting $-\mathbf{Tr}\left(f_m(\mathbf{R})\right)$ and using sub-additivity of the absolute value; the inequality of eqn. (4.25) follows by eqn. (4.20) and eqn. (4.22); the inequality of eqn. (4.26) follows again by eqn. (4.20); and the inequality of eqn. (4.27) follows by using $\varepsilon\leq 1$.

The failure probability of Algorithm 12 is at most 2δ; the sum of the failure probabilities of Algorithm 2 and Algrorithm 3. This concludes the proof.

$\square$

4.3.3 Running time

Finally, we discuss the running time of Algorithm 12, which is equal to

$$\mathcal{O}\left(s \cdot m \cdot \mathrm{nnz}\left(\mathbf{R}\right)\right)$$

. Using the values for m and s from eqn.. (4.21) and eqn. (4.23), the running time becomes (after accounting for the running time of Algorithm 2)

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

4.3.4 Comparison between Theorems 4 and 5

The similarities between Theorems 4 and Theorem 5 are obvious: same assumptions and directly comparable accuracy guarantees. The only difference is in the running times: the Taylor series approach has a milder dependency on ε, while the Chebyschev-based approximation has a milder dependency on the ratio u/ℓ, which controls the behavior of the probabilities p_i. However, for small values of ℓ ($\ell \to 0$),

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

Thus, the Chebyschev-based approximation has a milder dependency on u but not necessarily ℓ when compared to the Taylor-series approach. We also note that the discussion following Theorem 4 is again applicable here.

4.3.5 A comparison with the results of [WBS14]

The work of [WBS14] culminates in the error bounds described in Theorem 5 (and the ensuing discussion). In our parlance, [WBS14] first derives the error bound of eqn. (4.18). It is worth emphasizing that the bound of eqn. (4.18) holds even if the

p_i's are not necessarily strictly positive, as assumed by Theorem 5: the bound holds even if some of the p_i's are equal to zero.

Unfortunately, without imposing a lower bound assumption on the p_i's it is difficult to get a meaningful error bound and an efficient algorithm. Indeed, the error implied by eqn. (4.18) (without any assumption on the p_i's) necessitates setting m to at least $\Omega(\sqrt{n})$ (perhaps up to a logarithmic factor, as we will discuss shortly). To understand this, note that the entropy of the density matrix $\mathbf{R}$ ranges between zero and $\log(k)$, where k is the rank of the matrix $\mathbf{R}$, i.e., the number of non-zero p_i's. Clearly, $k \leq n$ and thus $\log(n)$ is an upper bound for $\mathcal{H}(\mathbf{R})$. Notice that if $\mathcal{H}(\mathbf{R})$ is smaller than $n/(2m^2)$, the error bound of eqn. (4.18) does not even guarantee that the resulting approximation will be positive, which is, of course, meaningless as an approximation to the entropy.

In order to guarantee a relative error bound of the form $\varepsilon\mathcal{H}(\mathbf{R})$ via eqn. (4.18), we need to set m to be at least

$$m \geq \sqrt{\frac{n}{2\varepsilon\mathcal{H}(\mathbf{R})}}, \tag{4.28}$$

which even for "large" values of $\mathcal{H}(\mathbf{R})$ (i.e., values close to the upper bound $\log(n)$) still implies that m is $\mathcal{O}\left(\varepsilon^{-1/2}\sqrt{n/\log(n)}\right)$. Even with such a large value for m, we are still not done: we need an efficient trace estimation procedure for the matrix $-f_m(\mathbf{R})$. While this matrix is always symmetric, it is not necessarily positive or negative semi-definite (unless additional assumptions are imposed on the p_i's, like we did in Theorem 5).

4.4 Approaches via Taylor and Chebyshev expansions for Hermitian density matrices

Hermitian positive definite matrices, frequently arise in quantum mechanics. The analyses of Sections 4.2 and 4.3 focus on real density matrices; we now briefly discuss how they can be extended to apply on Hermitian density matrices. Recall that both approaches follow the same algorithmic scheme. First, the dominant eigenvalue of the density matrix is estimated via the power method; a trace estimation follows using Gaussian trace estimators on either the truncated Taylor expansion of a suitable matrix function or on a Chebyshev polynomial approximation of the same matrix function. Interestingly, the Taylor expansions, as well as the Chebyshev polynomial approximations, both work when the input matrix is complex. However, the estimation of the dominant eigenvalue of $\mathbf{R}$ poses a theoretical difficulty: to the best of our knowledge, there is no known bound for the accuracy of the power method in the case where $\mathbf{R}$ is complex. Lemma 4 guarantees relative error approximations to the dominant eigenvalue of real matrices, but we are not aware of any provable relative error bound for the complex case. To avoid this issue we will be using one as a (loose) upper bound for the dominant eigenvalue.

The crucial step in order to guarantee relative error approximations to the Von Neumann entropy of a Hermitian positive definite matrix is to guarantee relative error approximations for the trace of a Hermitian positive definite matrix. Lemma 5 assumes symmetric positive semi-definite matrices; we now prove that the same lemma can be applied on Hermitian positive definite matrices to achieve the same guarantees.

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

$$\mathbf{A} = \mathbf{B} + i\mathbf{C}, \tag{4.29}$$

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

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

Theorem 7. *The trace of a Hermitian matrix $\mathbf{A} \in \mathbb{C}^{n \times n}$ expressed as in eqn.* (4.29) *is equal to the trace of its real part:*

$$\mathbf{Tr}\left(\mathbf{A}\right) = \mathbf{Tr}\left(\mathbf{B}\right).$$

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

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

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

In words, Theorem 7 states that the trace of a Hermitian matrix equals the trace of its real part. Similarly, Theorem 6 states that the real part of a Hermitian positive semi-definite matrix is symmetric positive semi-definite. Combining both theorems we conclude that we can estimate the trace of a Hermitian positive definite matrix up to relative error, using the Gaussian trace estimator of Lemma 5 on its real part. Therefore, both approaches generalize to Hermitian positive definite matrices using one as an upper bound instead of u for the dominant eigenvalue. Algorithms 13 and 14 are modified versions of Algorithms 10 and 12 respectively that work on Hermitian inputs (the function $\mathrm{Re}(\cdot)$ returns the real part of its argument in an entry-wise manner).

Algorithm 13 A Taylor series approach to estimate the Von Neumann entropy.

```
Input: $\mathbf{R} \in \mathbb{C}^{n\times n}$, accuracy parameter $\varepsilon > 0$, failure probability $\delta$, and integer $m > 0$.
Output: $\widehat{\mathcal{H}}(\mathbf{R})$, the approximation to $\mathcal{H}(\mathbf{R})$.
1: $s = \lceil 20\log(2/\delta)/\varepsilon^2 \rceil$.
2: Generate $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_s \in \mathbb{R}^n$ i.i.d. random Gaussian vectors.
3: return
   $$\widehat{\mathcal{H}}(\mathbf{R}) = \frac{1}{s}\sum_{i=1}^{s}\sum_{k=1}^{m}\frac{\mathbf{g}_i^\top\left(\mathrm{Re}\left[\mathbf{R}(\mathbf{I}_n - \mathbf{R})^k\right]\right)\mathbf{g}_i}{k}.$$
```

Theorems 8 and 9 are our main quality-of-approximation results for Algorithm 13 and 14.

Algorithm 14 A Chebyschev polynomial approach to estimate the Von Neumann entropy.

```
Input: $\mathbf{R} \in \mathbb{C}^{n\times n}$, accuracy parameter $\varepsilon > 0$, failure probability $\delta$, and integer
    $m > 0$.
Output: $\widehat{\mathcal{H}}(\mathbf{R})$, the approximation to $\mathcal{H}(\mathbf{R})$.
1: $s = \lceil 20\log(2/\delta)/\varepsilon^2 \rceil$.
2: Generate $\mathbf{g}_1, \mathbf{g}_2, \dots, \mathbf{g}_s \in \mathbb{R}^n$ i.i.d. random Gaussian vectors.
3: return
    $$\widehat{\mathcal{H}}(\mathbf{R}) = -\frac{1}{s}\sum_{i=1}^{s} \mathbf{g}_i^\top \left(\mathrm{Re}\left[f_m(\mathbf{R})\right]\right)\mathbf{g}_i.$$
```

Theorem 8. *Let* $\mathbf{R}$ *be a complex density matrix such that all probabilities* p_i*,* $i = 1\dots n$ *satisfy* $0 < \ell \le p_i$*. Let* $\widehat{\mathcal{H}}(\mathbf{R})$ *be the output of Algorithm 13 on inputs* $\mathbf{R}$*,* m*, and* $\varepsilon < 1$*. Then, with probability at least* $1-\delta$*,*

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

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

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

Theorem 9. *Let* $\mathbf{R}$ *be a density matrix such that all probabilities* p_i*,* $i = 1\dots n$ *satisfy* $0 < \ell \le p_i$*. Let* $\widehat{\mathcal{H}}(\mathbf{R})$ *be the output of Algorithm 14 on inputs* $\mathbf{R}$*,* m*, and* $\varepsilon < 1$*. Then, with probability at least* $1-\delta$*,*

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

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

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

Algorithm 15 Approximating the entropy via random projection matrices

```
Input: Integer $n$ (dimensions of matrix $\mathbf{R}$) and integer $k$ (with rank of $\mathbf{R}$ at most $k \ll n$, see eqn. (4.1)).
Output: $\tilde{p}_i$, $i = 1 \ldots k$, the approximation to the top-$k$ eigenvalues of $\mathbf{R}$ and $\widehat{\mathcal{H}}(\mathbf{R})$, the approximation to $\mathcal{H}(\mathbf{R})$.
1: Construct the random projection matrix $\mathbf{\Pi} \in \mathbb{R}^{n \times s}$ ▷ see Section 2.5.3 for details on $\mathbf{\Pi}$ and $s$.
2: $\tilde{\mathbf{R}} = \mathbf{R}\mathbf{\Pi} \in \mathbb{R}^{n \times s}$.
3: Compute the (at most) $k$ non-zero singular values of $\tilde{\mathbf{R}}$, $\tilde{p}_i$, $i = 1 \ldots k$.
4: return $\tilde{p}_i$, $i = 1 \ldots k$ and
       $$\widehat{\mathcal{H}}(\mathbf{R}) = \sum_{i=1}^{k} \tilde{p}_i \log \frac{1}{\tilde{p}_i}$$
   .
```

4.5 An approach via random projection matrices

Finally, we focus on perhaps the most interesting special case: the setting where at most k (out of n, with $k \ll n$) of the probabilities p_i of the density matrix $\mathbf{R}$ of eqn. (4.1) are non-zero. In this setting, we prove that elegant random-projection-based techniques achieve relative error approximations to all probabilities p_i, $i = 1 \ldots k$. The running time of the proposed approach depends on the particular random projection that is used and can be made to depend on the sparsity of the input matrix.

4.5.1 Algorithm

The proposed algorithm uses a random projection matrix $\mathbf{\Pi}$ to create a "sketch" of $\mathbf{R}$ in order to approximate the p_i's. In words, Algorithm 15 creates a sketch of the input matrix $\mathbf{R}$ by post-multiplying $\mathbf{R}$ by a random projection matrix; this is a well-known approach from the RandNLA literature (see, [DM16] for details). Assuming that $\mathbf{R}$ has rank at most k, which is equivalent to assuming that at most k of the probabilities p_i in eqn. (4.1) are non-zero (e.g., the system underlying the density matrix $\mathbf{R}$ has at most k pure states), then the rank of $\mathbf{R}\mathbf{\Pi}$ is also at most k. In this setting, Algorithm 15 returns the non-zero singular values of $\mathbf{R\Pi}$ as approximations to the p_i's, $i = 1 \ldots k$.

4.5.2 Error bound

Theorem 10 is our main quality-of-approximation result for Algorithm 15.

Theorem 10. *Let* $\mathbf{R}$ *be a density matrix with at most* $k \ll n$ *non-zero probabilities and let* $\varepsilon < 1/2$ *be an accuracy parameter. Then, with probability at least* 0.9*, the output of Algorithm 15 satisfies*

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

for all $i = 1 \ldots k$*. Additionally,*

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

Algorithm 15 (combined with Algorithm 5 below) runs in time

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

Proof. At the heart of the proof of Theorem 10 lies the following perturbation bound from [DV92, Theorem 2.3].

Theorem 11. *Let* $\mathbf{DAD}$ *be a symmetric positive definite matrix such that* $\mathbf{D}$ *is a diagonal matrix and* $\mathbf{A}_{ii} = 1$ *for all* i*. Let* $\mathbf{DED}$ *be a perturbation matrix such that* $\|\mathbf{E}\|_2 < \lambda_{\min}(\mathbf{A})$*. Let* λ_i *be the* i*-the eigenvalue of* $\mathbf{DAD}$ *and let* λ_i' *be the* i*-th eigenvalue of* $\mathbf{D}(\mathbf{A}+\mathbf{E})\mathbf{D}$*. Then, for all* i*,*

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

We note that $\lambda_{\min}(\mathbf{A})$ in the above theorem is a real, strictly positive number[4]. Now consider the matrix $\mathbf{R}\mathbf{\Pi}\mathbf{\Pi}^\top\mathbf{R}^\top$; we will use the above theorem to argue that its singular values are good approximations to the singular values of the matrix $\mathbf{R}\mathbf{R}^\top$. Recall that $\mathbf{R} = \mathbf{\Psi}\mathbf{\Sigma}_p\mathbf{\Psi}^\top$ where $\mathbf{\Psi}$ has orthonormal columns. Note that the eigenvalues of $\mathbf{R}\mathbf{R}^\top = \mathbf{\Psi}\mathbf{\Sigma}_p^2\mathbf{\Psi}^\top$ are equal to the eigenvalues of the matrix $\mathbf{\Sigma}_p^2$; similarly, the eigenvalues of $\mathbf{\Psi}\mathbf{\Sigma}_p\mathbf{\Psi}^\top\mathbf{\Pi}\mathbf{\Pi}^\top\mathbf{\Psi}\mathbf{\Sigma}_p\mathbf{\Psi}^\top$ are equal to the eigenvalues of $\mathbf{\Sigma}_p\mathbf{\Psi}^\top\mathbf{\Pi}\mathbf{\Pi}^\top\mathbf{\Psi}\mathbf{\Sigma}_p$. Thus, we can compare the matrices

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

In the parlance of Theorem 11, $\mathbf{E} = \mathbf{\Psi}^\top\mathbf{\Pi}\mathbf{\Pi}^\top\mathbf{\Psi} - \mathbf{I}_k$. Applying either Lemma 7 (after rescaling the matrix $\mathbf{\Pi}$) or Lemma 8, we immediately get that $\|\mathbf{E}_A\|_2 \leq \varepsilon < 1$ with probability at least 0.9. Since $\lambda_{\min}(\mathbf{I}_k) = 1$, the assumption of Theorem 11 is satisfied. We note that the eigenvalues of $\mathbf{\Sigma}_p\mathbf{I}_k\mathbf{\Sigma}_p$ are equal to p_i^2 for $i = 1 \ldots k$ (all positive, which guarantees that the matrix $\mathbf{\Sigma}_p\mathbf{I}_k\mathbf{\Sigma}_p$ is symmetric positive definite, as mandated by Theorem 11) and the eigenvalues of $\mathbf{\Sigma}_p\mathbf{\Psi}^\top\mathbf{\Pi}\mathbf{\Pi}^\top\mathbf{\Psi}\mathbf{\Sigma}_p$ are equal to $\tilde{p}_i^2$, where $\tilde{p}_i$ are the singular values of $\mathbf{\Sigma}_p\mathbf{\Psi}^\top\mathbf{\Pi}$. (Note that these are exactly equal to the outputs returned by Algorithm 15, since the singular values of $\mathbf{\Sigma}_p\mathbf{\Psi}^\top\mathbf{\Pi}$ are equal to the singular values of $\mathbf{\Psi}\mathbf{\Sigma}_p\mathbf{\Psi}^\top\mathbf{\Pi} = \mathbf{R}\mathbf{\Pi}$). Thus, we can conclude that:

$$\left|p_i^2 - \tilde{p}_i^2\right| \leq \varepsilon p_i^2. \tag{4.30}$$

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

The above result guarantees that all p_i's can be approximated up to *relative error* using Algorithm 15. We now investigate the implication of the above bound to approximating the Von Neumann entropy of $\mathbf{R}$. Indeed,

$$
\begin{aligned}
\sum_{i=1}^{k} \tilde{p}_i \log \frac{1}{\tilde{p}_i} &\leq \sum_{i=1}^{k} (1+\varepsilon)^{1/2} p_i \log \frac{1}{(1-\varepsilon)^{1/2} p_i} \\
&\leq (1+\varepsilon)^{1/2} \left(\sum_{i=1}^{k} p_i \log \frac{1}{p_i} + \sum_{i=1}^{k} p_i \log \frac{1}{(1-\varepsilon)^{1/2}} \right) \\
&= (1+\varepsilon)^{1/2} \mathcal{H}(\mathbf{R}) + \frac{\sqrt{1+\varepsilon}}{2} \log \frac{1}{1-\varepsilon} \\
&\leq (1+\varepsilon)^{1/2} \mathcal{H}(\mathbf{R}) + \frac{\sqrt{1+\varepsilon}}{2} \log(1+2\varepsilon) \qquad (4.31) \\
&\leq (1+\sqrt{\varepsilon}) \mathcal{H}(\mathbf{R}) + \sqrt{\frac{3}{2}} \varepsilon. \qquad (4.32)
\end{aligned}
$$

In the inequality of eqn. (4.31) we used $1/(1-\varepsilon) \leq 1+2\varepsilon$ for any $\varepsilon \leq 1/2$ and in the inequality of eqn. (4.32) we used $\log(1+2\varepsilon) \leq 2\varepsilon$ for $\varepsilon \in (0, 1/2)$. Similarly, we can prove that:

$$
\sum_{i=1}^{k} \tilde{p}_i \log \frac{1}{\tilde{p}_i} \geq (1-\sqrt{\varepsilon}) \mathcal{H}(\mathbf{R}) - \frac{1}{2} \varepsilon. \qquad (4.33)
$$

Combining the bounds (4.32) and (4.33) we get

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

This conclude the proof. □

4.5.3 Running time

We conclude by discussing the running time of Algorithm 15, which is dominated by the low rank approximation of step 3. Given the running time of $\mathcal{O}\left(\mathrm{nnz}\left(\mathrm{A}\right) + \mathrm{nk}^2\right)$

(see, [NN13, Figure 1]) for step 3 combined with Algorithm 5 for the construction of the matrix $\mathbf{\Pi}$we get a total running time of:

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

4.5.4 Comparison between Theorems 10, 4 and 5

Comparing the above result with Theorems 4 and 5, we note that Theorem 10 does not necessitate imposing any constraints on the probabilities p_i, $i = 1 \ldots k$. Instead, it suffices to have k non-zero probabilities. The final result is an additive-relative error approximation to the entropy of $\mathbf{R}$ (as opposed to the relative error approximations of Theorems 4 and 5); under the mild assumption $\mathcal{H}\left(\mathbf{R}\right) \geq \sqrt{\varepsilon}$, the above bound becomes a true relative error approximation[5].

4.5.5 The Hermitian case

The above approach via random projections critically depends on Lemmas 7 and 8, which, to the best of our knowledge, have only been proven for the real case. These results are typically proven using matrix concentration inequalities, which are well-explored for sums of random real matrices but less explored for sums of random complex matrices.

4.6 Empirical evaluation

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

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

We generated random density matrices for most of which we used the QETLAB Matlab toolbox [Joh16] to derive (real-valued) density matrices of size $5,000 \times 5,000$, on which most of our extensive evaluations were run. We also tested our methods on a much larger $30,000 \times 30,000$ density matrix, which was close to the largest matrix that Matlab would allow us to load. We used the function `RandomDensityMatrix` of QETLAB and the Haar measure; we also experimented with the Bures measure to generate random matrices, but we did not observe any qualitative differences worth reporting. Recall that exactly computing the Von Neumann entropy using eqn. (1.8) presumes knowledge of the entire spectrum of the matrix; to compute all singular values of a matrix we used the `svd` function of Matlab. The accuracy of our proposed approximation algorithms was evaluated by measuring the relative error; wall-clock times were reported in order to quantify the speedup that our approximation algorithms were able to achieve.

4.6.1 Empirical results for the Taylor and Chebyshev approximation algorithms

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

Consider a $5,000 \times 5,000$ random density matrix and let m (the number of terms retained in the Taylor series approximation or the degree of the polynomial used in the Chebyshev polynomial approximation) range between five and 30 in increments of five. Let s, the number of random Gaussian vectors used to estimate the trace, be set to $\{50, 100, 200, 300\}$. Recall that our error bounds for Algorithms 10 and 12 depend on u, an estimate for the largest eigenvalue of the density matrix. We used the power method to estimate the largest eigenvalue (let $\tilde{\lambda}_{\max}$ be the estimate) and we set u to $\tilde{\lambda}_{\max}$ and $6\tilde{\lambda}_{\max}$. Figures 4.1 and 4.2 show the relative error (out of 100%) for all combinations of m, s, and u for the Taylor and Chebyshev approximation algorithms. It is worth noting that we also report the error when no trace estimation (NTE) is used in order to highlight that most of the accuracy loss is due to the Taylor/Chebyshev approximation and not the trace estimation.

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

The second dataset that we experimented with was a much larger density matrix of size $30,000 \times 30,000$. This matrix was the largest matrix for which the memory was sufficient to perform operations like the full SVD. Notice that since the increase in the matrix size is six-fold compared to the previous one and SVD's running time grows cubically with the input size, we expect the running time to compute the exact SVD to be roughly $6^3 \cdot 90$ seconds, which is approximately 5.4 hours; indeed, the exact computation of the Von Neumann entropy took approximately 5.6 hours. We evaluated both the Taylor and the Chebyshev approximation schemes by setting the parameters m and s to take values in the sets $\{5, 10, 15, 20\}$ and $\{50, 100, 200\}$, respectively. The parameter u was set to $\tilde{\lambda}_{\max}$, where the latter value was computed using the power method, which took approximately 3.6 minutes. We report the wall-clock running times and relative error (out of 100%) in Figures 4.5 and 4.4.

We observe that the relative error is always less than 1% for both methods, with the Chebyshev approximation yielding almost always slightly better results. Note that our Chebyshev-polynomial-based approximation algorithm significantly outperformed

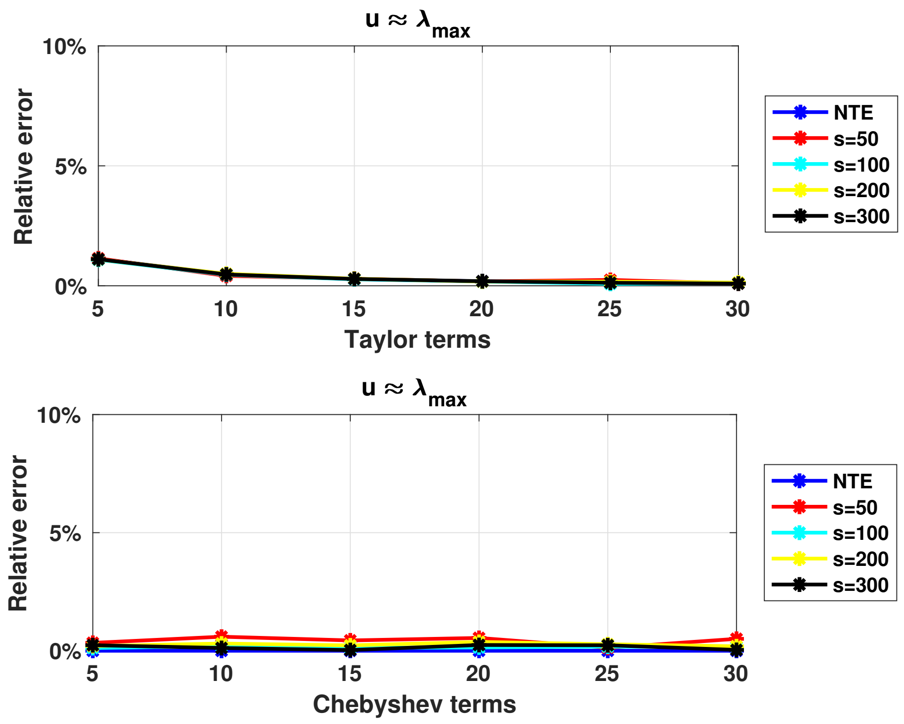


Figure 4.1.: Relative error for $5,000 \times 5,000$ density matrix using the Taylor and the Chebyshev approximation algorithms with $u = \tilde{\lambda}_{\max}$.

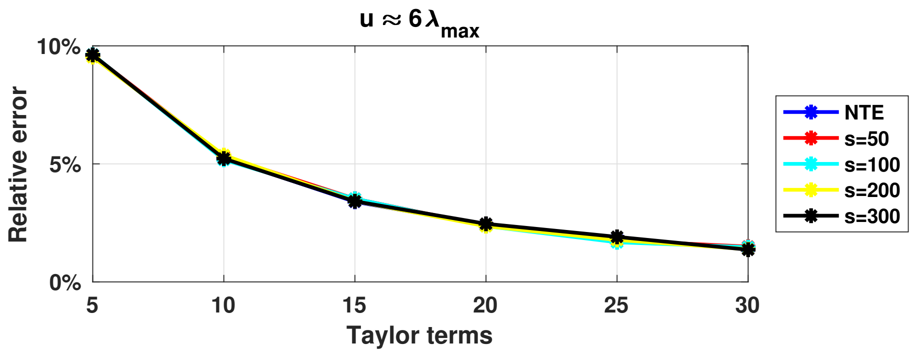
the exact computation: e.g., for $m = 5$ and $s = 50$, our estimate was computed in less than ten minutes and achieved less than .2% relative error.

The third dataset we experimented with was the tridiagonal matrix from [HMS15, Section 5.1]:

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 & \dots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \dots & 0 & -1 & 2 \end{bmatrix} \tag{4.34}$$

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

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



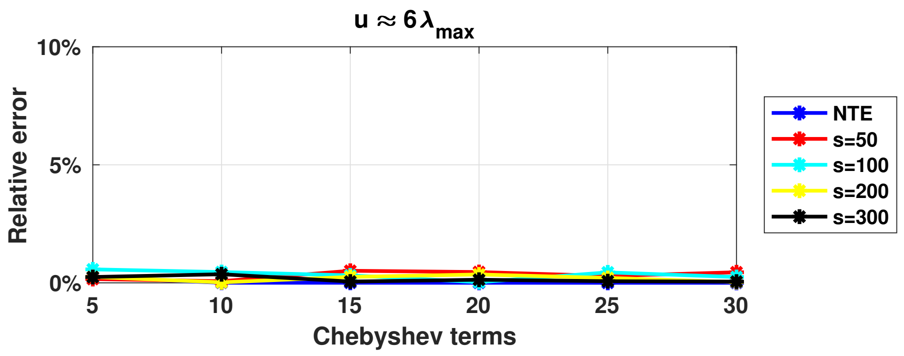


Figure 4.2.: Relative error for $5,000 \times 5,000$ density matrix using the Taylor and the Chebyshev approximation algorithms with $u = 6\tilde{\lambda}_{\max}$.

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

$$\lambda_i = \frac{4}{2n}\sin^2\left(\frac{i\pi}{2n+2}\right), \quad i = 1, \ldots, n \tag{4.35}$$

to compute the eigenvalues of $\mathbf{A}$ (after normalization) and we set u to $\lambda_{\max}$ and $6\lambda_{\max}$. Figures 4.6 and 4.7 show the relative error (out of 100%) for all combinations of m,

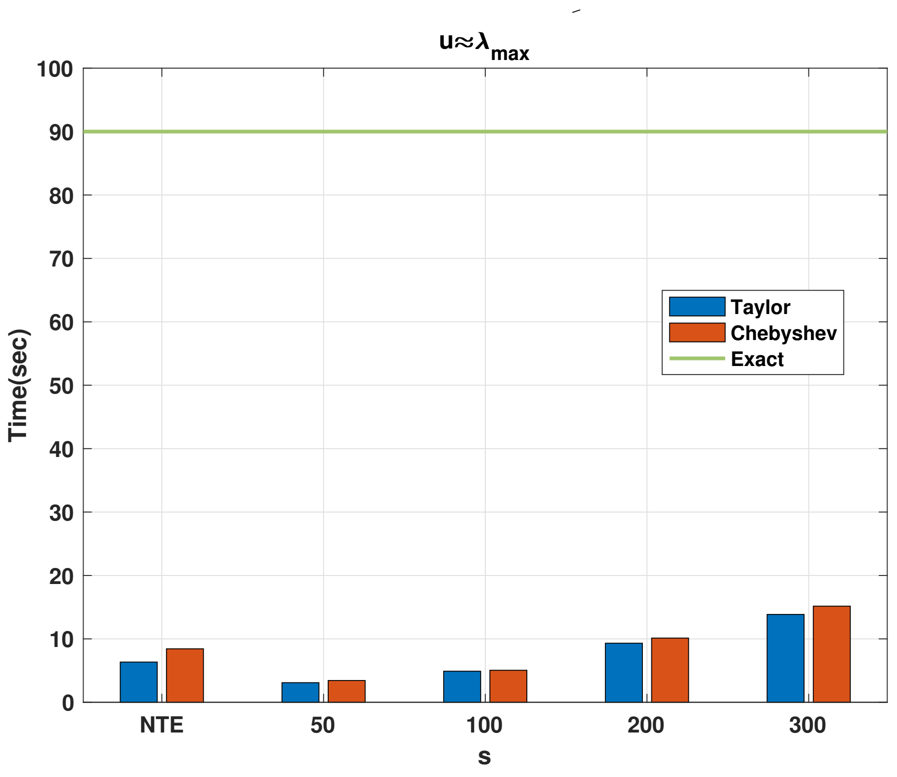


Figure 4.3.: Time (in seconds) to run the approximate algorithms for the $5,000 \times 5,000$ density matrix for $m = 5$. *Exactly* computing the Von Neumann entropy took approximately 90 seconds, designated by the straight horizontal line in the figure.

s, and u for the Taylor and Chebyshev approximation algorithms. We also report the error when no trace estimation (NTE) is used.

We observe that the relative error is higher than the one observed for the $5,000 \times 5,000$ random density matrix. We report wall-clock running times in Figure 4.8. The Chebyshev-polynomial-based algorithm returns better approximations for all choices of the parameters and, in most cases, is faster than the Taylor-polynomial-based algorithm, e.g. for $m = 5$, $s = 50$ and $u = \lambda_{\max}$, our estimate was computed in about two seconds and achieved less than .5% relative error.

We further considered a $10^8 \times 10^8$ tridiagonal matrix of the form of eqn. (4.34). Although an exact computation of the singular values of $\mathbf{A}$ is not feasible (at least with our computational resources), such a computation is not necessary since eqn. (4.35)

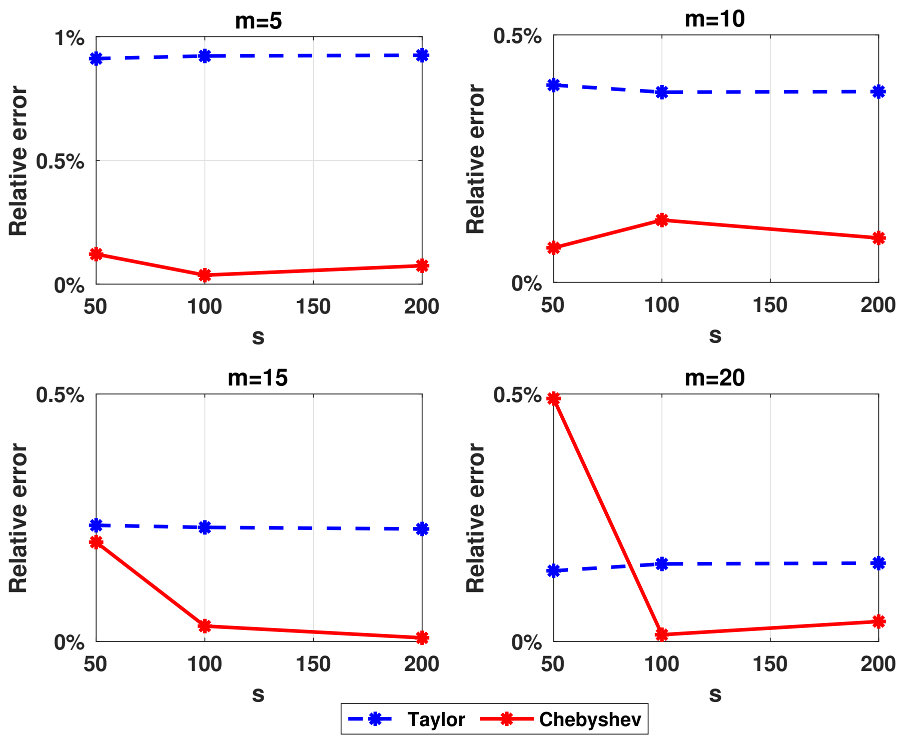


Figure 4.4.: Relative error for $30,000 \times 30,000$ density matrix using the Taylor and the Chebyshev approximation algorithms with $u = \tilde{\lambda}_{\max}$.

provides a closed formula for its eigenvalues and, thus, its entropy. Let m (the number of terms retained in the Taylor series approximation or the degree of the polynomial used in the Chebyshev polynomial approximation) be equal to five or ten and let s, the number of random Gaussian vectors used to estimate the trace be equal to 50 or 100. Figures 4.9 and 4.10 show the relative error (out of 100%) and the runtime, respectively, for all combinations of m and s for both the Taylor and Chebyshev approximation algorithms. We observe that in both cases we estimated the entropy in less than ten minutes with a relative error below 0.15%.

The fourth dataset we experimented with includes $5,000 \times 5,000$ density matrices whose first top-k eigenvalues follow a linear decay and the remaining $5,000 - k$ a uniform distribution. Let k, the number of eigenvalues that follow the linear decay, take values in the set $\{50,\ 1000,\ 3500,\ 5000\}$. Let m, the number of terms

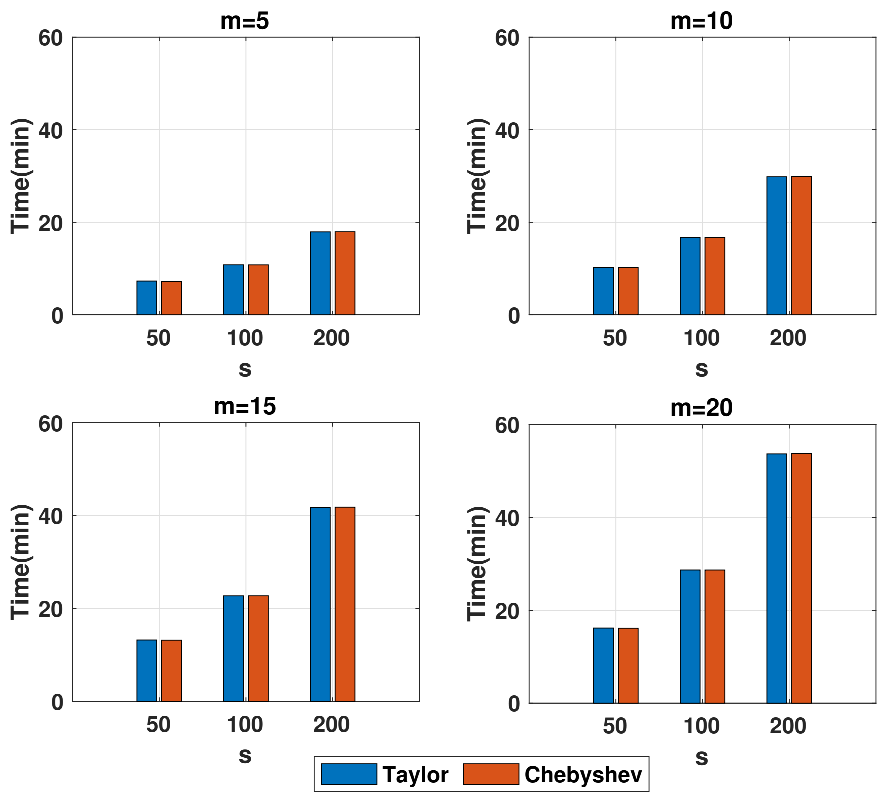


Figure 4.5.: Wall-clock times: Taylor approximation (blue) and Chebyshev approximation (red) for $u = \tilde{\lambda}_{\max}$. Exact computation needed approximately 5.6 hours.

retained in the Taylor series approximation or the degree of the polynomial used in the Chebyshev polynomial approximation, range between five and 30 in increments of five. Let s, the number of random Gaussian vectors used to estimate the trace, be set to $\{50, 100, 200, 300\}$. The estimate of the largest eigenvalue u is set to $\tilde{\lambda}_{\max}$. Figures 4.11 to 4.14 show the relative error (out of 100%) for all combinations of k, m, s, and u for the Taylor and Chebyshev approximation algorithms.

We observe that the relative error is decreasing as k increases. It is worth noting that when $k = 3,500$ and $k = 5,000$ the Taylor-polynomial-based algorithm returns better relative error approximation than the Chebyshev-polynomial-based algorithm. In the latter case we observe that the relative error of the Taylor-based algorithm is almost zero. This observation has a simple explanation. Figure 4.15 shows the

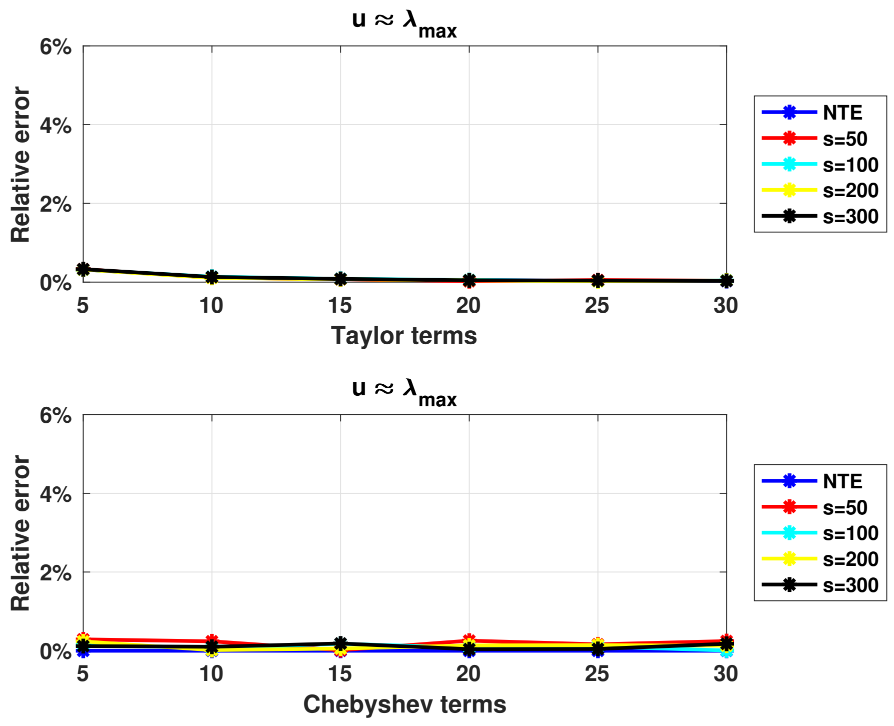


Figure 4.6.: Relative error for $5,000 \times 5,000$ tridiagonal density matrix using the Taylor and the Chebyshev approximation algorithms with $u = \lambda_{\text{max}}$.

distribution of the eigenvalues in the four cases we examine. We observe that for $k = 50$ the eigenvalues are spread in the interval $(10^{-2}, 10^{-4})$; for $k = 1,000$ the eigenvalues are spread in the interval $(10^{-3}, 10^{-4})$; while for $k = 3,500$ or $k = 5,000$ the eigenvalues are of order 10^{-4}. It is well known that the Taylor polynomial returns highly accurate approximations when it is computed on values lying inside the open disc centered at a specific value u, which, in our case, is the approximation to the dominant eigenvalue. The radius of the disk is roughly $r = \lambda_{m+1}/\lambda_m$, where m is the degree of the Taylor polynomial. If $r \leq 1$ then the Taylor polynomial converges; otherwise it diverges. Figure 4.16 shows the convergence rate for various values of k. We observe that for $k = 50$ the polynomial diverges, which leads to increased errors for the Taylor-based approximation algorithm (reported error close to 23%). In all

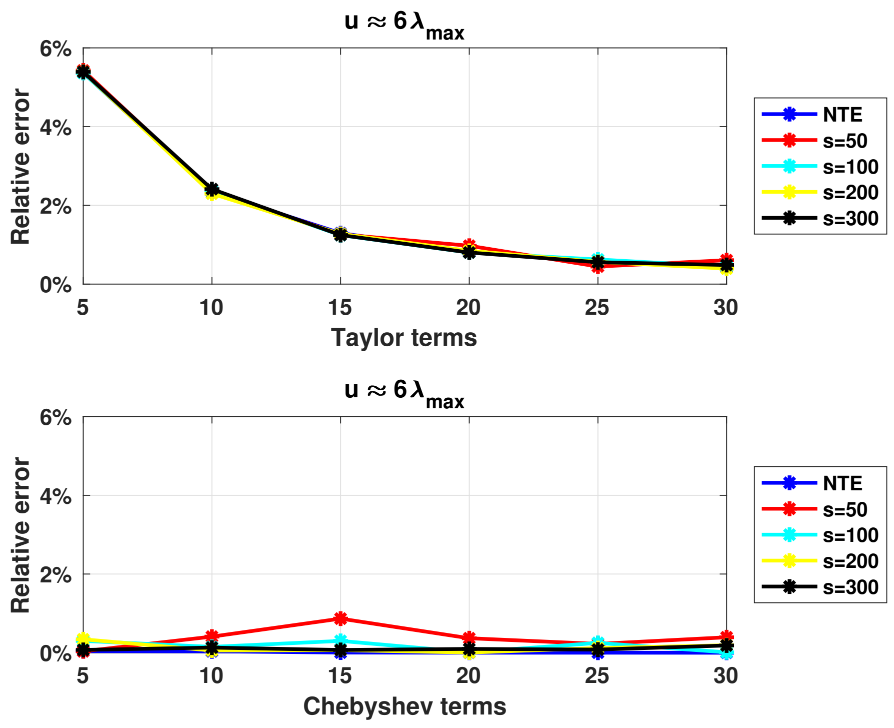


Figure 4.7.: Relative error for $5,000 \times 5,000$ tridiagonal density matrix using the Taylor and the Chebyshev approximation algorithms with $u = 6\lambda_{\max}$.

other cases, the convergence rate is close to one, resulting in negligible impact to the overall error.

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

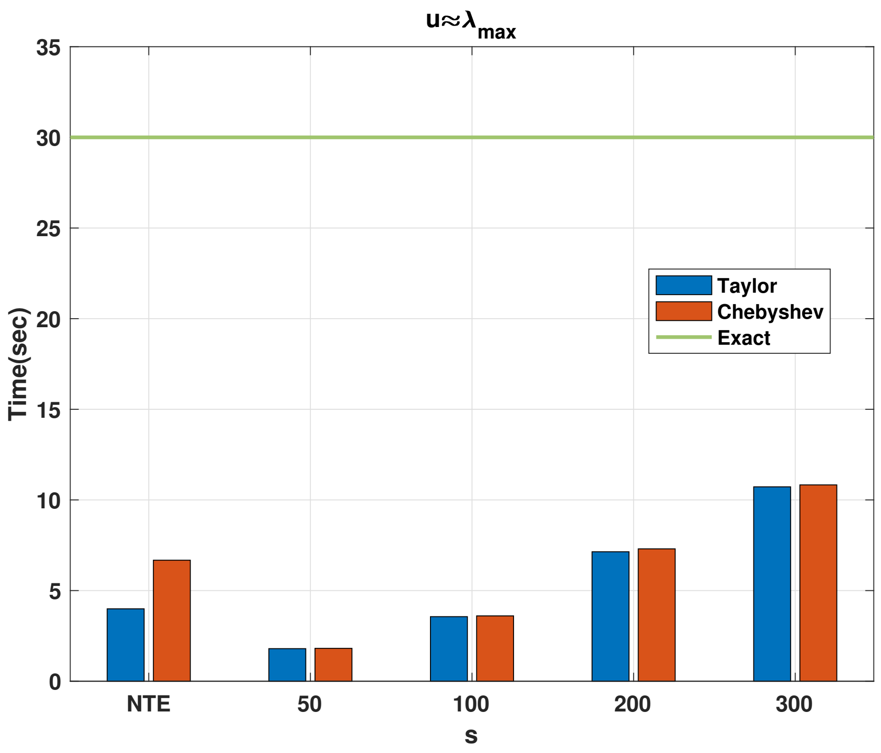


Figure 4.8.: Wall-clock times: Taylor approximation (blue) and Chebyshev approximation (red) for $m = 5$. Exact computation needed approximately 30 seconds, designated by the straight horizontal line in the figure.

the condition number is in the order of hundreds which is significant larger than the roughly constant condition number when $k = 5,000$.

4.6.2 Empirical results for the Hermitian case

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

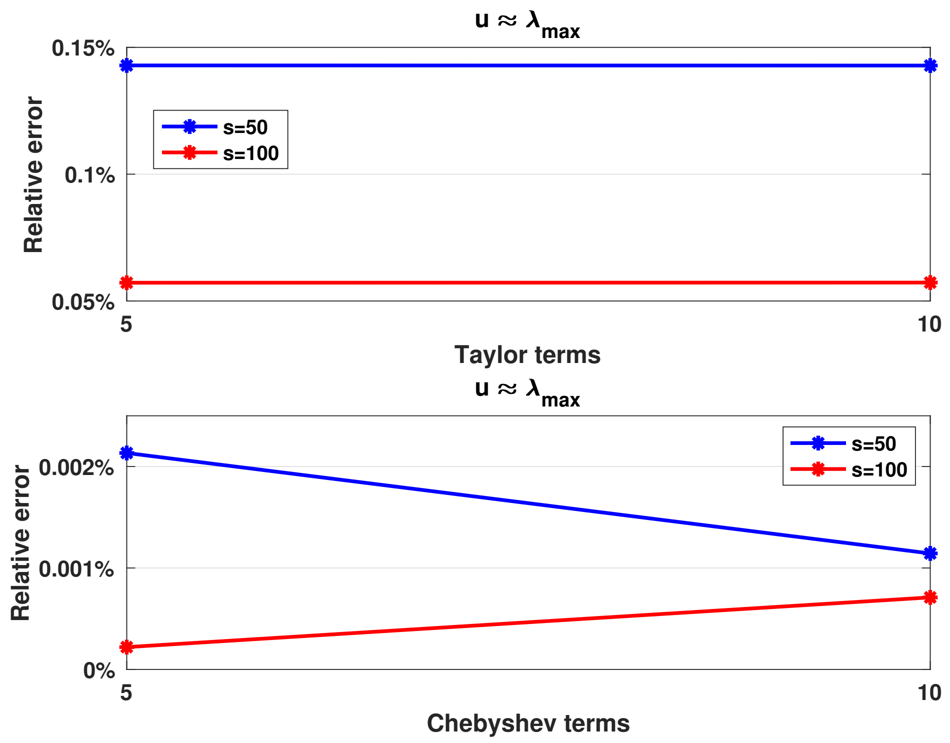


Figure 4.9.: Relative error for the $10^8 \times 10^8$ tridiagonal density matrix using the Taylor and the Chebyshev approximation algorithms with $u = \lambda_{\max}$.

Figure 4.17 shows the relative error (out of 100%) for all combinations of m, s, and u for the Taylor-based and Chebyshev-based approximation algorithms.

We observe that the relative error is always small, typically below 1%, for any choice of the parameters s and m. The NTE line (no trace estimation) in the plots serves as a lower bound for the relative error. We note that computing the exact Von Neumann entropy took approximately 52 seconds for matrices of this size. Finally, our algorithm seems to outperform exact computation of the Von Neumann entropy by approximating it in about ten seconds (for the Taylor-based approach) with a relative error of 0.5% using 100 random Gaussian vectors and retaining ten Taylor terms (see, Fig. 4.18) or in about 18 seconds (for the Chebyshev-based approach) with a relative error of 0.2% using 50 random Gaussian vectors and five Chebyshev polynomials (see, Fig.4.19) .

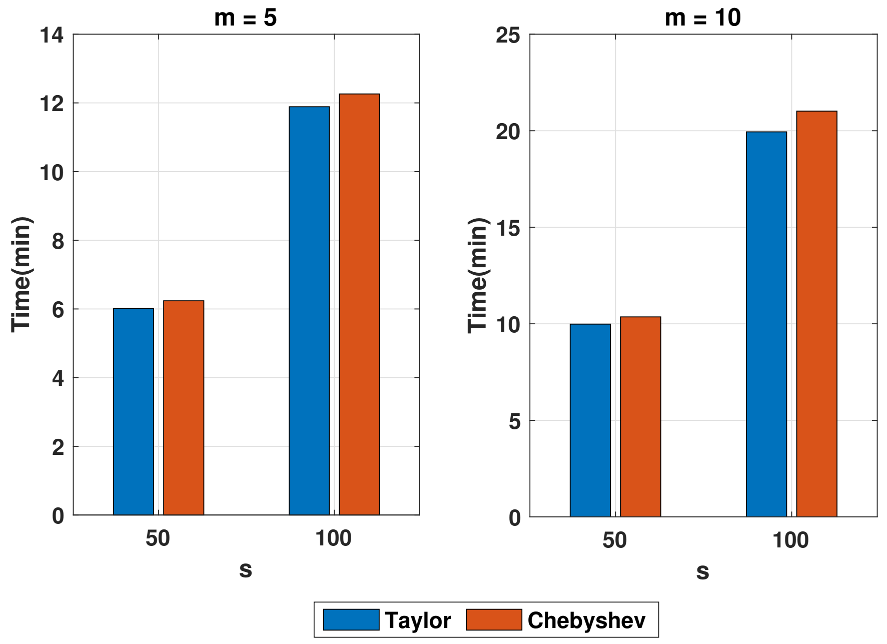
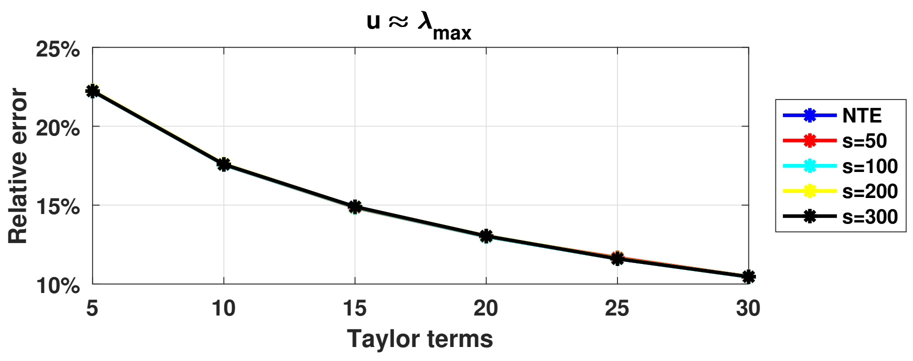


Figure 4.10.: Wall-clock times: Taylor approximation (blue) and Chebyshev approximation (red) for the $10^8 \times 10^8$ triadiagonal density matrix. Exact computation using the Singular Value Decomposition was infeasible using our computational resources.

4.6.3 Empirical results for the random projection approximation algorithms

In order to evaluate our third algorithm, we generated low-rank random density matrices (recall that the algorithm of Section 4.5 works only for random density matrices of rank k with $k \ll n$). Additionally, in order to evaluate the subsampled randomized Hadamard transform and avoid padding with all-zero rows, we focused on values of n (the number of rows and columns of the density matrix) that are powers of two. Finally, we also evaluated a simpler random projection matrix, namely the Gaussian random matrix, whose entries are all Gaussian random variables with zero mean and unit variance.

We generated *low rank* random density matrices with exponentially (using the QETLAB Matlab toolbox) and linearly decaying eigenvalues. The sizes of the density



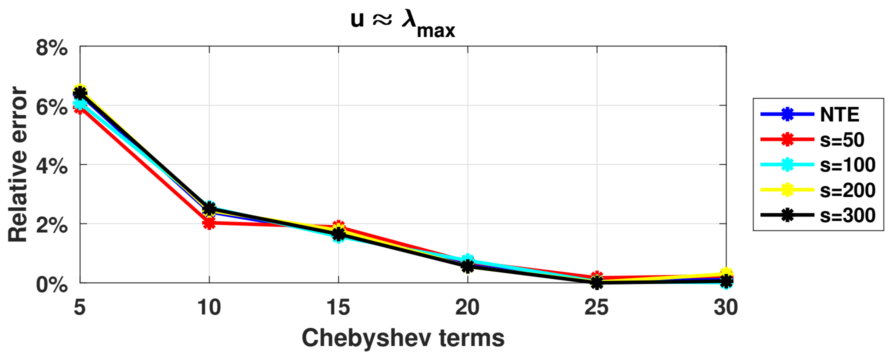


Figure 4.11.: Relative error for $5,000 \times 5,000$ density matrix with the top-50 eigenvalues decaying linearly using the Taylor and the Chebyshev approximation algorithms with $u = \lambda_{\max}$.

matrices we tested were $4,096 \times 4,096$ and $16,384 \times 16,384$. We also generated much larger $30,000 \times 30,000$ random matrices on which we only experimented with the Gaussian random projection matrix.

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

We start by reporting results for Algorithm 15 using the Gaussian, the subsampled randomized Hadamard transform (Algorithm 4), and the input-sparsity transform (Algorithm 5) random projection matrices. Consider the $4,096 \times 4,096$ low rank

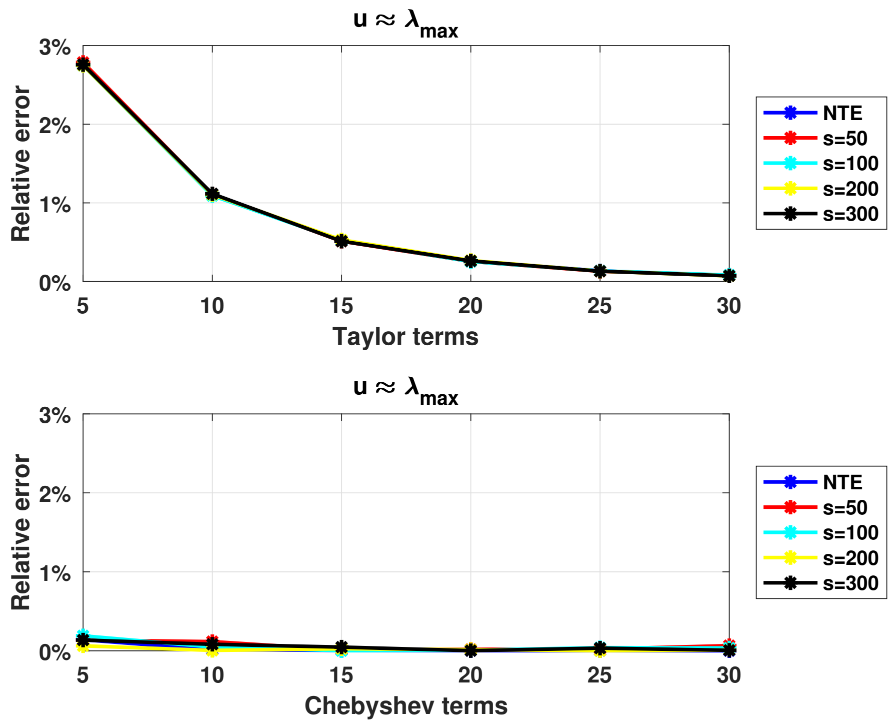


Figure 4.12.: Relative error for $5,000 \times 5,000$ density matrix with the top-1000 eigenvalues decaying linearly using the Taylor and the Chebyshev approximation algorithms with $u = \lambda_{\max}$.

density matrices and let k, the rank of the matrix, be 10, 50, 100, and 300. Let s, the number of columns of the random projection matrix, range from 50 to $1,000$ in increments of 50. Figures 4.20 and 4.21 depict the relative error (out of 100%) for all combinations of k and s. We also report the wall-clock running times for values of s between 300 and 450 at Figure 4.22.

We observe that in the case of the random matrix with exponentially decaying eigenvalues and for all algorithms the relative error is under 0.3% for any choice of the parameters k and s and, as expected, decreases as the dimension of the projection space s grows larger. Interestingly, all three random projection matrices returned essentially identical accuracies and very comparable wall-clock running time results. This observation is due to the fact that for all choices of k, after scaling the matrix to

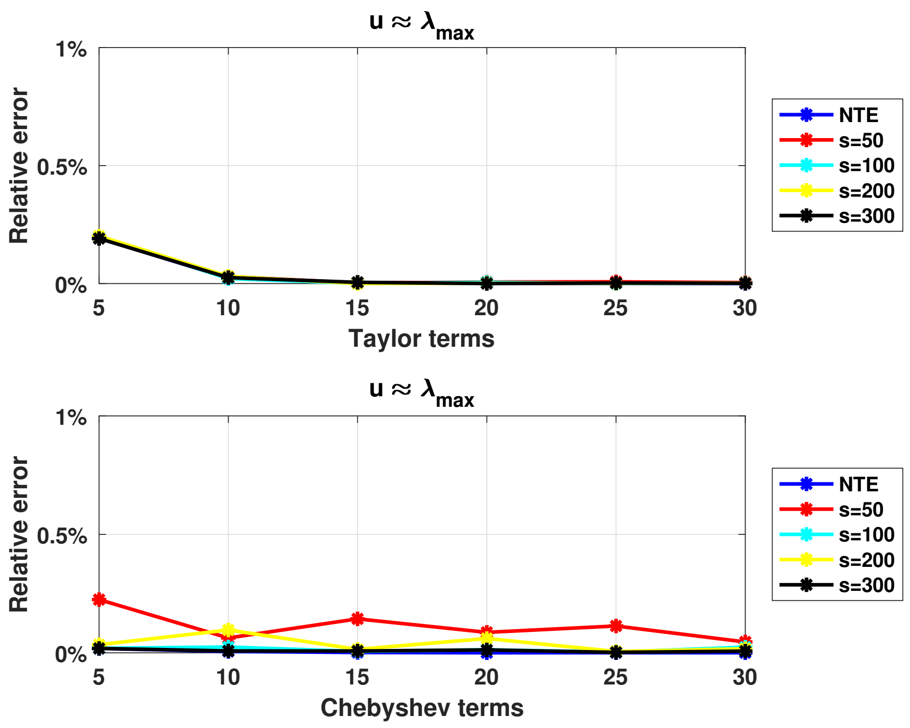
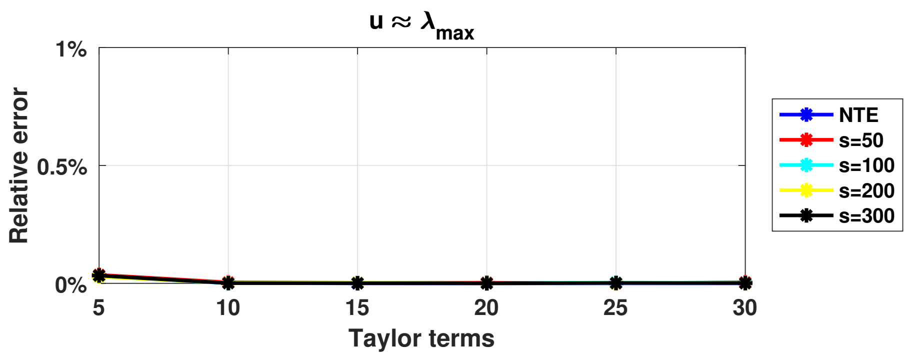


Figure 4.13.: Relative error for $5,000 \times 5,000$ density matrix with the top-3500 eigenvalues decaying linearly using the Taylor and the Chebyshev approximation algorithms with $u = \lambda_{\max}$.

unit trace, the only eigenvalues that were numerically non-zero were the 10 dominant ones.

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

The running time depends not only on the size of the matrix, but also on its rank, e.g. for $k = 100$ and $s = 450$, our approximation was computed in about 2.5



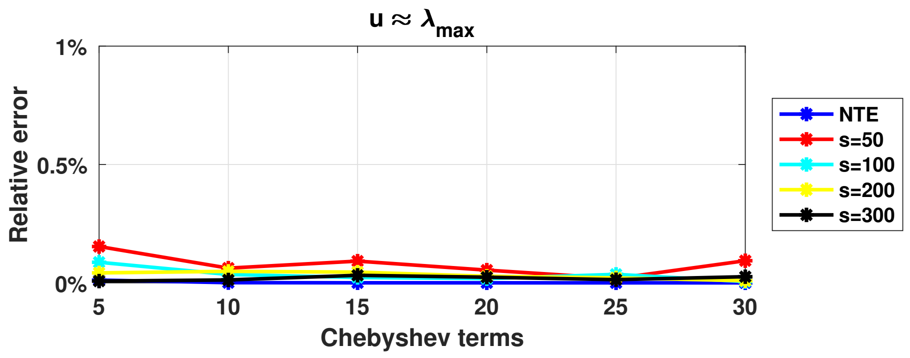


Figure 4.14.: Relative error for $5,000 \times 5,000$ density matrix with the top-5000 eigenvalues decaying linearly using the Taylor and the Chebyshev approximation algorithms with $u = \lambda_{\max}$.

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

The second dataset we experimented with was a $16,384 \times 16,384$ low rank density matrix. We set $k = 50$ and $k = 500$ and we let s take values in the set $\{500,\ 1000,\ 1500, \ldots, 3000,\ 3500\}$. We report the relative error (out of 100%) for all

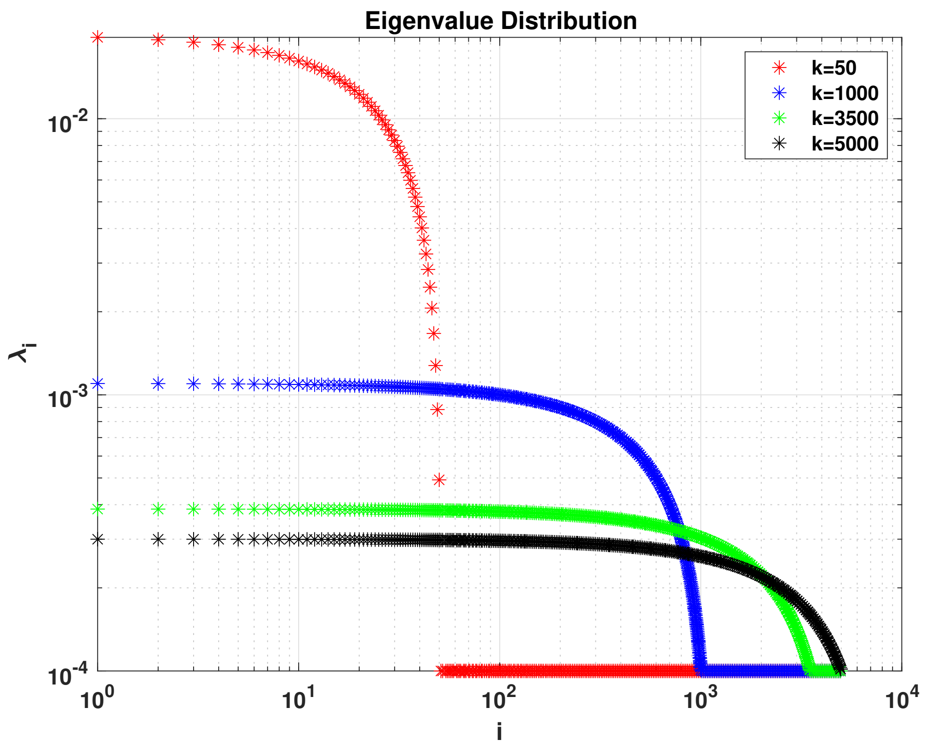


Figure 4.15.: Eigenvalue distribution of $5,000 \times 5,000$ density matrices with the top-$k = \{50,\ 1000,\ 3500,\ 5000\}$ eigenvalues decaying linearly and the remaining ones $(5,000 - k)$ following a uniform distribution.

combinations of k and s in Figure 4.23 for the matrix with exponentially decaying eigenvalues and in Figure 4.24 for the matrix with linearly decaying eigenvalues. We also report the wall-clock running times for s between 500 and $2,000$ in Figure 4.25. We observe that the relative error is typically around 1% for both types of matrices, with running times ranging between ten seconds and four minutes, significantly outperforming the exact entropy computation which took approximately 1.6 minutes for the rank 50 approximation and 20 minutes for the rank 500 approximation.

The last dataset we experimented with was a $30,000 \times 30,000$ low rank density matrix on which we ran Algorithm 15 using a Gaussian random projection matrix. We set $k = 50$ and $k = 500$ and we let s take values in the set $\{500,\ 1000,\ 1500, \ldots, 3000,\ 3500\}$. We report the relative error (out of 100%) for all combinations of k and s in Fig-

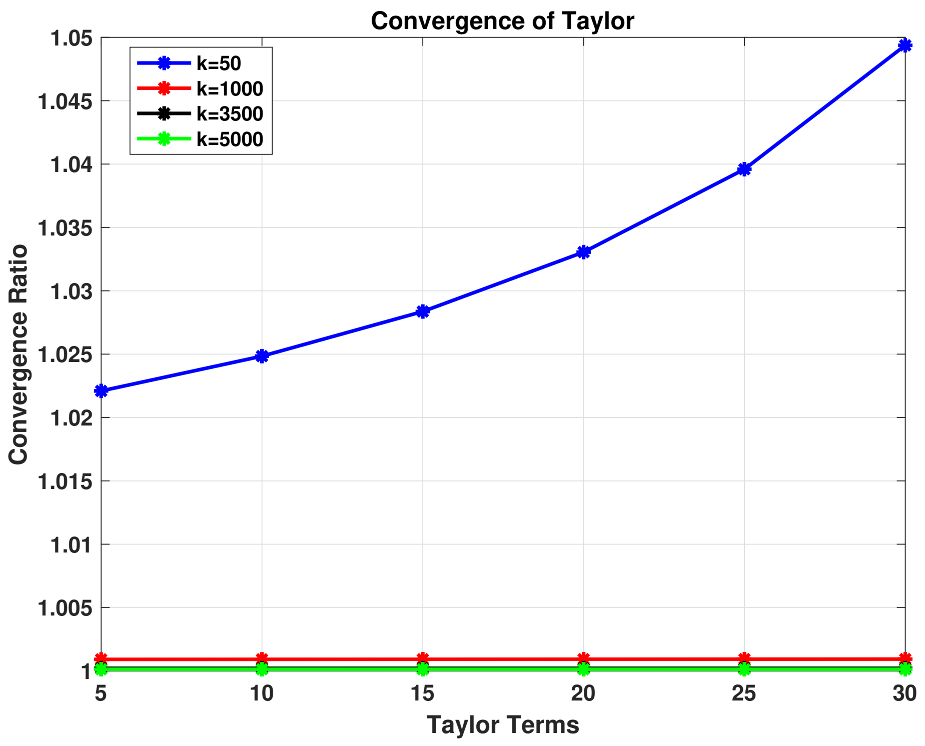


Figure 4.16.: Convergence radius of the Taylor polynomial for the $5,000 \times 5,000$ density matrices with the top-$k = \{50,\ 1000,\ 3500,\ 5000\}$ eigenvalues decaying linearly and the remaining ones $(5,000 - k)$ following a uniform distribution.

ure 4.26 for the matrix with exponentially decaying eigenvalues and in Figure 4.27 for the matrix with the linearly decaying eigenvalues. We also report the wall-clock running times for s ranging between 500 and $2,000$ in Figure 4.28. We observe that the relative error is typically around 1% for both types of matrices, with the running times ranging between 30 seconds and two minutes, outperforming the exact entropy which was computed in six minutes for the rank 50 approximation and in one hour for the rank 500 approximation.

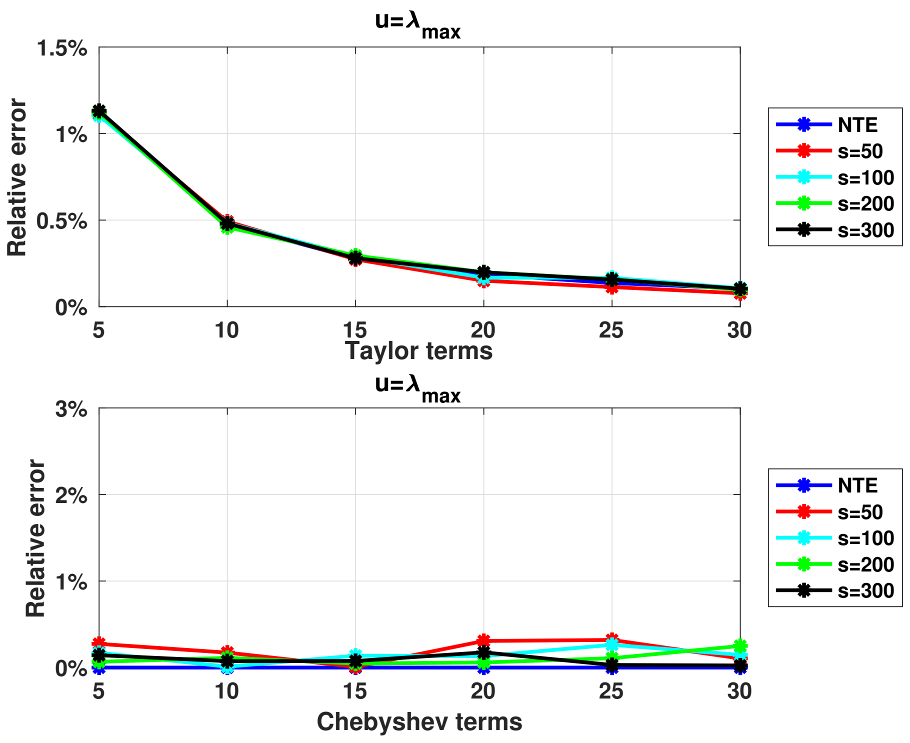


Figure 4.17.: Relative error for $5,000 \times 5,000$ Hermitian density matrix using the Taylor and Chebyshev approximation algorithms.

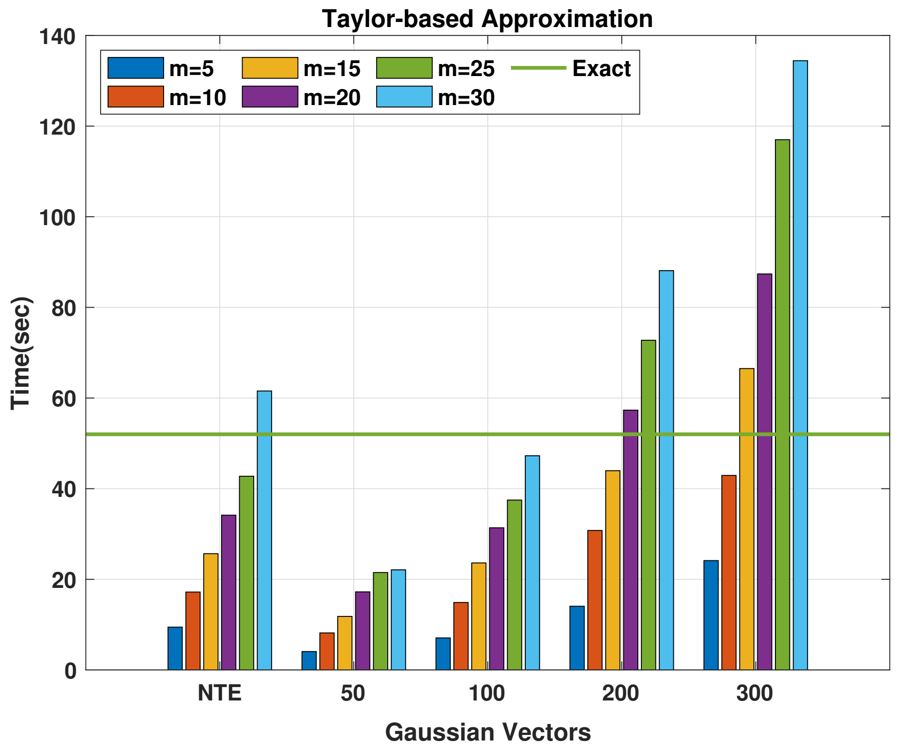


Figure 4.18.: Time (in seconds) to run the Taylor-based algorithm for the $5,000 \times 5,000$ density matrix for all combinations of m and s. *Exactly* computing the Von Neumann entropy took approximately 52 seconds, designated by the straight horizontal line in the figure.

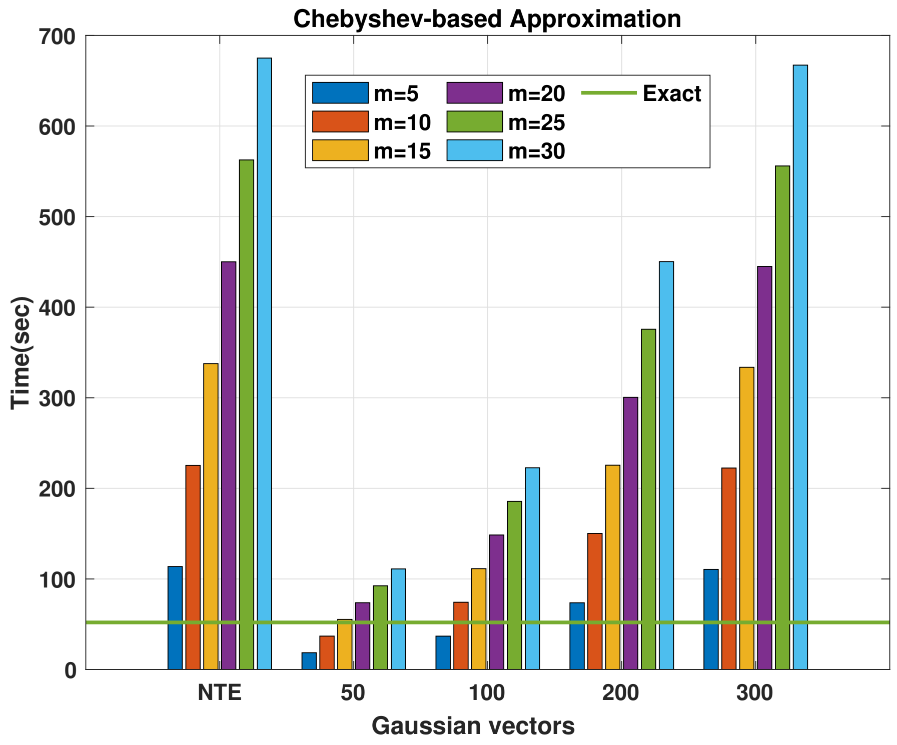


Figure 4.19.: Time (in seconds) to run the Chebyshev-based algorithm for the $5,000 \times 5,000$ density matrix for all combinations of m and s. *Exactly* computing the Von Neumann entropy took approximately 52 seconds, designated by the straight horizontal line in the figure.

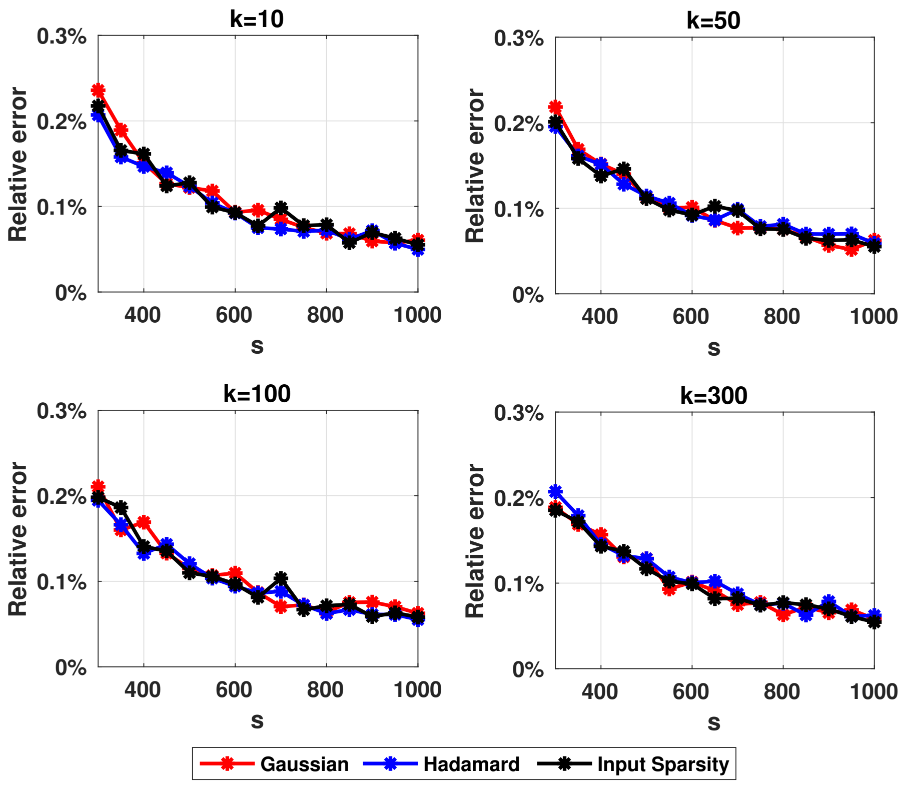


Figure 4.20.: Relative error for the $4,096 \times 4,096$ rank-k density matrix with exponentially decaying eigenvalues using Algorithm 15 with the Gaussian (red), the subsampled randomized Hadamard transform (blue), and the input sparsity transform (black) random projection matrices.

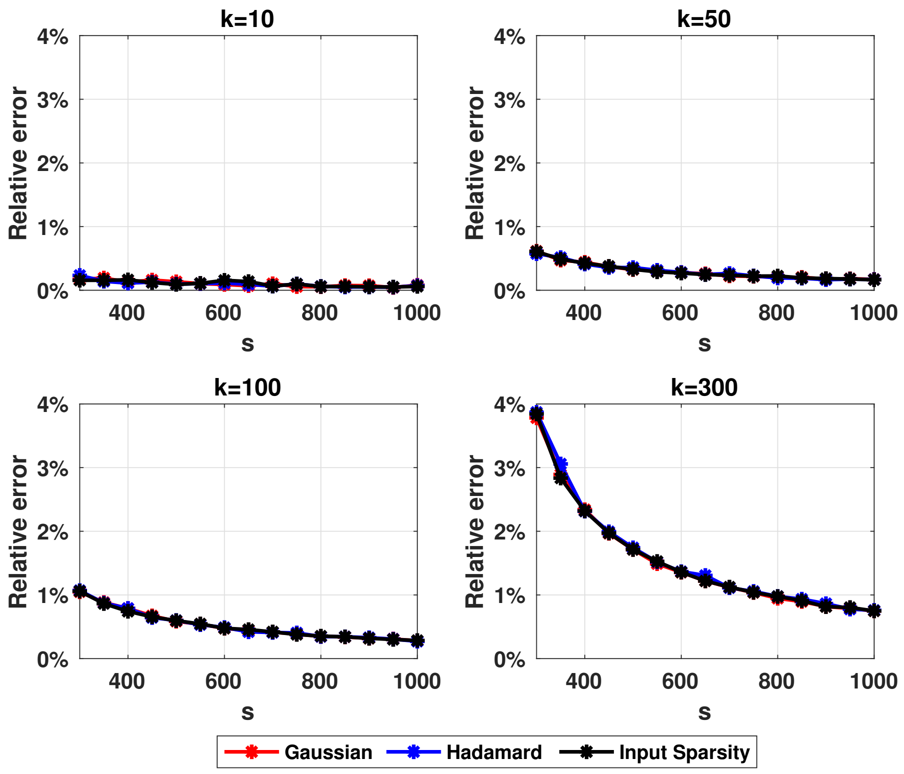


Figure 4.21.: Relative error for the $4,096 \times 4,096$ rank-k density matrix with linearly decaying eigenvalues using Algorithm 15 with the Gaussian (red), the subsampled randomized Hadamard transform (blue), and the input sparsity transform (black) random projection matrices.

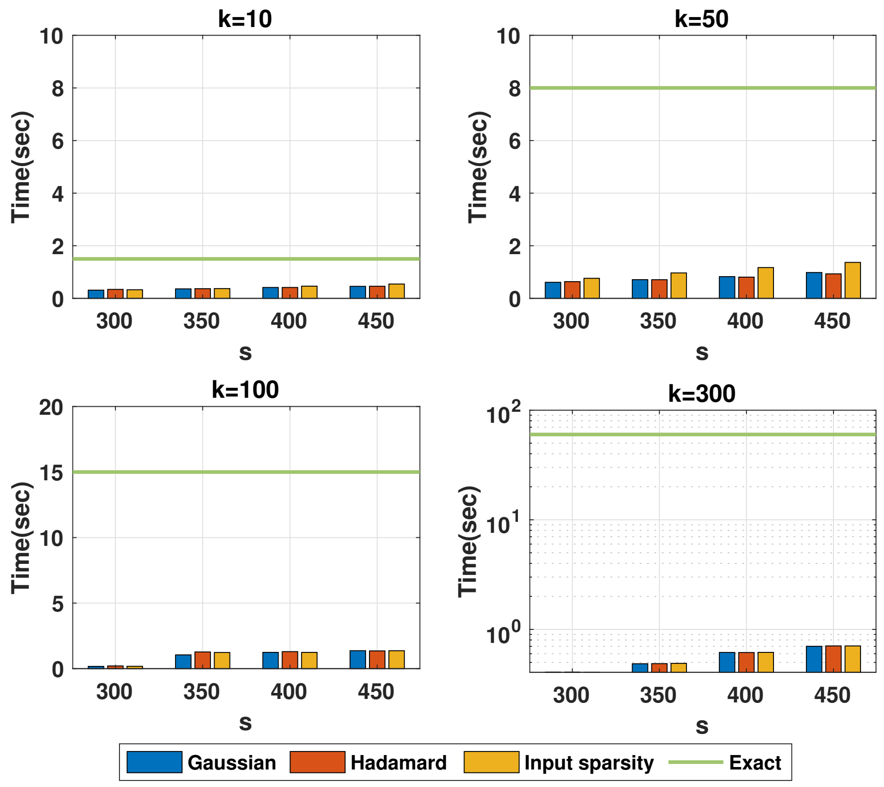


Figure 4.22.: Wall-clock times: Algorithm 15 on $4,096 \times 4,096$ random matrices, with the Gaussian (blue), the subsampled randomized Hadamard transform (red) and the input sparsity transform (orange) projection matrices. The exact entropy was computed in 1.5 seconds for the rank-10 approximation, in eight seconds for the rank-50 approximation, in 15 seconds for the rank-100 approximation, and in one minute for the rank-300 approximation as depicted by the straight horizontal lines in the figure.

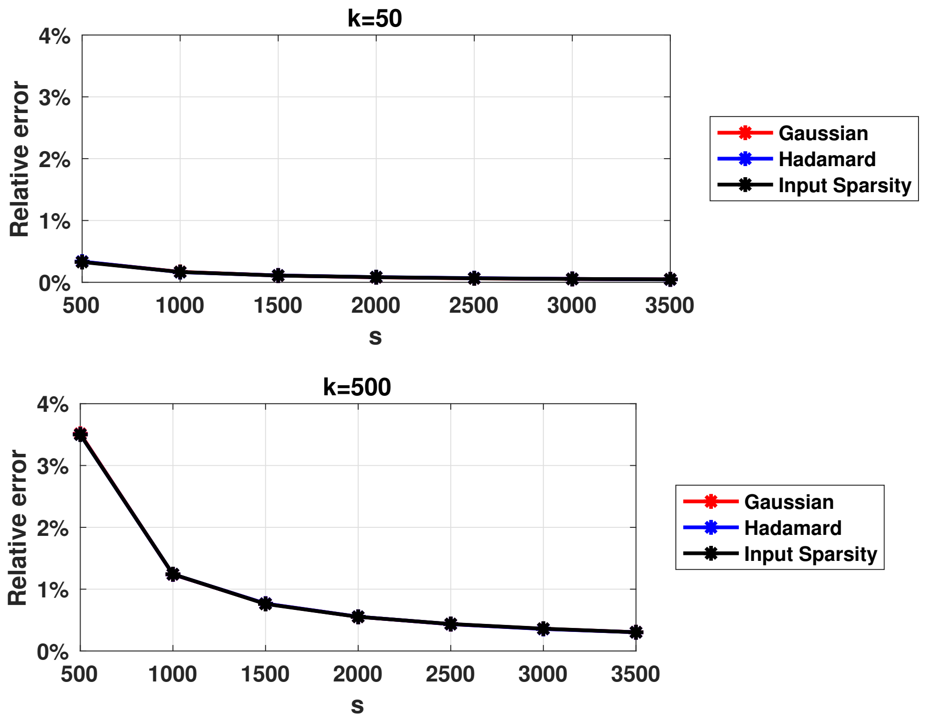


Figure 4.23.: Relative error for the $16,384 \times 16,384$ rank-k density matrix with exponentially decaying eigenvalues using Algorithm 15 with the Gaussian (red), the subsampled randomized Hadamard transform (blue), and the input sparsity transform (black) random projection matrices.

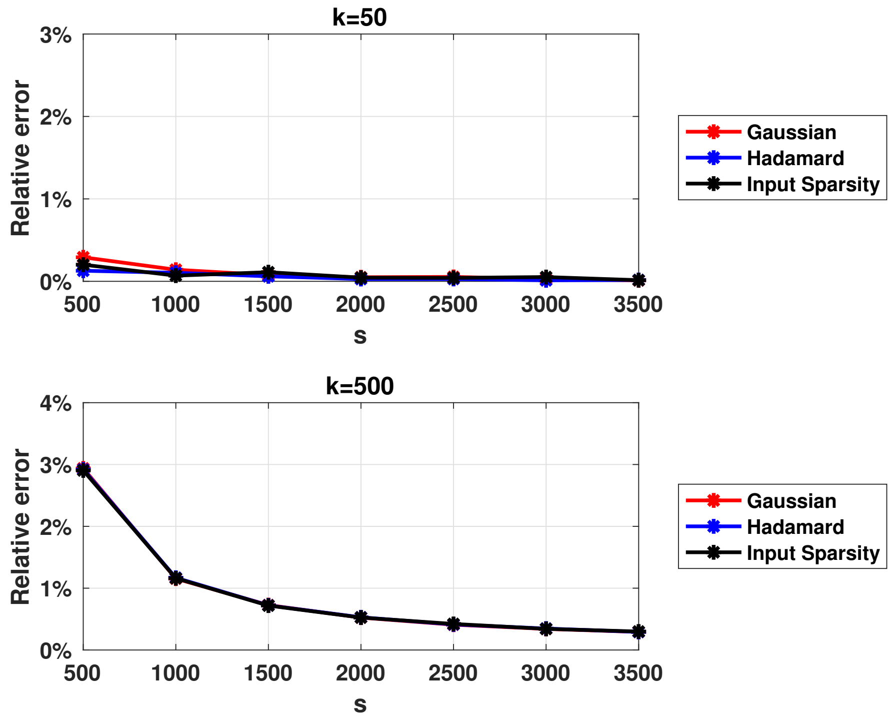


Figure 4.24.: Relative error for the $16,384 \times 16,384$ rank-k density matrix with linearly decaying eigenvalues using Algorithm 15 with the Gaussian (red), the subsampled randomized Hadamard transform (blue), and the input sparsity transform (black) random projection matrices.

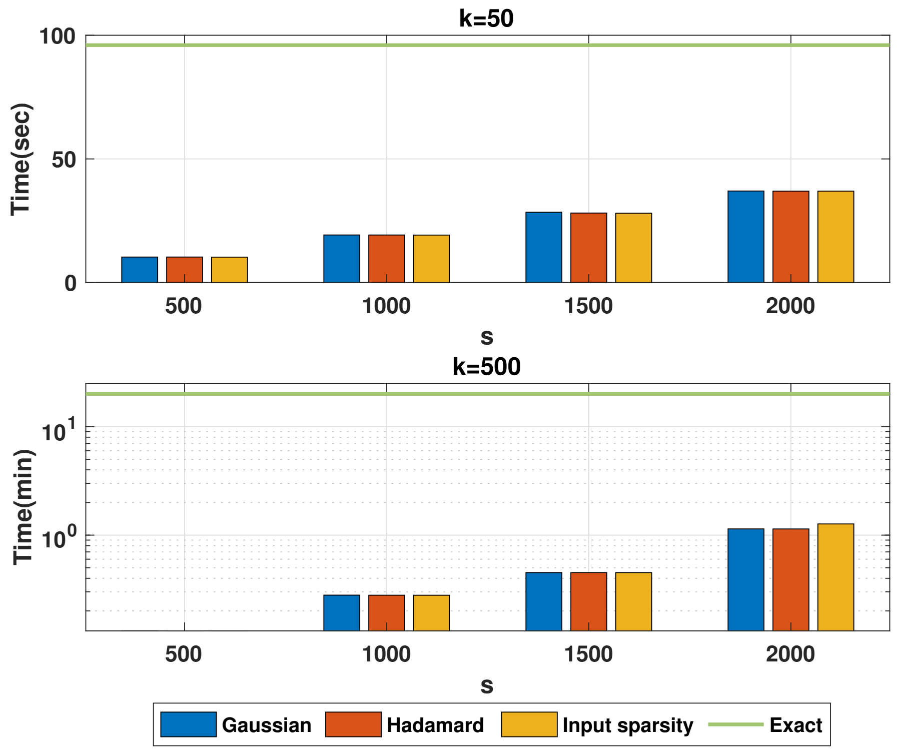


Figure 4.25.: Wall-clock times of Algorithm 15 for the $16,384 \times 16,384$ rank-k density matrix with linearly decaying eigenvalues using the Gaussian (blue), the subsampled randomized Hadamard transform (red) and the input sparsity transform (orange) projection matrices. The exact entropy was computed in 1.6 minutes for the rank 50 approximation and in 20 minutes for the rank 500 approximation, as depicted by the straight horizontal lines in the figure.

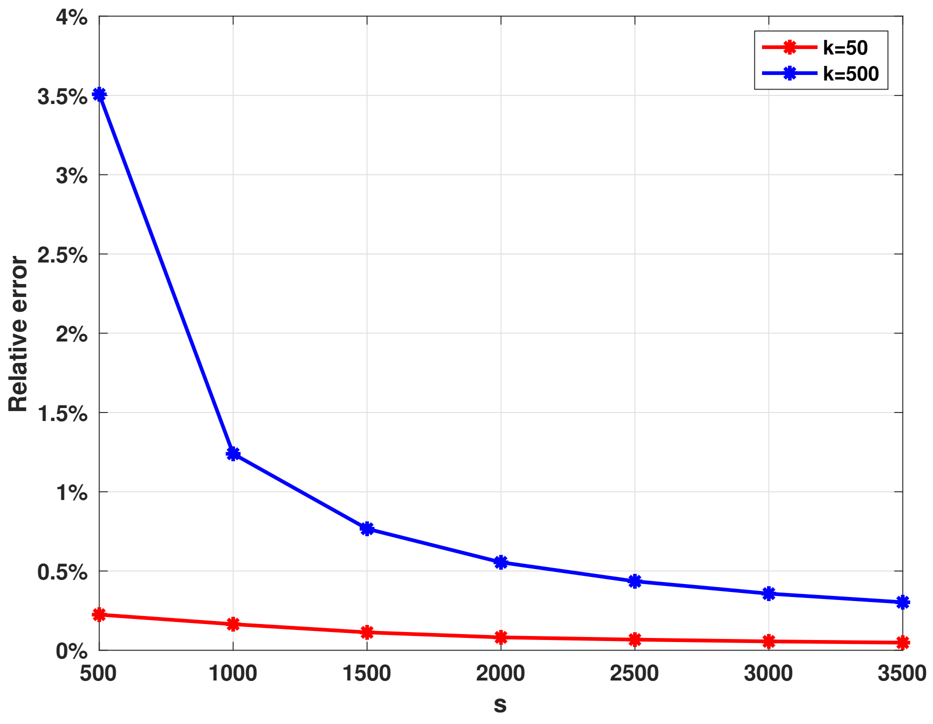


Figure 4.26.: Relative error for the $30,000 \times 30,000$ rank-k density matrix with exponentially decaying eigenvalues using Algorithm 15 with the Gaussian random projection matrix for $k = 50$ (red) and for $k = 500$ (blue).

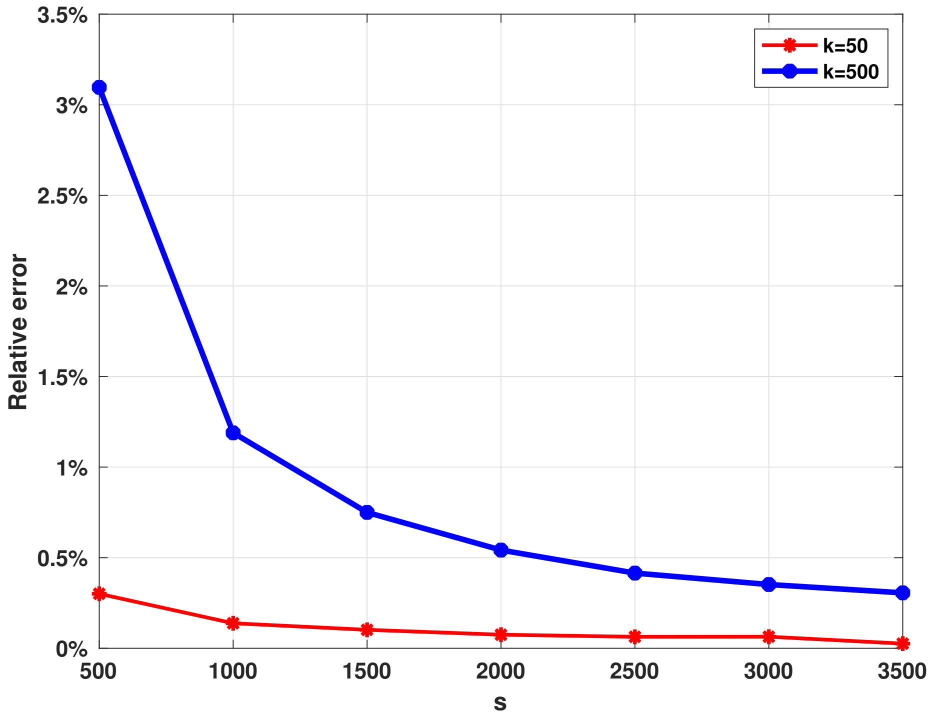


Figure 4.27.: Relative error for the $30,000 \times 30,000$ rank-k density matrix with linearly decaying eigenvalues using Algorithm 15 with the Gaussian random projection matrix for $k = 50$ (red) and for $k = 500$ (blue).

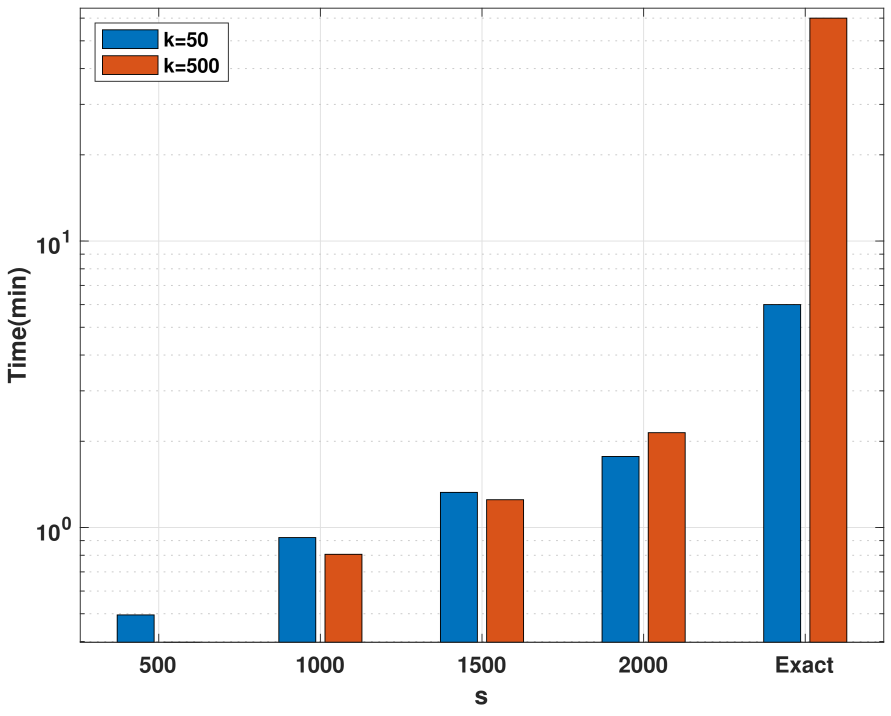


Figure 4.28.: Wall-clock times: rank-50 approximation (blue) and rank-500 approximation (red). Exact computation needed about six minutes and one hour respectively as depicted by the "Exact" bars in the figure.

5 A RANDOMIZED ROUNDING ALGORITHM FOR SPARSE PRINCIPAL COMPONENT ANALYSIS (PCA)

Chapter 5 presents our two-step randomized rounding algorithm to estimate sparse principal components. We further analyze our algorithm providing an additive-error guarantee to the optimal solution of eqn. (1.14).

This chapter is structured as follows: in Section 5.1 we provide the setting under which we will work through out the chapter. In Section 5.2 we present our algorithm and its analysis. In Section 5.3 we provide an extensive empirical evaluation to emphasize the applicability and competitiveness of our algorithm.

Sections of chapter 5 have been published in [FKKD17]

A Randomized Rounding Algorithm for Sparse PCA, K. Fountoulakis, A. Kundu, E-M. Kontopoulou, P. Drineas, in the ACM Transactions on Knowledge Discovery from Data (2017), Vol. 11(3), No. 38, pp. 1-26

5.1 Setting

We focus on real input matrices $\mathbf{X} \in \mathbb{R}^{m\times n}$ with covariance matrix $\mathbf{A} = \mathbf{X}^\top\mathbf{X} \in \mathbb{R}^{n\times n}$. Our goal is to compute a sparse vector $\hat{\mathbf{x}}_{opt}$ that will approximate, given certain guarantees, the solution, $\tilde{\mathbf{x}}_{opt}$, of eqn. (5.1) that follows. First, notice that the minimization problem of eqn. (1.13) although convex, features a non-convex constraint.

In our problem setting we relax constraint (1.14b) to the tighter ℓ_1 constraint (see, eqn. (5.1b)).

$$\tilde{\mathcal{Z}}_{opt} = \max_{\mathbf{x} \in \mathbb{R}^n} \quad \mathbf{x}^\top \mathbf{A}\, \mathbf{x} \tag{5.1a}$$

$$\text{s.t.} \quad \|\mathbf{x}\|_1 \leq \sqrt{k} \tag{5.1b}$$

$$\|\mathbf{x}\|_2 \leq 1 \tag{5.1c}$$

5.2 Randomized rounding algorithm

It is important to mention that problem (5.1) is difficult and all we can hope in practice is to calculate a stationary point. However, one should not discount the quality of stationary points. In Section 5.3 we show that by calculating stationary points we capture as much of the variance as computationally expensive convex relaxations. Having said that, the theoretical analysis of Section 5.2.2 assumes that we work with the globally optimal solution of problem (5.1c).

5.2.1 Algorithm

Let $\tilde{\mathbf{x}}_{opt}$ be a vector that achieves the optimal value $\tilde{\mathcal{Z}}_{opt}$ for problem (5.1). Clearly, $\tilde{\mathbf{x}}_{opt}$ is not necessarily sparse. Therefore, we employ a randomized rounding strategy to sparsify it by keeping larger entries of $\tilde{\mathbf{x}}_{opt}$ with higher probability. Specifically, we employ Algorithm 16 on the vector $\tilde{\mathbf{x}}_{opt}$ to get a sparse vector $\hat{\mathbf{x}}_{opt}$ that is our approximate solution to the sparse PCA formulation of eqn. (1.14).

Algorithm 16 Vector sparsification

```
Input: $\mathbf{x} \in \mathbb{R}^n$, integer $s > 0$.
Output: $\hat{\mathbf{x}} \in \mathbb{R}^n$ with $\mathbf{E}[\|\hat{\mathbf{x}}\|_0] \leq s$.
1: for $i = 1, \ldots, n$ do
2:     $p_i = \min\left\{\frac{s|\mathbf{x}_i|}{\|\mathbf{x}\|_1}, 1\right\}$
3:     $\hat{\mathbf{x}}_i = \begin{cases} \frac{1}{p_i}\mathbf{x}_i, & \text{with probability } p_i. \\ 0, & \text{otherwise.} \end{cases}$
4: end for
```

Surprisingly, this simple randomized rounding approach has not been analyzed in prior work on Sparse PCA. Theorem 12 is our main theoretical result and guarantees an additive error approximation to the NP-hard problem of eqn. (1.14).

Algorithm 17 sketches our algorithm for sparse PCA. Steps 2,-4 describe the approach we followed to compute the stationery point, that is a solution to problem (5.1). This procedure can also be described as a projected gradient ascent (see, [JNRS10, Section 4]).

Algorithm 17 Sparse PCA with vector sparsification

```
Input: $\mathbf{A} \in \mathbb{R}^{n \times n}$, integer $k > 0$, $u$ estimation to the dominant eigenvalue.
Output: $\hat{\mathbf{x}} \in \mathbb{R}^n$, the solution to the relaxed Problem (5.1)
1: $\tilde{\mathbf{x}} = \mathbf{1}_n$.                                    ▷ initialize $\tilde{\mathbf{x}}$ to all ones.
2: repeat
3:     Compute the gradient: $\mathbf{g} = -\mathbf{A}^\top \mathbf{A} \tilde{\mathbf{x}}$.
4:     Make a gradient step: $\tilde{\mathbf{x}} = \tilde{\mathbf{x}} - u^{-1}\mathbf{g}$.
5:     Project $\tilde{\mathbf{x}}$ onto the $\ell_1$ ball with radius $\sqrt{k}$.
6: until convergence
7: Use Algorithm 16 with input $\tilde{\mathbf{x}}$ and $k$ and return $\hat{\mathbf{x}}$.    ▷ sparsification step.
8: return $\hat{\mathbf{x}}$.
```

5.2.2 Error bound

For simplicity of presentation, we will assume that the rows (and therefore columns) of the matrix $\mathbf{A}$ have at most unit norms[1]. In words, Theorem 12 states that our sparse vector $\hat{\mathbf{x}}_{opt}$ is almost as good as the optimal vector $\mathbf{x}_{opt}$ in terms of capturing (with constant probability) almost as much of the spectrum of $\mathbf{A}$ as $\mathbf{x}_{opt}$ does. This comes at a penalty: the sparsity of $\mathbf{x}_{opt}$, which is equal to k, has to be relaxed to $\mathcal{O}(k/\varepsilon^2)$. This provides an elegant trade-off between sparsity and accuracy[2]. It is worth emphasizing that one should not worry about the small success probability of our approach: by repeating the rounding t times and keeping the vector $\hat{\mathbf{x}}_{opt}$ that

[1]We can relax this assumption by increasing s – our sampling factor – by a factor that depends on the upper bound of the row and column norms of $\mathbf{A}$.

[2]A less important artifact of our approach is the fact that the Euclidean norm of the vector $\hat{\mathbf{x}}_{opt}$ is slightly larger than one.

satisfies the second bound and maximizes $\hat{\mathbf{x}}_{opt}^\top \mathbf{A}\hat{\mathbf{x}}_{opt}$, we can immediately guarantee that we will achieve both bounds with probability at least $1-2^{-t}$.

Theorem 12. *Let $\mathbf{x}_{opt}$ be the optimal solution of the Sparse PCA problem of eqn.* (1.14) *satisfying $\|\mathbf{x}_{opt}\|_2 = 1$ and $\|\mathbf{x}_{opt}\|_0 \leq k$. Let $\hat{\mathbf{x}}_{opt}$ be the vector returned when Algorithm 16 is applied on the optimal solution $\tilde{\mathbf{x}}_{opt}$ of the optimization problem of eqn.* (5.1)*, with $s = 200k/\varepsilon^2$, where $\varepsilon \in (0,1]$ is an accuracy parameter. Then, $\hat{\mathbf{x}}_{opt}$ has the following properties:*

1. $\mathbf{E}\left[\|\hat{\mathbf{x}}_{opt}\|_0\right] \leq s$.

2. *With probability at least 3/4,*

$$\|\hat{\mathbf{x}}_{opt}\|_2 \leq 1 + 0.15\varepsilon.$$

3. *With probability at least 3/4,*

$$\hat{\mathbf{x}}_{opt}^\top \mathbf{A}\hat{\mathbf{x}}_{opt} \geq \mathbf{x}_{opt}^\top \mathbf{A}\mathbf{x}_{opt} - \varepsilon. \tag{5.2}$$

Proof. We begin the proof of Theorem 12 by proving a bound for $\mathbf{E}\left[\|\hat{\mathbf{x}}_{opt}\|_0\right]$. We continue by bounding $\|\hat{\mathbf{x}}_{opt}\|_2$, and we conclude by proving eqn. (5.2). For simplicity of notation we will drop the subscript *opt* from $\mathbf{x}_{opt}$, $\tilde{\mathbf{x}}_{opt}$ and $\hat{\mathbf{x}}_{opt}$ in all proofs of this section.

A bound for $\mathbf{E}\left[\|\hat{\mathbf{x}}\|_0\right]$ By definition (see, Algorithm 16) , $p_i \leq s\left|\tilde{\mathbf{x}}_i\right| / \|\tilde{\mathbf{x}}\|_1$, therefore

$$\mathbf{E}\left[\|\hat{\mathbf{x}}\|_0\right] = \sum_{i=1}^{n} p_i \leq \sum_{i=1}^{n} \frac{s\left|\tilde{\mathbf{x}}_i\right|}{\|\tilde{\mathbf{x}}\|_1} = s. \tag{5.3}$$

which proves the first bound in Theorem 12.

A bound for $\|\hat{\mathbf{x}}\|_2$ The following lemma immediately implies the second bound of Theorem 12 by setting $s = 200k/\varepsilon^2$.

Lemma 15. *Given our notation, with probability at least 3/4,*

$$\|\hat{\mathbf{x}}\|_2 \leq 1 + 2\sqrt{\frac{k}{s}}.$$

Proof. Consider the indicator random variables $\mathbb{D}_i$ for all $i = 1 \dots n$ which take the following values:

$$\mathbb{D}_i = \begin{cases} \frac{1}{p_i} & \text{, with probability } p_i \\ 0 & \text{, otherwise} \end{cases}$$

It is easy to see that $\hat{\mathbf{x}}_i = \mathbb{D}_i \tilde{\mathbf{x}}_i$ for all $i = 1 \dots n$. The following trivial properties hold for all i and will be used repeatedly in the proofs: $\mathbf{E}[\mathbb{D}_i] = 1$, $\mathbf{E}[1 - \mathbb{D}_i] = 0$, and $\mathbf{E}[1 - \mathbb{D}_i]^2 = p_i^{-1} - 1$.

It is more intuitive to provide a bound on the expectation of $\|\hat{\mathbf{x}} - \tilde{\mathbf{x}}\|_2^2$ and then leverage the triangle inequality in order to bound $\|\hat{\mathbf{x}}\|_2$. Using the indicator variables $\mathbb{D}_i$ and linearity of expectation, we get

$$\begin{aligned} \mathbf{E}\left[\|\hat{\mathbf{x}} - \tilde{\mathbf{x}}\|_2^2\right] &= \mathbf{E}\left[\sum_{i=1}^{n} (1 - \mathbb{D}_i)^2 \tilde{\mathbf{x}}_i^2\right] \\ &= \sum_{i=1}^{n} \tilde{\mathbf{x}}_i^2 \mathbf{E}[1 - \mathbb{D}_i]^2 \\ &= \sum_{i=1}^{n} \left(\frac{1}{p_i} - 1\right) \tilde{\mathbf{x}}_i^2. \end{aligned}$$

We will now prove the following inequality, which will be quite useful in later proofs:

$$\sum_{i=1}^{n} \left(\frac{1}{p_i} - 1\right) \tilde{\mathbf{x}}_i^2 \leq \frac{k}{s}. \tag{5.4}$$

Towards that end, we will split the set of indices $\{1 \dots n\}$ in two subsets: the set $I^{=1}$ corresponding to indices i such that $p_i = 1$ and the set $I^{<1}$ corresponding to indices i such that $p_i < 1$. Note that for all $i \in I^{<1}$ it must be the case that

$$p_i = \frac{s\,|\tilde{\mathbf{x}}_i|}{\|\tilde{\mathbf{x}}\|_1}.$$

We now proceed as follows:

$$\begin{aligned}
\mathbf{E}\left[\|\hat{\mathbf{x}}-\tilde{\mathbf{x}}\|_2^2\right] &= \sum_{i\in I^{=1}}\left(\frac{1}{p_i}-1\right)\tilde{\mathbf{x}}_i^2+\sum_{i\in I^{<1}}\left(\frac{1}{p_i}-1\right)\tilde{\mathbf{x}}_i^2 \\
&= \sum_{i\in I^{<1}}\left(\frac{1}{p_i}-1\right)\tilde{\mathbf{x}}_i^2 \\
&\le \sum_{i\in I^{<1}}\frac{1}{p_i}\tilde{\mathbf{x}}_i^2 \\
&= \sum_{i\in I^{<1}}\frac{\|\tilde{\mathbf{x}}\|_1}{s\left|\tilde{\mathbf{x}}_i\right|}\tilde{\mathbf{x}}_i^2 \\
&\le \frac{\|\tilde{\mathbf{x}}\|_1}{s}\sum_{i=1}^{n}\left|\tilde{\mathbf{x}}_i\right| \\
&= \frac{\|\tilde{\mathbf{x}}\|_1^2}{s}\le\frac{k}{s}. \qquad (5.5)
\end{aligned}$$

For the inequality of eqn. (5.5) we used the fact that $\|\tilde{\mathbf{x}}\|_1 \le \sqrt{k}$. We now use Markov's inequality to conclude that, with probability at least 3/4,

$$\|\hat{\mathbf{x}}-\tilde{\mathbf{x}}\|_2^2 \le \frac{4k}{s}. \qquad (5.6)$$

To conclude the proof note that, from the triangle inequality,

$$\|\hat{\mathbf{x}}-\tilde{\mathbf{x}}\|_2 \ge \left|\|\hat{\mathbf{x}}\|_2-\|\tilde{\mathbf{x}}\|_2\right|$$

and thus

$$\|\hat{\mathbf{x}}\|_2 \le \|\tilde{\mathbf{x}}\|_2+\|\hat{\mathbf{x}}-\tilde{\mathbf{x}}\|_2 \le 1+\|\hat{\mathbf{x}}-\tilde{\mathbf{x}}\|_2. \qquad (5.7)$$

Combining eqn. (5.6) with eqn. (5.6), after taking the square root of both sides, concludes the proof of the lemma. □

Proving eqn. (5.2) The following lemma states that the solution of problem (5.1) is at least as good as the solution of problem (1.14).

Lemma 16. *Given our notation,* $\mathbf{x}$ *is a feasible solution of the relaxed Sparse PCA formulation of eqn.* (5.1)*. Thus,* $\tilde{\mathcal{Z}}_{opt} = \tilde{\mathbf{x}}^\top \mathbf{A}\tilde{\mathbf{x}}^\top \geq \mathbf{x}^\top \mathbf{A}\mathbf{x}$.

Proof. Recall that $\mathbf{x}$ is a unit norm vector whose zero norm is at most k. Then, if we let $\mathbf{sgn}(\mathbf{x})$ denote the vector of signs for $\mathbf{x}$ (with the additional convention that if $\mathbf{x}_i$ is equal to zero then the i-th entry of $\mathbf{sgn}(\mathbf{x})$ is also set to zero), we get

$$\|\mathbf{x}\|_1 = \left|\mathbf{sgn}(\mathbf{x})^\top \mathbf{x}\right| \leq \|\mathbf{sgn}(\mathbf{x})\|_2 \|\mathbf{x}\|_2 \leq \sqrt{k}.$$

The second inequality follows since $\mathbf{sgn}(\mathbf{x})$ has at most k non-zero entries. Thus, $\mathbf{x}$ is feasible for the optimization problem of eqn. (5.1) and the conclusion of the lemma follows immediately. □

Getting a lower bound for $\hat{\mathbf{x}}^\top \mathbf{A}\hat{\mathbf{x}}$ is the toughest part of Theorem 12. Towards that end, the next lemma bounds the error $\left|\tilde{\mathbf{x}}^\top \mathbf{A}\tilde{\mathbf{x}} - \hat{\mathbf{x}}^\top \mathbf{A}\hat{\mathbf{x}}\right|$ as a function of two other quantities which will be easier to bound.

Lemma 17. *Given our notation,*

$$\left|\tilde{\mathbf{x}}^\top \mathbf{A}\tilde{\mathbf{x}} - \hat{\mathbf{x}}^\top \mathbf{A}\hat{\mathbf{x}}\right| \leq\ 2\left|\tilde{\mathbf{x}}^\top \mathbf{A}\left(\tilde{\mathbf{x}} - \hat{\mathbf{x}}\right)\right| + \left|\left(\hat{\mathbf{x}} - \tilde{\mathbf{x}}\right)^\top \mathbf{A}\left(\tilde{\mathbf{x}} - \hat{\mathbf{x}}\right)\right|.$$

Proof. We start with

$$\begin{aligned}\left|\tilde{\mathbf{x}}^\top \mathbf{A}\tilde{\mathbf{x}} - \hat{\mathbf{x}}^\top \mathbf{A}\hat{\mathbf{x}}\right| &= \left|\tilde{\mathbf{x}}^\top \mathbf{A}\tilde{\mathbf{x}} - \hat{\mathbf{x}}^\top \mathbf{A}\tilde{\mathbf{x}} + \hat{\mathbf{x}}^\top \mathbf{A}\tilde{\mathbf{x}} - \hat{\mathbf{x}}^\top \mathbf{A}\hat{\mathbf{x}}\right| \\ &\leq \left|\left(\tilde{\mathbf{x}} - \hat{\mathbf{x}}\right)^\top \mathbf{A}\tilde{\mathbf{x}}\right| + \left|\hat{\mathbf{x}}^\top \mathbf{A}\left(\tilde{\mathbf{x}} - \hat{\mathbf{x}}\right)\right|. \qquad (5.8)\end{aligned}$$

Next,

$$\begin{aligned}\left|\left(\hat{\mathbf{x}} - \tilde{\mathbf{x}}\right)^\top \mathbf{A}\left(\tilde{\mathbf{x}} - \hat{\mathbf{x}}\right)\right| &= \left|\hat{\mathbf{x}}^\top \mathbf{A}\left(\tilde{\mathbf{x}} - \hat{\mathbf{x}}\right) - \tilde{\mathbf{x}}^\top \mathbf{A}\left(\tilde{\mathbf{x}} - \hat{\mathbf{x}}\right)\right| \\ &\geq \left|\left|\hat{\mathbf{x}}^\top \mathbf{A}\left(\tilde{\mathbf{x}} - \hat{\mathbf{x}}\right)\right| - \left|\tilde{\mathbf{x}}^\top \mathbf{A}\left(\tilde{\mathbf{x}} - \hat{\mathbf{x}}\right)\right|\right|, \qquad (5.9)\end{aligned}$$

which implies

$$\left|\tilde{\mathbf{x}}^\top \mathbf{A}\left(\tilde{\mathbf{x}}-\hat{\mathbf{x}}\right)\right| \quad \leq \quad \left|\tilde{\mathbf{x}}^\top \mathbf{A}\left(\tilde{\mathbf{x}}-\hat{\mathbf{x}}\right)\right| + \left|\left(\hat{\mathbf{x}}-\tilde{\mathbf{x}}\right)^\top \mathbf{A}\left(\tilde{\mathbf{x}}-\hat{\mathbf{x}}\right)\right|. \tag{5.10}$$

We now combine eqns. (5.8) and (5.9) to get

$$\begin{aligned}\left|\tilde{\mathbf{x}}^\top \mathbf{A}\tilde{\mathbf{x}} - \hat{\mathbf{x}}^\top \mathbf{A}\hat{\mathbf{x}}\right| &\leq \left|\left(\tilde{\mathbf{x}}-\hat{\mathbf{x}}\right)^\top \mathbf{A}\tilde{\mathbf{x}}\right| + \left|\tilde{\mathbf{x}}^\top \mathbf{A}\left(\tilde{\mathbf{x}}-\hat{\mathbf{x}}\right)\right| + \left|\left(\hat{\mathbf{x}}-\tilde{\mathbf{x}}\right)^\top \mathbf{A}\left(\tilde{\mathbf{x}}-\hat{\mathbf{x}}\right)\right| \\ &= 2\left|\tilde{\mathbf{x}}^\top \mathbf{A}\left(\tilde{\mathbf{x}}-\hat{\mathbf{x}}\right)\right| + \left|\left(\hat{\mathbf{x}}-\tilde{\mathbf{x}}\right)^\top \mathbf{A}\left(\tilde{\mathbf{x}}-\hat{\mathbf{x}}\right)\right|.\end{aligned}$$

□

Our next lemma will provide a bound for the first of the two quantities of interest in Lemma 17.

Lemma 18. *Given our notation, with probability at least 7/8,*

$$\left|\tilde{\mathbf{x}}_{opt}^\top \mathbf{A}\left(\tilde{\mathbf{x}}_{opt}-\hat{\mathbf{x}}_{opt}\right)\right| \leq \sqrt{8k/s}.$$

Proof. Let $\mathbf{D} \in \mathbb{R}^{n\times n}$ be a diagonal matrix with entries $\mathbf{D}_{ii} = \mathbb{D}_i$ for all $i = 1\ldots n$. Hence, we can write $\hat{\mathbf{x}} = \mathbf{D}\tilde{\mathbf{x}}$. We have that

$$\left(\tilde{\mathbf{x}}-\hat{\mathbf{x}}\right)^\top \mathbf{A}\tilde{\mathbf{x}} = \tilde{\mathbf{x}}^\top\left(I_n - \mathbf{D}\right)\mathbf{A}\tilde{\mathbf{x}} = \tilde{\mathbf{x}}^\top \begin{bmatrix} (1-\mathbb{D}_1)\mathbf{A}_{1*}\tilde{\mathbf{x}} \\ (1-\mathbb{D}_2)\mathbf{A}_{2*}\tilde{\mathbf{x}} \\ \vdots \\ (1-\mathbb{D}_n)\mathbf{A}_{n*}\tilde{\mathbf{x}} \end{bmatrix} = \sum_{i=1}^{n}\left(1-\mathbb{D}_i\right)\tilde{\mathbf{x}}_i\mathbf{A}_{i*}\tilde{\mathbf{x}},$$

where $\mathbf{A}_{i*}$ is the i-th row of the matrix A as a row vector. Squaring the above expression, we get

$$\left(\left(\tilde{\mathbf{x}}-\hat{\mathbf{x}}\right)^\top \mathbf{A}\tilde{\mathbf{x}}\right)^2 \sum_{i=1}^{n}\sum_{j=1}^{n}\left(\tilde{\mathbf{x}}_i\mathbf{A}_{i*}^\top\tilde{\mathbf{x}}\right)\left(\tilde{\mathbf{x}}_j\mathbf{A}_{j*}\tilde{\mathbf{x}}\right)\left(1-\mathbb{D}_i\right)\left(1-\mathbb{D}_j\right). \tag{5.11}$$

Recall that $\mathbf{E}\left[1-\mathbb{D}_i\right] = 0$ for all i; thus, for all $i \neq j$, $1-\mathbb{D}_i$ and $1-\mathbb{D}_j$ are independent random variables and therefore the expectation of their product is equal to zero. Thus, we can simplify the above expression as follows:

$$\begin{aligned}\mathbf{E}\left[\left(\tilde{\mathbf{x}}-\hat{\mathbf{x}}\right)^\top \mathbf{A}\tilde{\mathbf{x}}\right]^2 &= \sum_{i=1}^n \mathbf{E}\left[1-\mathbb{D}_i\right]^2 \left(\tilde{\mathbf{x}}_i \mathbf{A}_{i*}\tilde{\mathbf{x}}\right)^2 \\ &= \sum_{i=1}^n \left(\frac{1}{p_i}-1\right)\tilde{\mathbf{x}}_i^2\left(\mathbf{A}_{i*}\tilde{\mathbf{x}}\right)^2 \\ &\leq \sum_{i=1}^n \left(\frac{1}{p_i}-1\right)\tilde{\mathbf{x}}_i^2.\end{aligned}$$

In the last inequality we used $|\mathbf{A}_{i*}\tilde{\mathbf{x}}| \leq \|\mathbf{A}_{i*}\|_2\|\tilde{\mathbf{x}}\|_2 \leq 1$. The last term in the above derivation can be bounded as shown in eqn. (5.4), and thus we conclude

$$\mathbf{E}\left[\left(\tilde{\mathbf{x}}-\hat{\mathbf{x}}\right)^\top \mathbf{A}\tilde{\mathbf{x}}\right]^2 \leq k/s.$$

Markov's inequality now implies that with probability at least 7/8

$$\left(\left(\tilde{\mathbf{x}}-\hat{\mathbf{x}}\right)^\top \mathbf{A}\tilde{\mathbf{x}}\right)^2 \leq 8k/s.$$

□

Our next lemma will provide a bound for the second of the two quantities of interest in Lemma 17. The proof of the lemma is tedious and a number of cases need to be considered.

Lemma 19. *Given our notation, with probability at least 7/8,*

$$\left|\left(\tilde{\mathbf{x}}_{opt}-\hat{\mathbf{x}}_{opt}\right)^\top \mathbf{A}\left(\tilde{\mathbf{x}}_{opt}-\hat{\mathbf{x}}_{opt}\right)\right| \leq \left(24k^2/s^2+6k^2/s^3+54\sqrt{k}/s\right)^{1/2}.$$

Proof. Using the indicator variables $\mathbb{D}_i$ and linearity of expectation, we get

$$\mathbf{E}\left[(\tilde{\mathbf{x}}-\hat{\mathbf{x}})^\top \mathbf{A}(\tilde{\mathbf{x}}-\hat{\mathbf{x}})\right]^2 = \mathbf{E}\left[\sum_{i,j=1}^n \mathbf{A}_{ij}\left(1-\mathbb{D}_i\right)\tilde{\mathbf{x}}_i\left(1-\mathbb{D}_j\right)\tilde{\mathbf{x}}_j\right]^2 =$$

$$\sum_{i_1,j_1,i_2,j_2=1}^n \mathbf{A}_{i_1j_1}\mathbf{A}_{i_2j_2}\tilde{\mathbf{x}}_{i_1}\tilde{\mathbf{x}}_{i_2}\tilde{\mathbf{x}}_{j_1}\tilde{\mathbf{x}}_{j_2} \times \mathbf{E}\left[\left(1-\mathbb{D}_{i_1}\right)\left(1-\mathbb{D}_{i_2}\right)\left(1-\mathbb{D}_{j_1}\right)\left(1-\mathbb{D}_{j_2}\right)\right]. \quad (5.12)$$

Recall that $\mathbf{E}\left[1-\mathbb{D}_i\right]=0$ for all i. Notice that if any of the four indices i_1, i_2, j_1, j_2 appears only once, then the expectation $\mathbf{E}\left[\left(1-\mathbb{D}_{i_1}\right)\left(1-\mathbb{D}_{i_2}\right)\left(1-\mathbb{D}_{j_1}\right)\left(1-\mathbb{D}_{j_2}\right)\right]$ corresponding to those indices equals zero. This expectation is non-zero if the four indices are paired in couples or if all four are equal. That is, non-zero expectation happens if

$$\begin{array}{lll}
(A) : & i_1 = i_2 \neq j_1 = j_2 & (n^2 - n \text{ terms}) \\
(B) : & i_1 = j_1 \neq i_2 = j_2 & (n^2 - n \text{ terms}) \\
(C) : & i_1 = j_2 \neq i_2 = j_1 & (n^2 - n \text{ terms}) \\
(D) : & i_1 = i_2 = j_1 = j_2 & (n \text{ terms}).
\end{array}$$

For case (A), let $i_1 = i_2 = k$ and let $j_1 = j_2 = \ell$, in which case the corresponding terms in eqn. (5.12) become (notice that $\mathbb{D}_k$ and $\mathbb{D}_\ell$ are independent random variables since $k \neq \ell$):

$$\sum_{k,\ell=1,\ k\neq\ell}^n \mathbf{A}_{k\ell}^2\tilde{\mathbf{x}}_k^2\tilde{\mathbf{x}}_\ell^2\mathbf{E}\left[\left(1-\mathbb{D}_k\right)^2\left(1-\mathbb{D}_\ell\right)^2\right] =$$

$$\sum_{k,\ell=1,\ k\neq\ell}^n \mathbf{A}_{k\ell}^2\tilde{\mathbf{x}}_k^2\tilde{\mathbf{x}}_\ell^2\left(\frac{1}{p_k}-1\right)\left(\frac{1}{p_\ell}-1\right) \leq$$

$$\sum_{k=1}^n \tilde{\mathbf{x}}_k^2\left(\frac{1}{p_k}-1\right)\sum_{\ell=1}^n \tilde{\mathbf{x}}_\ell^2\left(\frac{1}{p_\ell}-1\right) =$$

$$\left(\sum_{k=1}^n \tilde{\mathbf{x}}_k^2\left(\frac{1}{p_k}-1\right)\right)^2 \leq \frac{k^2}{s^2}. \quad (5.13)$$

In the first inequality we used $|\mathbf{A}_{k\ell}| \leq 1$ for all k and ℓ and added extra positive terms (corresponding to $k = \ell$), which reinforce the inequality. The last inequality follows from eqn. (5.4).

For case (B), let $i_1 = j_1 = k$ and let $i_2 = j_2 = \ell$, in which case the corresponding terms in eqn. (5.12) become (notice that $\mathbb{D}_k$ and $\mathbb{D}_\ell$ are independent random variables since $k \neq \ell$):

$$\begin{aligned} \sum_{k,\ell=1,\ k\neq\ell}^{n} \mathbf{A}_{kk}\mathbf{A}_{\ell\ell}\tilde{\mathbf{x}}_k^2\tilde{\mathbf{x}}_\ell^2 \mathbf{E}\left[(1-\mathbb{D}_k)^2(1-\mathbb{D}_\ell)^2\right] &= \\ \sum_{k,\ell=1,\ k\neq\ell}^{n} \mathbf{A}_{kk}\mathbf{A}_{\ell\ell}\tilde{\mathbf{x}}_k^2\tilde{\mathbf{x}}_\ell^2 \left(\frac{1}{p_k}-1\right)\left(\frac{1}{p_\ell}-1\right) &\leq \\ \sum_{k=1}^{n} \tilde{\mathbf{x}}_k^2 \left(\frac{1}{p_k}-1\right) \sum_{\ell=1}^{n} \tilde{\mathbf{x}}_\ell^2 \left(\frac{1}{p_\ell}-1\right) &= \\ \left(\sum_{k=1}^{n} \tilde{\mathbf{x}}_k^2 \left(\frac{1}{p_k}-1\right)\right)^2 \leq \frac{k^2}{s^2}. \end{aligned} \tag{5.14}$$

In the first inequality we used $\mathbf{A}_{kk} \leq 1$ for all k and the fact that the diagonal entries of a symmetric positive definite matrix are non-negative, which allows us to add extra positive terms (corresponding to $k = \ell$) to reinforce the inequality. The remainder of the derivation is identical to case (A).

For case (C), let $i_1 = j_2 = k$ and let $i_2 = j_1 = \ell$, in which case the corresponding terms in eqn. (5.12) become (notice that $\mathbb{D}_k$ and $\mathbb{D}_\ell$ are independent random variables since $k \neq \ell$):

$$\begin{aligned} \sum_{k,\ell=1,\ k\neq\ell}^{n} \mathbf{A}_{k\ell}\mathbf{A}_{\ell k}\tilde{\mathbf{x}}_k^2\tilde{\mathbf{x}}_\ell^2 \mathbf{E}\left[(1-\mathbb{D}_k)^2(1-\mathbb{D}_\ell)^2\right] &= \\ \sum_{k,\ell=1,\ k\neq\ell}^{n} \mathbf{A}_{k\ell}^2\tilde{\mathbf{x}}_k^2\tilde{\mathbf{x}}_\ell^2 \left(\frac{1}{p_k}-1\right)\left(\frac{1}{p_\ell}-1\right) &\leq \tfrac{k^2}{s^2}. \end{aligned} \tag{5.15}$$

In the first equality we used the fact that $\mathbf{A}$ is symmetric; the remainder of the derivation is identical to case (A).

Finally, for case (D), let $i_1 = i_2 = j_1 = j_2 = k$, in which case the corresponding terms in eqn. (5.12) become:

$$\begin{aligned}
\sum_{k=1}^{n} \mathbf{A}_{kk}^2 \tilde{\mathbf{x}}_k^4 \mathbf{E}\left[1 - \mathbb{D}_k\right]^4 &= \sum_{k=1}^{n} \mathbf{A}_{kk}^2 \tilde{\mathbf{x}}_k^4 \left(\frac{6}{p_k} - \frac{4}{p_k^2} + \frac{1}{p_k^3} - 3\right) \\
&\leq \sum_{k=1}^{n} \tilde{\mathbf{x}}_k^4 \left(\frac{6}{p_k} - \frac{4}{p_k^2} + \frac{1}{p_k^3} - 3\right) \\
&= \sum_{k\in I^{<1}} \tilde{\mathbf{x}}_k^4 \left(\frac{6}{p_k} - \frac{4}{p_k^2} + \frac{1}{p_k^3} - 3\right) \\
&\leq \sum_{k\in I^{<1}} \tilde{\mathbf{x}}_k^4 \left(\frac{6}{p_k} + \frac{1}{p_k^3}\right).
\end{aligned}$$

In the above derivation, we used $|\mathbf{A}_{kk}| \leq 1$. We also split the set of indices $\{1 \dots n\}$ in two subsets: the set $I^{=1}$ corresponding to indices k such that $p_k = 1$ and the set $I^{<1}$ corresponding to indices k such that $p_k < 1$. Note that for all $k \in I^{<1}$ it must be the case that $p_k = s\left|\tilde{\mathbf{x}}_i\right| / \|\tilde{\mathbf{x}}\|_1$. Thus,

$$\begin{aligned}
\sum_{k=1}^{n} \mathbf{A}_{kk}^2 \tilde{\mathbf{x}}_k^4 \mathbf{E}\left[1 - \mathbb{D}_k\right]^4 &\leq \sum_{k\in I^{<1}} \tilde{\mathbf{x}}_k^4 \left(\frac{6\|\tilde{\mathbf{x}}\|_1}{s\left|\tilde{\mathbf{x}}_k\right|} + \frac{\|\tilde{\mathbf{x}}\|_1^3}{s^3\left|\tilde{\mathbf{x}}_k\right|^3}\right) \\
&\leq \sum_{k=1}^{n} \tilde{\mathbf{x}}_k^4 \left(\frac{6\|\tilde{\mathbf{x}}\|_1}{s\left|\tilde{\mathbf{x}}_k\right|} + \frac{\|\tilde{\mathbf{x}}\|_1^3}{s^3\left|\tilde{\mathbf{x}}_k\right|^3}\right) \\
&= \frac{6\|\tilde{\mathbf{x}}\|_1}{s} \sum_{k=1}^{n} \left|\tilde{\mathbf{x}}_k\right|^3 + \frac{\|\tilde{\mathbf{x}}\|_1^3}{s^3} \sum_{k=1}^{n} \left|\tilde{\mathbf{x}}_k\right| \\
&= \frac{6\|\tilde{\mathbf{x}}\|_1 \|\tilde{\mathbf{x}}\|_3^3}{s} + \frac{\|\tilde{\mathbf{x}}\|_1^4}{s^3}.
\end{aligned}$$

Recall that $\|\tilde{\mathbf{x}}\|_q \leq \|\tilde{\mathbf{x}}\|_p$ for any $1 \leq p \leq q \leq \infty$ and use $\|\tilde{\mathbf{x}}\|_2 \leq 1$ and $\|\tilde{\mathbf{x}}\|_1 \leq \sqrt{k}$ to get

$$\begin{aligned}\sum_{k=1}^{n} \mathbf{A}_{kk}^2 \tilde{\mathbf{x}}_k^4 \mathbf{E}\left[1-\mathbb{D}_k\right]^4 &\leq \frac{6\|\tilde{\mathbf{x}}\|_1\|\tilde{\mathbf{x}}\|_2^3}{s} + \frac{\|\tilde{\mathbf{x}}\|_1^4}{s^3} \\ &\leq \frac{6\sqrt{k}}{s} + \frac{k^2}{s^3}. \qquad (5.16)\end{aligned}$$

Combining all four cases (i.e., eqns. (5.13), (5.14), (5.15), and (5.16)), we get

$$\mathbf{E}\left[(\tilde{\mathbf{x}}-\hat{\mathbf{x}})^\top \mathbf{A}\,(\tilde{\mathbf{x}}-\hat{\mathbf{x}})\right]^2 \leq 3k^2/s^2 + k^2/s^3 + 6\sqrt{k}/s.$$

Using Markov's inequality and taking square roots concludes the proof of the lemma.

□

We can now complete the proof of the lower bound for $\hat{\mathbf{x}}_{opt}^\top \mathbf{A}\hat{\mathbf{x}}_{opt}$. First, combine Lemmas 17, 18, and 19 to get

$$\left|\tilde{\mathbf{x}}_{opt}^\top \mathbf{A}\tilde{\mathbf{x}}_{opt} - \hat{\mathbf{x}}_{opt}\mathbf{A}\hat{\mathbf{x}}_{opt}\right| \leq 2\sqrt{8k/s} + \left(24k^2/s^2 + 6k^2/s^3 + 54\sqrt{k}/s\right)^{1/2}.$$

Since each of Lemmas 18 and 19 fail with probability at most 1/8, it follows from the union bound that the above inequality holds with probability at least $1-2(1/8) = 3/4$. Therefore, setting $s = 200k/\varepsilon^2$ guarantees (after some algebra) that with probability at least 3/4,

$$\left|\tilde{\mathbf{x}}_{opt}^\top \mathbf{A}\tilde{\mathbf{x}}_{opt} - \hat{\mathbf{x}}_{opt}^\top \mathbf{A}\hat{\mathbf{x}}_{opt}\right| \leq \varepsilon.$$

Using the triangle inequality and Lemma 16 we get that with probability at least 3/4

$$\hat{\mathbf{x}}_{opt}^\top \mathbf{A}\hat{\mathbf{x}}_{opt} \geq \tilde{\mathbf{x}}_{opt}^\top \mathbf{A}\tilde{\mathbf{x}}_{opt} - \varepsilon \geq \mathbf{x}_{opt}^\top \mathbf{A}\mathbf{x}_{opt} - \varepsilon,$$

which concludes the proof of eqn. (5.2). □

Running time The running time of Algorithm 17 is variable and decreases as the sparsity ratio increases. This is an artifact of the projected gradient ascent, we have implemented in steps 2 - 4 of Algorithm 17 to find a stationary point. The smaller the sparsity ratio is, the harder the problem (5.1) becomes for projected gradient ascent. This is also apparent from our empirical evaluation (see, Section 5.3).

5.3 Empirical evaluation

We perform experiments on both real and synthetic datasets. We chose to compare our algorithm with the solution returned by the state-of-the-art `Spasm` toolbox of [SCLE18], which implements the approach proposed in [ZHT06]. We also compare our solution with the simple `MaxComp` heuristic [CJ95]: this method computes the top singular vector of matrix $\mathbf{A}$ and returns a sparse vector by keeping the top k largest components (in absolute value) and setting the remaining ones to zero.

Let $\mathbf{X} \in \mathbb{R}^{m\times n}$ with $n \gg m$ denote the object-feature data matrix, where every column has zero mean, and recall that $\mathbf{A} = \mathbf{X}^\top\mathbf{X}$ in eqn. (5.1). We use the following function to measure the quality of an approximate solution $\mathbf{x} \in \mathbb{R}^n$ to the sparse PCA problem:

$$f(\mathbf{x}) = \frac{\mathbf{x}^\top\mathbf{A}\mathbf{x}}{\|\mathbf{A}\|_2^2}. \tag{5.17}$$

Notice that $0 \leq f(\mathbf{x}) \leq 1$ for all $\mathbf{x}$ which satisfy $\|\mathbf{x}\|_2 \leq 1$. The closer $f(\mathbf{x})$ is to one the more the vector $\mathbf{x}$ is parallel to the top singular vector of $\mathbf{A}$, or, equivalently, the closer $f(\mathbf{x})$ is to one the more $\mathbf{x}$ captures the variance of the matrix $\mathbf{A}$ that corresponds to its top singular value. Our goal is to calculate sparse vectors $\mathbf{x}$ with $f(\mathbf{x}) \approx 1$.

Our approach first finds a stationary point of the optimization problem of eqn. (5.1) and then uses Algorithm 16 (with $s = k$) to obtain a sparse solution vector $\hat{\mathbf{x}}_{opt} \in \mathbb{R}^n$. We note that the chosen value of s is much smaller than the $\mathcal{O}\left(k/\epsilon^2\right)$ value stipulated by Theorem 12. Also, in practice, our choice of s works very well and results in solutions that are comparable or better than competing methods in our data.

We note that in our use of `Spasm`, we used soft thresholding by setting the `STOP` parameter to $-\|\hat{\mathbf{x}}_{opt}\|_0$ and $\delta = -\infty$ (both `STOP` and δ are parameters of `Spasm` toolbox). This implies that the solutions obtained by `Spasm` and our approach will have the same number of non-zeros, thus making the comparison fair. Similarly, for `MaxComp`, after computing the top singular vector of $\mathbf{A}$, we select the $\|\hat{\mathbf{x}}_{opt}\|_0$ largest (in absolute value) coordinates to form the sparse solution. A final technical note is that the solutions obtained using either our method or `MaxComp` may result in vectors with non-unit Euclidean norms. In order to achieve a fair comparison in terms of eqn. (5.17), there are two options. The first one (*naive approach*) is to simply normalize the resulting vectors. However, a better approach (*SVD-based*) is possible: given a sparse solution vector $\hat{\mathbf{x}}_{opt}$, we could keep the rows and columns of $\mathbf{A}$ that correspond to the non-zero entries in $\hat{\mathbf{x}}_{opt}$ and compute the top singular vector of the induced matrix. Note that the induced matrix would be a $\|\hat{\mathbf{x}}_{opt}\|_0 \times \|\hat{\mathbf{x}}_{opt}\|_0$ matrix and its top singular vector would be a $\|\hat{\mathbf{x}}_{opt}\|_0$-dimensional vector. Obviously, this latter vector would be a unit norm vector and it could be padded with zeros to derive a unit norm vector in $\mathbb{R}^n$ with the same sparsity pattern as $\hat{\mathbf{x}}_{opt}$. It is straightforward to argue that this vector would improve the value of f compared to the naive normalized vector $\hat{\mathbf{x}}_{opt}/\|\hat{\mathbf{x}}_{opt}\|_2$. In our experimental evaluation, we will evaluate both the *naive* and the *SVD-based* normalization methods.

We also compare the methods based on their wall-clock running times. All experiments were run on a Intel(R) Core(TM) i7-6700 machine running at 3.40GHz, with 16 GB of RAM.

In some of our experiments we will need to extract more than one sparse singular vectors. Towards that end, we can use a deflation approach (Algorithm 18) to construct more than one sparse singular vectors by first projecting the residual matrix into the space that is perpendicular to the top sparse singular vector and then computing the top sparse singular vector of the residual matrix. In Algorithm 18, `rspca` refers to the solution of the optimization problem of eqn. (5.1) followed by Algorithm 16).

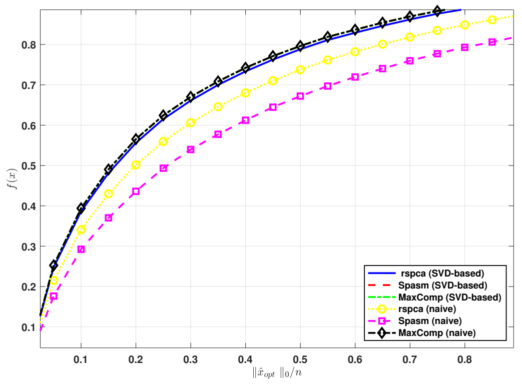
Algorithm 18 Computing k sparse principal components

```
Input: $\mathbf{X} \in \mathbb{R}^{m\times n}$, integer $k$.
Output: $\mathbf{U} = \{\mathbf{u}_1, \dots, \mathbf{u}_k\}$, $\mathbf{V} = \{\mathbf{v}_1, \dots, \mathbf{v}_k\}$.
1: $\mathbf{Y} = \mathbf{X}^\top$
2: $\mathbf{v}_1 = \texttt{rspca}(\mathbf{X})$ and $\mathbf{u}_1 = \texttt{rspca}(\mathbf{Y})$
3: for $i = 2, \dots, k$ do
4:     $\mathbf{X} = \mathbf{X} - \mathbf{X}\mathbf{v}_{i-1}\mathbf{v}_{i-1}^\top$ and $\mathbf{Y} = \mathbf{Y} - \mathbf{Y}\mathbf{u}_{i-1}\mathbf{u}_{i-1}^\top$
5:     $\mathbf{v}_i = \texttt{rspca}(\mathbf{X})$ and $\mathbf{u}_i = \texttt{rspca}(\mathbf{Y})$
6: end for
```

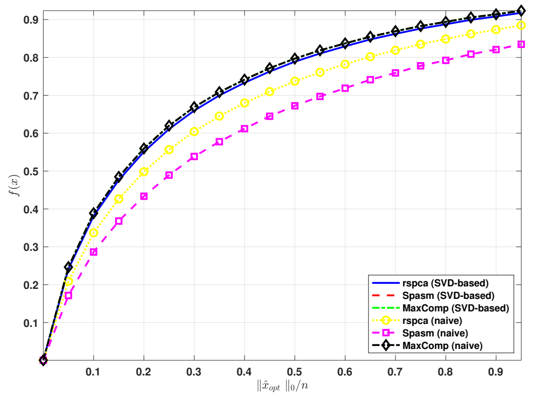
5.3.1 Empirical results for real datasets

We used 22 matrices (one for each chromosome) emerging from population genetics that encode all autosomal genotypes that are available in both the Human Genome Diversity Panel [Con07] and the HAPMAP [LAT$^+$08] project. Each of these matrices has approximately 2,500 samples (objects) described with respect to tens of thousands of features (Single Nucleotide Polymorphisms or SNPs); see [PLJD10] for details. We also used a gene expression dataset (GSE10072, lung cancer gene expression data) from the NCBI Gene Expression Omnibus database. This dataset includes 107 samples (58 lung tumor cases and 49 normal lung controls) measured using 22,215 probes (features) from the GPL96 platform annotation table. Both the population genetics and the gene expression datasets are interesting in the context of sparse PCA beyond numerical evaluations, since the sparse components can be directly interpreted to identify small sets of SNPs or genes that capture the data variance.

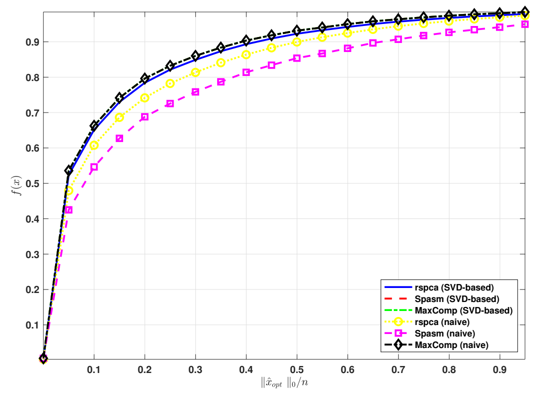
We present the performance of the different methods in Figures 5.1(a), 5.1(b), and 5.1(c). We note that in the population genetics data, our method has approximately the same or better performance compared to both `MaxComp` and `Spasm`. Not surprisingly, the *naive* normalization approach is consistently worse than the *SVD-based* one. It is worth noting that our *SVD-based* normalization approach easily improves the output of `Spasm`. This is because `Spasm` does detect the correct sparsity pattern but fails to compute the appropriate coefficients of the resulting sparse vectors.



(a) HapMap+HGDP data (chromosome 1): $n = 37493$.



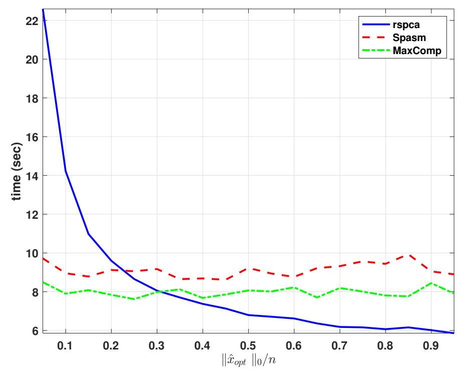
(b) HapMap+HGDP data (chromosome 2): $n = 40844$.



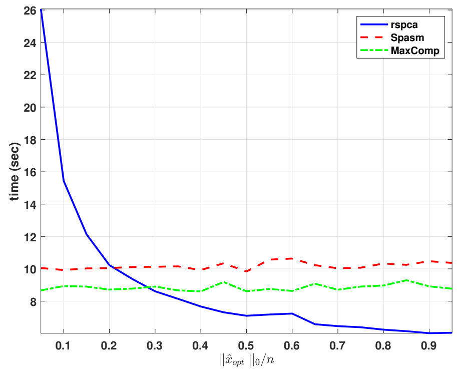
(c) Lung cancer gene expression data: $n = 22,215$.

Figure 5.1.: $f(\mathbf{x})$ vs. sparsity ratio $\|\hat{\mathbf{x}}_{opt}\|_0/n$ for HapMap+HGDP chromosomes one and two datasets and lung cancer gene expression dataset.

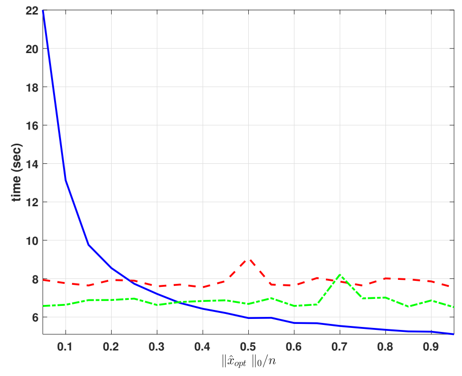
In Figure 5.2 we plot (in the y-axis) the running time for our method (`rspca`), the `Spasm` software, and the `MaxComp` heuristic on HapMap+HGDP chromosomes one up to four data. We also note that for each of the methods we use the `SVD-based` approach. The x-axis shows the sparsity ratio of the resulting vector. In practice we observed that the smaller k is the more iterations were needed for projected gradient ascent to converge since the algorithm balances between the standard PCA objective and the sparsity constraint. Notice that `MaxComp` is the fastest method, but it is less accurate. The running time of `MaxComp` and `Spasm` appears to be constant. This is because both methods require computation of the principal components. `MaxComp` is computing the first principal component which is then sparsified, while `Spasm` is computing the economy size SVD, to initialize the algorithm.



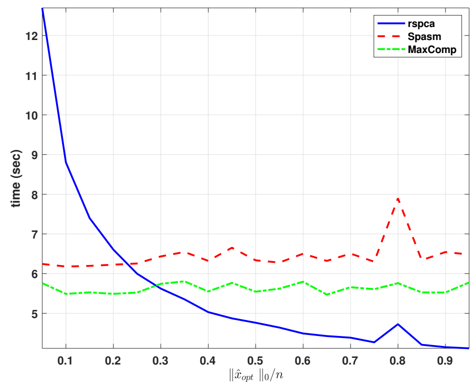
(a) HapMap+HGDP data (chromosome 1): $n = 37493$.



(b) HapMap+HGDP data (chromosome 2): $n = 40844$.



(c) HapMap+HGDP data (chromosome 3): $n = 34258$.



(d) HapMap+HGDP data (chromosome 4): $n = 30328$.

Figure 5.2.: Running time vs. sparsity ratio $\|\hat{\mathbf{x}}_{opt}\|_0/n$ for HapMap+HGDP chromosomes one to four data.

The performance plots for chromosomes three up to 22 and the running time plots for chromosomes five up to 22, can be found in [FKKD17, Appendix].

Our second real dataset comes from the field of text categorization and information retrieval. In such applications, documents are often represented as a "bag of words" and a vector space model is used. We use a subset of the Classic-3[3]

[3]ftp://ftp.cs.cornell.edu/pub/smart/

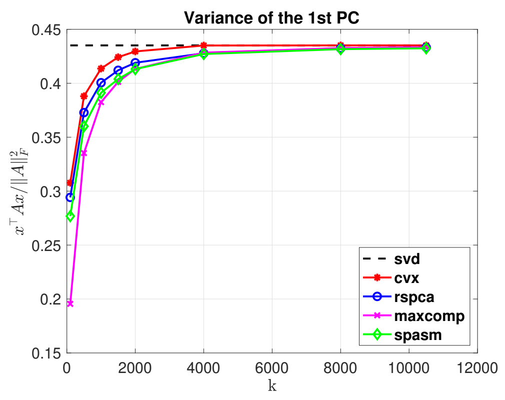
document collection, which we will call Classic-2. This subset consists of the CISI (Comités Interministériels pour la Société de l'Information) collection (1,460 information science abstracts) and the CRANFIELD collection (1,398 aeronautical systems abstracts). We created a document-by-term matrix using the Text-to-Matrix Generator (TMG) [ZG06][4]; the final matrix is a sparse $2,858 \times 12,427$ matrix with entries between zero and one, representing the weight of each term in the corresponding document.

First, we run Algorithm 18 to obtain two sparse singular vectors and we use the number of their non-zero entries to compute two sparse singular vectors for `MaxComp` and `Spasm`[5]. This way we can guarantee the same sparsity levels for all three pairs of singular vectors. We repeat this procedure for eight different values of k (sparsity parameter in eqn. (1.13)). Figure 5.3 depicts, for each k, the variance and the sparsity captured by the top two principal components using PCA, randomized sPCA (`rspca`), `MaxComp` heuristic and `Spasm` or solving eqn. (5.1c) (cvx).

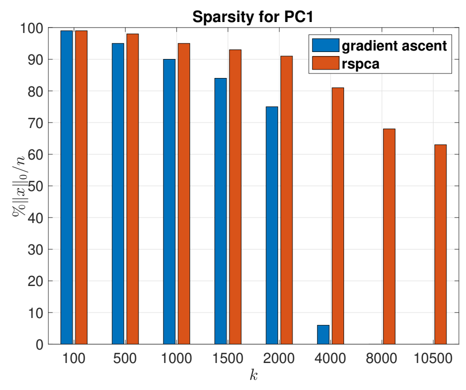
It seems that `Spasm` and `MaxComp` capture less variance than the `rspca`. Furthermore, the variance captured by `rspca` is constantly close to the one captured by the solution of eqn. (5.1c), but with a sparser component as Figure 5.3 indicates.

Table 5.1 summarizes the terms with non-zero weights in `rspca` principal components with sparsity parameter $k = 100$. The terms are ranked in descending order with respect to their weights. Notice that the first principal component reveals terms that appear more often in the CRANFIELD collection while the second principal component reveals terms that appear mostly in the CISI collection. CRANFIELD's terms are more singular than these of CISI's and they tend to dominate the singular vectors since they tend to appear more in the documents associated with the CRANFIELD collection than in the entire Classic-2 collection (e.g., the word "boundary" has one appearance in CISI and 459 in CRANFIELD). The exact opposite happens for terms in CISI: a significant amount of these terms appear with high weights in

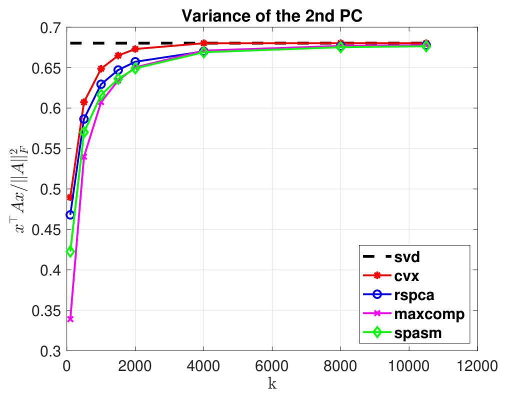
[4]The matrix was created using the stoplist provided by TMG, the `tf-idf` weighting scheme, and document normalization. We removed numbers and alphanumerics and we didn't use stemming.
[5]`Spasm` does not support sparse input matrices.



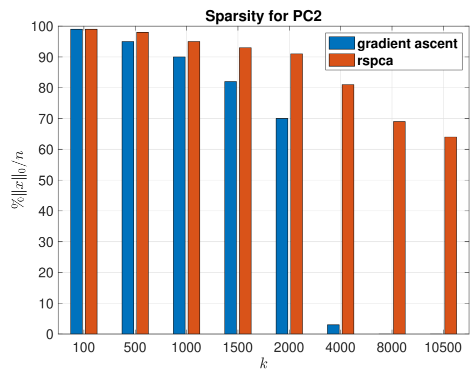
(a) Variance PC1



(b) Sparsity PC1



(c) Variance PC2



(d) Sparsity PC2

Figure 5.3.: Variance and sparsity captured by the principal components. PCA results in dense principal components, while `Spasm` and `MaxComp` share the same sparsity with `rspca`.

both collections (e.g., the word "information" has 664 appearances in CISI and 44 in CRANFIELD). This indicates that these terms will appear in singular vectors that do not separate the two collections.

Table 5.1.: Non-zero terms of the `rspca` when $k = 100$

1st Principal Component	**2nd Principal Component**
boundary	information
layer	library

continued on next page

Table 5.1.: *continued*

heat	retrieval
flow	systems
transfer	system
laminar	scientific
plate	science
shock	libraries
turbulent	research
hypersonic	layer
pressure	services
temperature	boundary
wall	indexing
flat	search
surface	literature
mach	heat
friction	university
compressible	documents
velocity	users
reynolds	classification
stagnation	technology
skin	
number	
stream	
equations	
transition	
solution	
viscous	

continued on next page

Table 5.1.: *continued*

layers
separation
solutions
gradient
injection
dimensional
incompressible
fluid
edge
cylinder
shear
point
interaction
numbers
body
wave
method
leading
region
bodies
supersonic
gas
measurements
approximate
local
equation

continued on next page

Table 5.1.: *continued*

case
constant

5.3.2 Empirical results for synthetic datasets

Our synthetic dataset has been carefully designed in order to highlight a setting where the `MaxComp` heuristic will fail. More specifically, the absolute values of the entries of the largest singular vector of a matrix in this family of matrices is not a good indicator of the importance of the respective entry in a sparse PCA solution. Instead, the vector that emerges from the optimization problem of eqn. (5.1) is a much better indicator. In order to generate our synthetic dataset, we generated the following matrix:

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\intercal} + \mathbf{E}_{\sigma},$$

where $\mathbf{U} \in \mathbb{R}^{m\times m}$ and $\mathbf{V} \in \mathbb{R}^{n\times n}$ are orthonormal. The matrix $\mathbf{\Sigma} \in \mathbb{R}^{m\times n}$ has m distinct singular values σ_i in its diagonal and the matrix $\mathbf{E}_{\sigma} \in \mathbb{R}^{m\times n}$ is a noise matrix parameterized by $\sigma > 0$.

We set $\mathbf{U}$ to be a Hadamard matrix with normalized columns; we set $\mathbf{\Sigma}$ to have entries $\sigma_1 = 100$ and $\sigma_i = 1/e^i$ for all $i = 2, \ldots, m$. The entries of the matrix $\mathbf{E}_{\sigma}$ follow a zero-mean normal distribution with standard deviation $\sigma = 10^{-3}$. We now describe how to construct the matrix $\mathbf{V}$: we set $\mathbf{V} = \mathbf{G}_d(\theta)\tilde{\mathbf{V}}$, where $\tilde{\mathbf{V}}$ is also a Hadamard matrix with normalized columns. Here $\mathbf{G}_n(\theta)$ is a composition of Givens rotations. In particular, $\mathbf{G}_n(\theta)$ is a composition of $n/4$ Givens rotations with the same angle θ for every rotation. More precisely, let $\mathbf{G}(i,j,\theta) \in \mathbb{R}^{n\times n}$ be a Givens rotation matrix, which rotates the plane i-j by an angle θ:

$$\mathbf{G}(i,j,\theta) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c_1 & \cdots & -c_2 & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & c_2 & \cdots & c_1 & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix},$$

where $i,j \in \{1,\dots,n\}$, $c_1 = \cos\theta$ and $c_2 = \sin\theta$. We define the composition as follows:

$$\mathbf{G}_n(\theta) = \mathbf{G}(i_1,j_1,\theta)\mathbf{G}(i_2,j_2,\theta)\cdots\cdots,\mathbf{G}(i_k,j_k,\theta),\cdots,\mathbf{G}(i_{n/4},j_{n/4},\theta)$$

with

$$i_k = \frac{n}{2} + 2k - 1, \quad j_k = \frac{n}{2} + 2k \quad \text{for} \quad k = 1,\dots,\frac{n}{4}.$$

The matrix $\mathbf{G}_n(\theta)$ rotates (in a pairwise manner) the bottom $n/2$ components of the columns of $\tilde{\mathbf{V}}$. Since the Hadamard matrix has entries equal to +1 or -1 (up to normalization), we will pick a value of θ that guarantees that, after rotation, $n/4$ components of the columns of $\tilde{\mathbf{V}}$ will be almost zero. Thus, the resulting matrices will have about a quarter of components set at a large value, a quarter of their components set at roughly zero, and the rest set at a moderate value. For example, let $n = 2^{12}$ and $\theta \approx 0.27\pi$; then, the difference between the first column of matrix $\mathbf{V}$ and $\tilde{\mathbf{V}}$ is presented in Figure 5.4, where we plotted the (sorted) absolute values of the components of the first column of the matrices $\mathbf{V}$ and $\tilde{\mathbf{V}}$.

First we compare the performance of the different methods on the synthetic dataset, using the data generator described earlier with $m = 2^7$, $n = 2^{12}$, and $\theta \approx 0.27\pi$. In Figure 5.5 we plot (in the y-axis) the value of the performance ra-

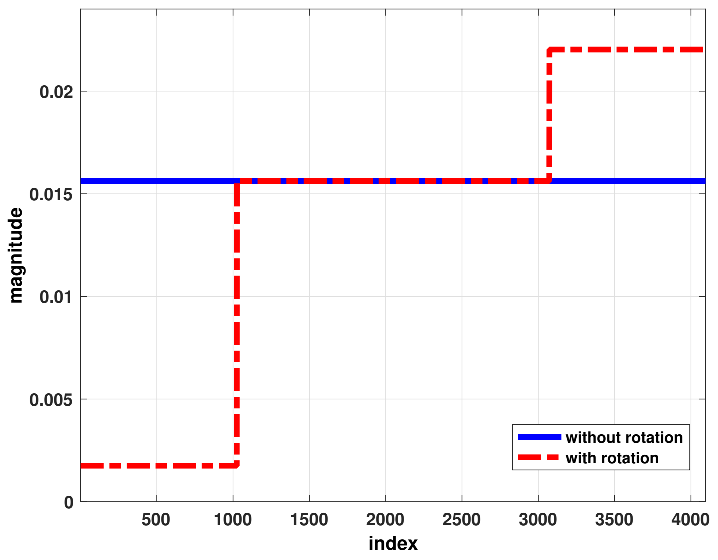


Figure 5.4.: The red dashed line corresponds to the sorted absolute values of the components of the first column of matrix $\mathbf{V}$. Similarly, the blue line corresponds to the first column of $\tilde{\mathbf{V}}$.

tio $f(\mathbf{x})$ (as defined in eqn. (5.17)) for our method (`rspca`), the `Spasm` software, and the `MaxComp` heuristic. We also note that for each of the three methods, we use two different approaches to normalize the resulting sparse vector: the *naive* and the *SVD-based* ones. As a result, we have a total of six possible methods to create and normalize a sparse vector for sparse PCA. The x-axis shows the sparsity ratio of the resulting vector, namely $\|\hat{\mathbf{x}}_{opt}\|_0/n$. We remark that all six methods produce sparse vectors with exactly the same sparsity in order to have a fair comparison. Notice that in Figure 5.5, the `MaxComp` heuristic has worse performance when compared to either our approach or to `Spasm`: this is expected, since we constructed this family of matrices in order to precisely guarantee that the largest components of the top singular vector would not be good elements to retain in the construction of a sparse vector. To further visualize this, we look at the sparse vectors, returned by the different methods, in Figure 5.6. In this figure, we present the resulting sparse vectors for each of the three methods (normalization, obviously, is not relevant in Figure 5.6) for a particular choice of the sparsity ratio ($\|\hat{\mathbf{x}}_{opt}\|_0/n \approx 0.1$). Notice that `MaxComp` fails to capture the right sparsity pattern, whereas our method and `Spasm` succeed.

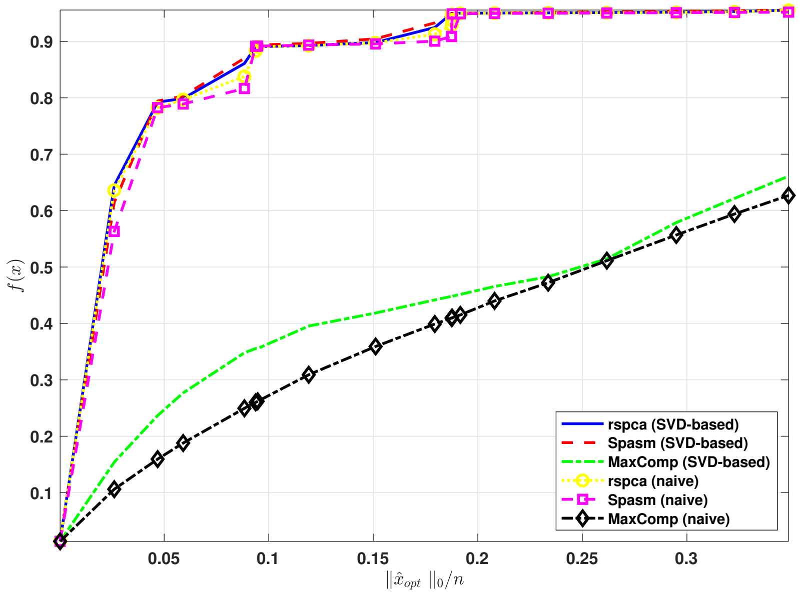
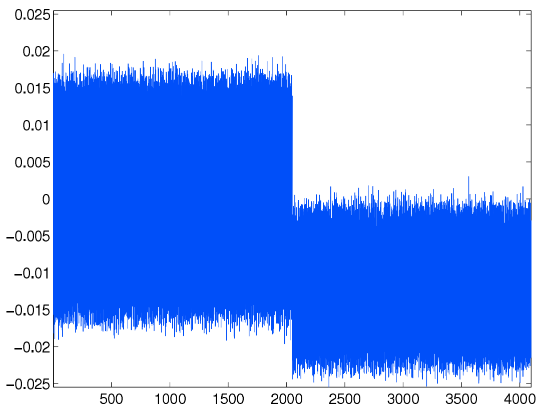
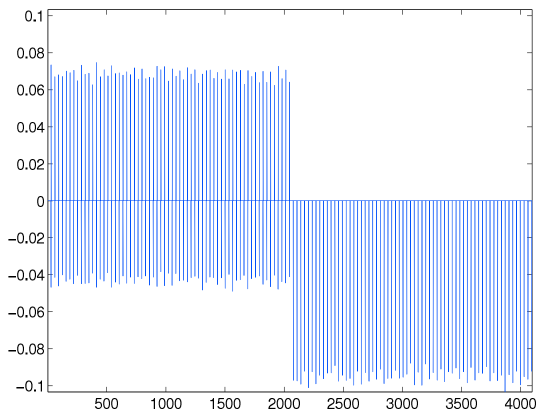


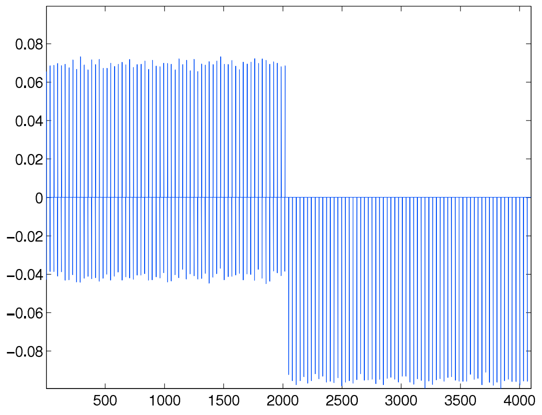
Figure 5.5.: $f(\mathbf{x})$ vs. sparsity ratio $\|\hat{\mathbf{x}}_{opt}\|_0/n$ for the synthetic dataset.



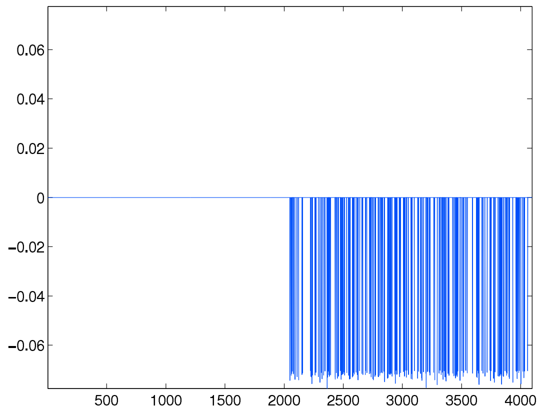
(a) Actual eigenvector



(b) `rspca`



(c) `Spasm`



(d) `MaxComp`

Figure 5.6.: Sparse PCA solution vectors for the synthetic dataset.

6 STRUCTURAL CONVERGENCE RESULTS FOR APPROXIMATION OF DOMINANT SUBSPACES FROM BLOCK KRYLOV SPASES

We present and prove our bounds for the distance between the Krylov space and the dominant left singular space and the distance between the dominant left singular space and a particular dominant subspace approximation from the aforementioned Krylov space

This chapter is organized as follows: In Section 6.1 we provide the setting under which we will work throughout the chapter. We also provide the mathematical background needed for building our bounds. In Section 6.2 we state and prove our bound for the angle between the block Krylov subspace and the dominant left singular space. In section 6.3 we state and prove our bound for the distance between the dominant left singular space and a particular dominant subspace approximation from the block Krylov subspace. Finally, in Section 6.6 we present TeraPCA a software package that computes principal components of terascale genetic data using the randomized subspace iteration.

Sections of chapter 6 have been published in [DIKMI18]

Structural Convergence Results for Approximation of Dominant Subspaces from Block Krylov Spaces P. Drineas, I. Ipsen, E-M. Kontopoulou, M. Magdon-Ismail, in SIAM Journal on Matrix Analysis and Applications (2018), Vol. 39(2), pp. 567-586

6.1 Setting

The approximation of the dominant left singular vector subspace of a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, given a starting guess $\mathbf{X} \in \mathbb{R}^{n\times s}$, is done by constructing the Krylov space in $\mathbf{A}\mathbf{A}^\top$ and $\mathbf{A}\mathbf{X}$,

$$\mathcal{K}_q \equiv \mathcal{K}_q(\mathbf{A}\mathbf{A}^\top, \mathbf{A}\mathbf{X}) = \text{range}\begin{pmatrix}\mathbf{A}\mathbf{X} & (\mathbf{A}\mathbf{A}^\top)\mathbf{A}\mathbf{X} & \cdots & (\mathbf{A}\mathbf{A}^\top)^{\mathrm{q}}\mathbf{A}\mathbf{X}\end{pmatrix}. \tag{6.1}$$

The assumption is that the Krylov space $\mathcal{K}_q$ has maximal dimension, $\dim(\mathcal{K}_q) = (q+1)s$.

Let $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ be the *full* SVD of $\mathbf{A}$, so that $\mathbf{\Sigma} \in \mathbb{R}^{m\times n}$, and $\mathbf{U} \in \mathbb{R}^{m\times m}$ and $\mathbf{V} \in \mathbb{R}^{n\times n}$ are orthogonal matrices. For a positive integer $1 \le k < \text{rank}(\mathbf{A})$, the dominant subspaces are:

$$\mathbf{\Sigma} = \begin{pmatrix}\mathbf{\Sigma}_k & \\ & \mathbf{\Sigma}_{k,\perp}\end{pmatrix}, \qquad \mathbf{U} = \begin{pmatrix}\mathbf{U}_k & \mathbf{U}_{k,\perp}\end{pmatrix}, \qquad \mathbf{V} = \begin{pmatrix}\mathbf{V}_k & \mathbf{V}_{k,\perp}\end{pmatrix},$$

where the diagonal matrix $\mathbf{\Sigma}_k$ contains the k largest non-zero singular values, hence is nonsingular and the rest $\min\{m,n\} - k$ singular values reside on the diagonal of $\mathbf{\Sigma}_{k,\perp}$. $\mathbf{U}_k$ and $\mathbf{V}_k$ are the orthonormal matrices of the top-k left and right singular vectors of $\mathbf{A}$, respectively. $\mathbf{U}_{k,\perp}$ and $\mathbf{V}_{k,\perp}$ are the orthonormal matrices of the rest $m-k$ left and $n-k$ right singular vectors of $\mathbf{A}$ respectively. The dominant subspaces is well-defined when the dominant k singular values of $\mathbf{A}$ are strictly larger than the remaining ones, $1/\|\mathbf{\Sigma}_k^{-1}\|_2 > \|\mathbf{\Sigma}_{k,\perp}\|_2 > 0$.

Optimal least squares References to optimal solutions for multiple right-hand side least squares problems in the two norm are hard to find. Lemma 20 below is easy to prove for the Frobenius norm. An elaborate proof for Schatten-p norms can be found in [Mah92, Theorems 3.2 and 3.3]. For the sake of completeness, we present a straightforward proof for the two norm.

Lemma 20 (Optimal Least Squares). *Let $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{B} \in \mathbb{R}^{m\times p}$. Then $\mathbf{A}^\dagger\mathbf{B}$ is a solution of*

$$\min_{\mathbf{X}\in\mathbb{R}^{n\times p}} \|\mathbf{B}-\mathbf{A}\mathbf{X}\|_2$$

with least squares residual $\|(\mathbf{I}-\mathbf{A}\mathbf{A}^\dagger)\mathbf{B}\|_2 = \min_{\mathbf{X}\in\mathbb{R}^{n\times p}} \|\mathbf{B}-\mathbf{A}\mathbf{X}\|_2$.

Proof. Let $\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$ be a thin SVD, and let $\begin{pmatrix}\mathbf{U} & \mathbf{U}_\perp\end{pmatrix} \in \mathbb{R}^{m\times m}$ be an orthogonal matrix. Any $\mathbf{X} \in \mathbb{R}^{n\times p}$ satisfies

$$\begin{aligned}
\|\mathbf{B}-\mathbf{A}\mathbf{X}\|_2^2 &= \|\mathbf{U}\mathbf{U}^\top(\mathbf{B}-\mathbf{A}\mathbf{X}) + \mathbf{U}_\perp\mathbf{U}_\perp^\top(\mathbf{B}-\mathbf{A}\mathbf{X})\|_2^2 \\
&= \|\mathbf{U}(\mathbf{U}^\top\mathbf{B}-\boldsymbol{\Sigma}\mathbf{V}^\top\mathbf{X}) + \mathbf{U}_\perp\mathbf{U}_\perp^\top\mathbf{B}\|_2^2 \\
&= \|\mathbf{U}\mathbf{T}_1 + \mathbf{U}_\perp\mathbf{T}_2\|_2^2,
\end{aligned}$$

where $\mathbf{T}_1 \equiv \mathbf{U}^\top\mathbf{B}-\boldsymbol{\Sigma}\mathbf{V}^\top\mathbf{X}$ and $\mathbf{T}_2 \equiv \mathbf{U}_\perp^\top\mathbf{B}$. Let $\mathbf{y}_{opt} \in \mathbb{R}^p$ with $\|\mathbf{y}_{opt}\|_2 = 1$ satisfy $\|\mathbf{U}_\perp\mathbf{T}_2\mathbf{y}_{opt}\|_2^2 = \|\mathbf{U}_\perp\mathbf{T}_2\|_2^2$. The vector Pythagoras theorem implies

$$\begin{aligned}
\|\mathbf{B}-\mathbf{A}\mathbf{X}\|_2^2 &\geq \|(\mathbf{U}\mathbf{T}_1 + \mathbf{U}_\perp\mathbf{T}_2)\mathbf{y}_{opt}\|_2^2 \\
&= \|\mathbf{U}\mathbf{T}_1\mathbf{y}_{opt}\|_2^2 + \|\mathbf{U}_\perp\mathbf{T}_2\mathbf{y}_{opt}\|_2^2 \geq \|\mathbf{U}_\perp\mathbf{T}_2\mathbf{y}_{opt}\|_2^2.
\end{aligned}$$

Combining all of the above gives

$$\|\mathbf{B}-\mathbf{A}\mathbf{X}\|_2^2 \geq \|(\mathbf{U}\mathbf{T}_1 + \mathbf{U}_\perp\mathbf{T}_2)\mathbf{y}_{opt}\|_2^2 \geq \|\mathbf{U}_\perp\mathbf{T}_2\|_2^2 = \|\mathbf{U}_\perp\mathbf{U}_\perp^\top\mathbf{B}\|_2^2.$$

This lower bound is achieved by $\mathbf{X}_{opt} = \mathbf{A}^\dagger\mathbf{B}$,

$$\|\mathbf{B}-\mathbf{A}\mathbf{X}_{opt}\|_2^2 = \|\mathbf{B}-\mathbf{A}\mathbf{A}^\dagger\mathbf{B}\|_2^2 = \|(\mathbf{I}-\mathbf{U}\mathbf{U}^\top)\mathbf{B}\|_2^2 = \|\mathbf{U}_\perp\mathbf{U}_\perp^\top\mathbf{B}\|_2^2.$$

□

The polynomial ϕ The elements of the Krylov space $\mathcal{K}_q$ in eqn. (6.1) can be expressed in terms of matrices of the form $\hat{\phi}(\mathbf{A}\mathbf{A}^\top)\mathbf{A}\mathbf{X} \in \mathbb{R}^{m\times s}$, where $\hat{\phi}$ is a polynomial of degree q. However, observe that we need a higher degree polynomial when the point of interest are the singular values of $\mathbf{A}$,

$$\hat{\phi}(\mathbf{A}\mathbf{A}^\top)\,\mathbf{A}\mathbf{X} = \mathbf{U}\,\hat{\phi}(\mathbf{\Sigma}\mathbf{\Sigma}^\top)\mathbf{\Sigma}\,\mathbf{V}^\top\mathbf{X} = \mathbf{U}\,\phi(\mathbf{\Sigma})\,\mathbf{V}^\top\mathbf{X}. \tag{6.2}$$

In the setting of eqn. (6.2) ϕ is a polynomial of degree $2q+1$ with odd powers only, and represents a *generalized matrix function* (see, [ABF16, HBI73]). Since

$$\mathbf{\Sigma} = \operatorname{diag}\begin{pmatrix}\sigma_1 & \cdots & \sigma_{\min\{m,n\}}\end{pmatrix} \in \mathbb{R}^{m\times n}$$

is rectangular, the polynomial ϕ is applied to the diagonal elements of $\mathbf{\Sigma}$ only, and returns a diagonal matrix of the same dimension,

$$\phi(\mathbf{\Sigma}) \equiv \operatorname{diag}\begin{pmatrix}\phi(\sigma_1) & \cdots & \phi(\sigma_{\min\{m,n\}})\end{pmatrix} \in \mathbb{R}^{m\times n}.$$

Now, we can define the elements in $\mathcal{K}_q$ by

$$\mathbf{\Phi} \equiv \mathbf{U}\phi(\mathbf{\Sigma})\mathbf{V}^\top\mathbf{X} \;\in \mathbb{R}^{m\times s}. \tag{6.3}$$

Clearly,

$$\operatorname{range}(\Phi) \subset \mathcal{K}_{\mathrm{q}}. \tag{6.4}$$

The assumption $\dim(\mathcal{K}_q) = (q+1)s \leq m$ in Step 1 of Algorithm 19 guarantees that $\mathbf{U}_K$ has exactly $(q+1)s$ orthonormal columns, which form a basis for $\mathcal{K}_q$.

Angles between subspaces Let $\mathbf{Q} \in \mathbb{R}^{n\times s}$ and $\mathbf{W}_k \in \mathbb{R}^{n\times k}$, with $k \leq s$, be matrices with orthonormal columns. Hence, the singular values $\sigma_j(\mathbf{W}_k^\top\mathbf{Q})$ lie between zero and one, and we can write $\sigma_j(\mathbf{W}_k^\top\mathbf{Q}) = \cos(\theta_j)$, $1 \leq j \leq k$. The *principal* or *canonical angles* between $\operatorname{range}(\mathbf{Q})$ and $\operatorname{range}(\mathbf{W}_{\mathrm{k}})$ are [GV13, Section 6.4.3]

$$0 \leq \theta_1 \leq \cdots \leq \theta_k \leq \pi/2,$$

where $\theta_j = \cos^{-1}(\sigma_j(\mathbf{W}_k^\top \mathbf{Q}))$. Following [SS90, Definition I.5.3], we define the diagonal matrix of principal angles between the subspaces spanned by the columns of $\mathbf{Q}$ and the columns of $\mathbf{W}_k$

$$\boldsymbol{\Theta}\left(\mathbf{Q}, \mathbf{W}_k\right) \equiv \operatorname{diag}\left(\theta_1 \quad \cdots \quad \theta_k\right).$$

Hence the singular values of $\mathbf{W}_k^\top \mathbf{Q}$ are the diagonal elements of $\cos\left[\boldsymbol{\Theta}\left(\mathbf{Q}, \mathbf{W}_k\right)\right]$. From [GV13, Section 6.4.3] and [SS90, Section I.5.3] follows that the distance between $\operatorname{range}\left(\mathbf{W}_\mathrm{k}\right)$ and $\operatorname{range}\left(\mathbf{Q}\right)$ in the two and Frobenius norms, respectively, equals

$$\|\sin\left[\boldsymbol{\Theta}\left(\mathbf{Q}, \mathbf{W}_k\right)\right]\|_{2,F} = \|(\mathbf{I} - \mathbf{W}_k \mathbf{W}_k^\top)\mathbf{Q}\|_{2,F}. \tag{6.5}$$

In particular, $\|\sin\left[\boldsymbol{\Theta}\left(\mathbf{Q}, \mathbf{W}_k\right)\right]\|_2 = \sin\left(\theta_k\right)$, so the two norm distance is determined by the largest principal angle. Assume that $\operatorname{range}\left(\mathbf{W}_\mathrm{k}\right)$ and $\operatorname{range}\left(\mathbf{Q}\right)$ are sufficiently close, so that the largest angle $\theta_k < \pi/2$. This is equivalent to $\cos\boldsymbol{\Theta}\left(\mathbf{Q}, \mathbf{W}_k\right)$ being nonsingular, and $\operatorname{rank}\left(\mathbf{W}_\mathrm{k}^\top \mathbf{Q}\right) = \mathrm{k}$. Then [ZK13, Section 3] implies that the tangents of the principal angles satisfy

$$\begin{aligned} \|\tan\left[\boldsymbol{\Theta}\left(\mathbf{Q}, \mathbf{W}_k\right)\right]\|_{2,F} &= \|\sin\left[\boldsymbol{\Theta}\left(\mathbf{Q}, \mathbf{W}_k\right)\right]\left(\cos\left[\boldsymbol{\Theta}\left(\mathbf{Q}, \mathbf{W}_k\right)\right]\right)^\dagger\|_{2,F} \\ &= \|(\mathbf{I} - \mathbf{W}_k \mathbf{W}_k^\top)\mathbf{Q}\left(\mathbf{W}_k^\top \mathbf{Q}\right)^\dagger\|_{2,F}. \end{aligned} \tag{6.6}$$

As above, $\|\tan\left[\boldsymbol{\Theta}\left(\mathbf{Q}, \mathbf{W}_k\right)\right]\|_2 = \tan(\theta_k)$, so the two norm tangent is determined by the largest principal angle. The following lemma will be used in subsequent derivations.

Lemma 21 (Theorem 3.1 in [ZK13])**.** *Let* $\mathbf{Q} \in \mathbb{R}^{n \times s}$ *have orthonormal columns, and let* $\mathbf{W} \equiv \begin{pmatrix}\mathbf{W}_k & \mathbf{W}_{k,\perp}\end{pmatrix} \in \mathbb{R}^{n \times n}$ *be an orthogonal matrix where* $\mathbf{W}_k \in \mathbb{R}^{n \times k}$ *with* $k \leq s$. *If* $\operatorname{rank}\left(\mathbf{W}_\mathrm{k}^\top \mathbf{Q}\right) = \mathrm{k}$ *then*

$$\|\tan\left[\boldsymbol{\Theta}(\mathbf{Q}, \mathbf{W}_k)\right]\|_{2,F} = \|(\mathbf{W}_{k,\perp}^\top \mathbf{Q})\left(\mathbf{W}_k^\top \mathbf{Q}\right)^\dagger\|_{2,F}.$$

Proof. From eqn. (6.6) and $\mathbf{I} = \mathbf{W}_k\mathbf{W}_k^\top + \mathbf{W}_{k,\perp}\mathbf{W}_{k,\perp}^\top$ follows

$$\begin{aligned}
\|\tan\left[\mathbf{\Theta}\left(\mathbf{Q}, \mathbf{W}_k\right)\right]\|_{2,F} &= \|(\mathbf{I} - \mathbf{W}_k\mathbf{W}_k^\top)\mathbf{Q}\left(\mathbf{W}_k^\top\mathbf{Q}\right)^\dagger\|_{2,F} \\
&= \|\mathbf{W}_{k,\perp}\mathbf{W}_{k,\perp}^\top\mathbf{Q}\left(\mathbf{W}_k^\top\mathbf{Q}\right)^\dagger\|_{2,F} \\
&= \|\mathbf{W}_{k,\perp}^\top\mathbf{Q}\left(\mathbf{W}_k^\top\mathbf{Q}\right)^\dagger\|_{2,F}. \qquad (6.7)
\end{aligned}$$

Eqn. (6.7) follows from the unitary invariance of the spectral and the Frobenius norms. □

Gap-amplifying Chebyshev polynomials We generalize the Chebyshev-based *gap-amplifying polynomials* in [MM15, Section 4.4], [WZZ15, Section 2.2]. Given an integer $q \geq 1$, define the polynomial

$$\psi_{q'}(x) = \psi_{2q+1}(x) = \sum_{j=0}^{q} a_{2j+1}x^{2j+1}$$

of degree $q' = 2q+1$ with only odd powers of x. The polynomial ψ is *gap-amplifying* if it satisfies three properties:

1. Small input values remain small,

$$\psi_{q'}(1) = 1, \qquad \text{and} \qquad |\psi_{q'}(x)| \leq 1 \quad \text{for } x \in [0,1].$$

2. Large input values are amplified,

$$\psi_{q'}(x) \geq x\,c\,2^{q'\,r(x)} \quad \text{for } x \geq 1,$$

   where the constant c and the function $r(x)$ are parameters of ψ.

3. Super linear growth for large input values,

$$\frac{\psi_{q'}(x)}{x} \geq \frac{\psi_{q'}(y)}{y} \quad \text{for } x \geq y \geq 1.$$

The simplest gap-amplifying polynomial is a Chebyshev polynomial.

Lemma 22 ([MM15, Lemma 5]). *The Chebyshev polynomial $T_{q'}(x)$ of the first kind contains only odd powers of x and is gap-amplifying with $c = 1/4$ and $r(x) = \min\{\sqrt{x-1}, 1\}$.*

Proof. We give a quick sketch of the proof of [MM15, Lemma 5]). Clearly, the Chebyshev polynomial $T_{q'}(x)$ satisfies Property 1. To prove Property 3, it suffices to show

$$T'_{q'}(z) \geq \frac{T_{q'}(y)}{y} \quad \text{for some } z \geq y \geq 1, \tag{6.8}$$

because the mean value theorem implies there exists a $z \in [y, x]$ with

$$\begin{aligned} T_{q'}(x) &= T_{q'}(y + x - y) = T_{q'}(y) + T'_{q'}(z)(x-y) \\ &\geq T_{q'}(y) + \frac{T_{q'}(y)}{y}(x-y) = x\frac{T_{q'}(y)}{y}. \end{aligned}$$

Our proof of Property 2 corrects a small typo in a similar proof in [MM15]. Although a bound equivalent to Property 2 is claimed in [MM15, Lemma 5] it is only proved that $T_{q'}(x) \geq c\, 2^{q'\, r(x)}$. However, only a slight modification is required for the stronger result. The proof of [MM15, Lemma 5] shows that

$$T_{q'}(x) \geq \tfrac{1}{2}\, 2^{q'\sqrt{x-1}} \quad \text{for } 1 \leq x \leq 2.$$

To derive a lower bound on $T_{q'}(x)/x$ for $x \geq 1$, first consider $1 \leq x \leq 2$, where

$$\frac{T_{q'}(x)}{x} \geq \frac{T_{q'}(x)}{2} \geq \tfrac{1}{4}\, 2^{q'\sqrt{x-1}}.$$

For the remaining case $x > 2$, Property 3 implies

$$\frac{T_{q'}(x)}{x} \geq \frac{T_{q'}(2)}{2} \geq \tfrac{1}{4}\, 2^{q'}.$$

Hence $T_{q'}(x) \geq \frac{x}{4}\, 2^{q' \min\{\sqrt{x-1}, 1\}}$. $\square$

For our analysis, we use a rescaled version of the gap-amplifying polynomial $T_{q'}(x)$, which has similar properties to the original.

Lemma 23. *Let*

$$\phi(x) = \frac{(1+\gamma)\,\alpha}{\psi_{q'}\,(1+\gamma)}\psi_{q'}(x/\alpha) \tag{6.9}$$

be the rescaled gap-amplifying polynomial. Then

$$|\phi(x)| \leq \frac{4\alpha}{2^{q' \min\left\{\sqrt{\gamma},1\right\}}} \qquad \textit{for } 0 \leq x \leq \alpha.$$

Proof. The proof is immediate since $|\psi_{q'}(x/\alpha)| \leq 1$ for $0 \leq x \leq \alpha$, and Property 2 implies

$$\psi_{q'}\,(1+\gamma) \geq \frac{(1+\gamma)}{4}\,2^{q' \min\left\{\sqrt{\gamma},1\right\}}.$$

□

Lemma 24. *The rescaled gap-amplifying polynomial $\phi(x)$ in eqn.* (6.9) *satisfies*

$$\phi(x) \geq x \qquad \textit{for } x \geq (1+\gamma)\alpha.$$

Proof. Property 3 implies

$$\frac{\psi_{q'}(x/\alpha)}{x/\alpha} \geq \frac{\psi_{q'}(1+\gamma)}{1+\gamma} \qquad \text{for } x \geq (1+\gamma)\alpha.$$

If we rearrange terms and apply the definition of $\phi(x)$ in eqn. (6.9) then we get Lemma 24. □

6.2 Krylov space angles

We present bounds for the distance between the Krylov space $\mathcal{K}_q$ and the dominant left singular vector space range $(\mathbf{U}_\mathrm{k})$. Theorem 13 bounds the distance between $\mathcal{K}_q$ and the whole space, while Theorem 14 bounds the distance between $\mathcal{K}_q$ and an individual left singular vector. The distances are represented by principal angles and they are bounded by generalized matrix functions in the sense of [ABF16, HBI73].

6.2.1 A bound for the whole space

Theorem 13 below is in the spirit of Rayleigh-Ritz bounds [BD09, Drm96]. It indicates how well the Krylov space $\mathcal{K}_q$ captures the targeted dominant left singular vector space $\text{range}\,(\mathbf{U}_\text{k})$ in both the two norm and the Frobenius norms. Denote by $\mathbf{\Theta}(\mathcal{K}_q, \mathbf{U}_k) \in \mathbb{R}^{k\times k}$ the diagonal matrix of principal angles between $\mathcal{K}_q$ and $\text{range}\,(\mathbf{U}_\text{k})$, and by $\mathbf{\Theta}(\mathbf{X}, \mathbf{V}_k) \in \mathbb{R}^{k\times k}$ the diagonal matrix of the principal angles between $\text{range}\,(\mathbf{X})$ and $\text{range}\,(\mathbf{V}_\text{k})$.

Theorem 13. *Let $\phi(x)$ be a polynomial of degree $2q+1$ with odd powers only, such that $\phi(\mathbf{\Sigma}_k)$ is nonsingular. If* $\text{rank}\left(\mathbf{V}_\text{k}^\top \mathbf{X}\right) = \text{k}$*, then*

$$\|\sin\left[\mathbf{\Theta}(\mathcal{K}_q, \mathbf{U}_k)\right]\|_{2,F} \quad \leq \quad \|\phi(\mathbf{\Sigma}_{k,\perp})\|_2\, \|\phi(\mathbf{\Sigma}_k)^{-1}\|_2\, \|\mathbf{V}_{k,\perp}^\top \mathbf{X}(\mathbf{V}_k^\top \mathbf{X})^\dagger\|_{2,F}.$$

If, in addition, $\mathbf{X}$ has orthornomal or linearly independent columns, then

$$\|\mathbf{V}_{k,\perp}^\top \mathbf{X}(\mathbf{V}_k^\top \mathbf{X})^\dagger\|_{2,F} = \|\tan\left[\mathbf{\Theta}(\mathbf{X}, \mathbf{V}_k)\right]\|_{2,F}$$

and

$$\|\sin\left[\mathbf{\Theta}(\mathcal{K}_q, \mathbf{U}_k)\right]\|_{2,F} \quad \leq \quad \|\phi(\mathbf{\Sigma}_{k,\perp})\|_2\, \|\phi(\mathbf{\Sigma}_k)^{-1}\|_2\, \|\tan\left[\mathbf{\Theta}(\mathbf{X}, \mathbf{V}_k)\right]\|_{2,F}.$$

Proof. We focus on the case where $\mathbf{X}$ is a general matrix, or has orthonormal columns. The first and most critical step of the proof makes a connection between principal angles and least-squares residuals.

Let $\mathcal{P}_q$ be the orthogonal projector onto the Krylov space $\mathcal{K}_q$. For $\mathbf{\Phi}$ in eqn. (6.3) let $\mathbf{\Phi}\mathbf{\Phi}^\dagger$ be the orthogonal projector onto $\operatorname{range}(\Phi)$, with $\operatorname{range}\left(\Phi\Phi^\dagger\right) \subset \operatorname{range}(\mathcal{P}_\mathrm{q})$ due to eqn. (6.4). Hence, eqn. (6.5) implies

$$\|\sin\left[\mathbf{\Theta}(\mathcal{K}_q, \mathbf{U}_k)\right]\|_{2,F} \quad = \quad \|(\mathbf{I} - \mathcal{P}_q)\,\mathbf{U}_k\|_{2,F} \leq \|(\mathbf{I} - \mathbf{\Phi}\mathbf{\Phi}^\dagger)\,\mathbf{U}_k\|_{2,F}. \qquad (6.10)$$

Lemma 20 implies that $\|(\mathbf{I} - \mathbf{\Phi}\mathbf{\Phi}^\dagger)\,\mathbf{U}_k\|_{2,F}$ is the residual of the least squares problem

$$\|(\mathbf{I} - \mathbf{\Phi}\mathbf{\Phi}^\dagger)\,\mathbf{U}_k\|_{2,F} = \min_{\mathbf{\Psi}} \|\mathbf{U}_k - \mathbf{\Phi}\mathbf{\Psi}\|_{2,F} = \|\mathbf{U}_k - \mathbf{\Phi}\mathbf{\Psi}_{opt}\|_{2,F},$$

where $\mathbf{\Psi}_{opt} = \mathbf{\Phi}^\dagger\mathbf{U}_k$ is a least squares solution.

Our next step focuses on the target space. Decompose $\mathbf{\Phi}$ into the target component $\operatorname{range}(\mathbf{U}_\mathrm{k})$ and the complementary subspace, $\mathbf{\Phi} = \mathbf{\Phi}_k + \mathbf{\Phi}_{k,\perp}$, where

$$\mathbf{\Phi}_k \equiv \mathbf{U}_k\phi(\mathbf{\Sigma}_k)\mathbf{V}_k^\top\mathbf{X}, \qquad \mathbf{\Phi}_{k,\perp} \equiv \mathbf{U}_{k,\perp}\phi(\mathbf{\Sigma}_{k,\perp})\mathbf{V}_{k,\perp}^\top\mathbf{X}. \qquad (6.11)$$

From $\operatorname{rank}\left(\mathbf{V}_\mathrm{k}^\top\mathbf{X}\right) = \mathrm{k}$ follows that $(\mathbf{V}_k^\top\mathbf{X})^\dagger$ is a right inverse, $(\mathbf{V}_k^\top\mathbf{X})(\mathbf{V}_k^\top\mathbf{X})^\dagger = \mathbf{I}_k$. With eqn. (2.8) this gives

$$\mathbf{\Phi}_k^\dagger = (\mathbf{V}_k^\top\mathbf{X})^\dagger\phi(\mathbf{\Sigma}_k)^{-1}\mathbf{U}_k^\top \quad \text{and} \quad \mathbf{\Phi}_k\mathbf{\Phi}_k^\dagger = \mathbf{U}_k\mathbf{U}_k^\top, \qquad (6.12)$$

meaning $\mathbf{\Phi}_k\mathbf{\Phi}_k^\dagger$ is the orthogonal projector onto the target space $\operatorname{range}(\mathbf{U}_\mathrm{k})$. The minimality of the least squares residual implies

$$\begin{aligned} \|(\mathbf{I} - \mathbf{\Phi}\mathbf{\Phi}^\dagger)\,\mathbf{U}_k\|_{2,F} &= \|\mathbf{U}_k - \mathbf{\Phi}\,(\mathbf{\Phi}^\dagger\mathbf{U}_k)\|_{2,F} \\ &\leq \|\mathbf{U}_k - \mathbf{\Phi}\,(\mathbf{\Phi}_k^\dagger\mathbf{U}_k)\|_{2,F} = \|(\mathbf{I} - \mathbf{\Phi}\mathbf{\Phi}_k^\dagger)\,\mathbf{U}_k\|_{2,F}. \end{aligned}$$

Now replace the other instance of $\mathbf{\Phi}$ by eqn. (6.11), and use eqn. (6.12) to simplify

$$\begin{aligned}\|(\mathbf{I}-\mathbf{\Phi}\mathbf{\Phi}^\dagger)\,\mathbf{U}_k\|_{2,F} &\leq \|(\mathbf{I}-\mathbf{\Phi}\mathbf{\Phi}_k^\dagger)\,\mathbf{U}_k\|_{2,F} = \|(\mathbf{I}-\mathbf{\Phi}_k\mathbf{\Phi}_k^\dagger)\mathbf{U}_k - \mathbf{\Phi}_{k,\perp}\mathbf{\Phi}_k^\dagger\mathbf{U}_k\|_{2,F}\\ &= \|(\mathbf{I}-\mathbf{U}_k\mathbf{U}_k^T)\mathbf{U}_k - \mathbf{\Phi}_{k,\perp}\mathbf{\Phi}_k^\dagger\mathbf{U}_k\|_{2,F}\\ &= \|\mathbf{\Phi}_{k,\perp}\mathbf{\Phi}_k^\dagger\mathbf{U}_k\|_{2,F}. \end{aligned} \tag{6.13}$$

Combining eqn. (6.10) with eqn. (6.13) gives

$$\|\sin\left[\mathbf{\Theta}(\mathcal{K}_q,\mathbf{U}_k)\right]\|_{2,F} \leq \|\mathbf{\Phi}_{k,\perp}\mathbf{\Phi}_k^\dagger\mathbf{U}_k\|_{2,F}. \tag{6.14}$$

The expressions for $\mathbf{\Phi}_{k,\perp}$ in eqn. (6.11) and $\mathbf{\Phi}_k^\dagger$ in eqn. (6.12), and sub-multiplicativity yield

$$\begin{aligned}\|\mathbf{\Phi}_{k,\perp}\mathbf{\Phi}_k^\dagger\mathbf{U}_k\|_{2,F} &= \|\mathbf{U}_{k,\perp}\,\phi(\mathbf{\Sigma}_{k,\perp})\,\mathbf{V}_{k,\perp}^\top\mathbf{X}(\mathbf{V}_k^\top\mathbf{X})^\dagger\,\phi(\mathbf{\Sigma}_k)^{-1}\mathbf{U}_k^\top\mathbf{U}_k\|_{2,F}\\ &= \|\phi(\mathbf{\Sigma}_{k,\perp})\,\mathbf{V}_{k,\perp}^\top\mathbf{X}(\mathbf{V}_k^\top\mathbf{X})^\dagger\,\phi(\mathbf{\Sigma}_k)^{-1}\|_{2,F}\\ &\leq \|\phi(\mathbf{\Sigma}_{k,\perp})\|_2\,\|\phi(\mathbf{\Sigma}_k)^{-1}\|_2\,\|\mathbf{V}_{k,\perp}^\top\mathbf{X}(\mathbf{V}_k^\top\mathbf{X})^\dagger\|_{2,F}. \end{aligned} \tag{6.15}$$

Combining the inequalities of eqn. (6.14) and eqn. (6.15) gives

$$\|\sin\left[\mathbf{\Theta}(\mathcal{K}_q,\mathbf{U}_k)\right]\|_{2,F} \leq \|\phi(\mathbf{\Sigma}_{k,\perp})\|_2\,\|\phi(\mathbf{\Sigma}_k)^{-1}\|_2\,\|\mathbf{V}_{k,\perp}^\top\mathbf{X}(\mathbf{V}_k^\top\mathbf{X})^\dagger\|_{2,F}.$$

This concludes the proof for general $\mathbf{X}$. The proof for the special case where $\mathbf{X}$ has linearly independent columns follows from Lemma 21. □

Theorem 13 is reminiscent of the eigenvalue bounds [Kny87, (2.18)] which contain a tangent on the left. The term $\|\mathbf{V}_{k,\perp}^\top\mathbf{X}(\mathbf{V}_k^\top\mathbf{X})^\dagger\|_{2,F}$ already appeared in previous analyses of randomized algorithms [DM16,DMM08,DMMS11,MD16], and bounds for it are discussed in Section 6.5. If $\mathbf{X}$ is a random starting guess, such as a random sign matrix, a random Gaussian matrix[1] or a matrix with randomly chosen orthonormal columns, then state-of-the-art matrix concentration inequalities can be called upon.

In the special case where $\mathbf{X}$ has linearly independent columns the bounds admit a geometric interpretation: They depend on the tangents of angles between range$\,(\mathbf{X})$ and the dominant right singular vector space range$\,(\mathbf{V}_{\mathrm{k}})$. The full-rank assumption for $\mathbf{V}_k^\top \mathbf{X}$ means that the spaces range$\,(\mathbf{V}_{\mathrm{k}})$ and range$\,(\mathbf{X})$ are sufficiently close, with all principal angles being less than $\pi/2$.

6.2.2 A bound for one singular vector

Theorem 14 bounds the distances between $\mathcal{K}_q$ and individual left singular vectors of $\mathbf{A}$. To this end, distinguish the k dominant singular values and associated left singular vectors,

$$\mathbf{\Sigma}_k = \operatorname{diag}\left(\sigma_1 \quad \cdots \quad \sigma_k\right), \qquad \mathbf{U}_k = \begin{pmatrix}\mathbf{u}_1 & \cdots & \mathbf{u}_k\end{pmatrix}.$$

Theorem 14. *Let $\phi(x)$ be a polynomial of degree $2q+1$ with odd powers only, such that $\phi(\mathbf{\Sigma}_k)$ is nonsingular. If* $\operatorname{rank}\left(\mathbf{V}_{\mathrm{k}}^\top \mathbf{X}\right) = \mathrm{k}$*, then*

$$|\sin\left[\mathbf{\Theta}(\mathcal{K}_q, \mathbf{u}_i)\right]| \;\leq\; \frac{\|\phi(\mathbf{\Sigma}_{k,\perp})\|_2}{|\phi(\sigma_i)|}\, \|\mathbf{V}_{k,\perp}^\top \mathbf{X}(\mathbf{V}_k^\top \mathbf{X})^\dagger\|_2, \qquad 1 \leq i \leq k.$$

If, in addition, $\mathbf{X}$ has orthonormal columns, then

$$|\sin\left[\mathbf{\Theta}(\mathcal{K}_q, \mathbf{u}_i)\right]| \;\leq\; \frac{\|\phi(\mathbf{\Sigma}_{k,\perp})\|_2}{|\phi(\sigma_i)|}\, \|\tan \mathbf{\Theta}(\mathbf{X}, \mathbf{V}_k)\|_2, \qquad 1 \leq i \leq k.$$

Proof. The proof imitates that of Theorem 13, and simply substitutes the vectors $\mathbf{u}_i$ for the matrix $\mathbf{U}_k$. Note that $(\mathbf{I} - \mathbf{U}_k\mathbf{U}_k^\top)\mathbf{u}_i = 0$ for $1 \leq i \leq k$, which implies

$$|\sin\left[\mathbf{\Theta}(\mathcal{K}_q, \mathbf{u}_i)\right]| \leq \|\mathbf{\Phi}_{k,\perp}\mathbf{\Phi}_k^\dagger \mathbf{u}_i\|_2, \qquad 1 \leq i \leq k.$$

[1]The elements of a random Gaussian matrix are independent identically distributed normal random variables with mean zero and variance one.

The expressions for $\mathbf{\Phi}_{k,\perp}$ in eqn. (6.11) and $\mathbf{\Phi}_k^\dagger$ in eqn. (6.12), and submultiplicativity yield

$$\begin{aligned}
\|\mathbf{\Phi}_{k,\perp}\mathbf{\Phi}_k^\dagger\mathbf{u}_k\|_2 &= \|\mathbf{U}_{k,\perp}\,\phi(\mathbf{\Sigma}_{k,\perp})\,\mathbf{V}_{k,\perp}^\top\mathbf{X}(\mathbf{V}_k^\top\mathbf{X})^\dagger\,\phi(\mathbf{\Sigma}_k)^{-1}\mathbf{U}_k^\top\mathbf{u}_i\|_2 \\
&= \|\phi(\mathbf{\Sigma}_{k,\perp})\,\mathbf{V}_{k,\perp}^\top\mathbf{X}(\mathbf{V}_k^\top\mathbf{X})^\dagger\,\phi(\sigma_i)^{-1}\|_2 \\
&\leq \|\phi(\mathbf{\Sigma}_{k,\perp})\|_2\,|\phi(\sigma_i)^{-1}|\;\|\mathbf{V}_{k,\perp}^\top\mathbf{X}(\mathbf{V}_k^\top\mathbf{X})^\dagger\|_2.
\end{aligned}$$

Combining the previous two sets of inequalities gives

$$|\sin\left[\mathbf{\Theta}(\mathcal{K}_q,\mathbf{u}_i)\right]| \leq \|\phi(\mathbf{\Sigma}_{k,\perp})\|_2\,|\phi(\sigma_i)^{-1}|\;\|\mathbf{V}_{k,\perp}^\top\mathbf{X}(\mathbf{V}_k^\top\mathbf{X})^\dagger\|_2.$$

This concludes the proof of the case for general $\mathbf{X}$. The proof for the special case where $\mathbf{X}$ has orthonormal columns follows from Lemma 21. □

In the special case when $\mathbf{X}$ has orthonormal columns, the angle between a *single* left singular vector and $\mathcal{K}_q$ is bounded by *all* angles between $\mathbf{X}$ and the right singular vector space $\operatorname{range}(\mathbf{V}_\mathrm{k})$.

6.3 Low-rank approximations from a Krylov space

The results here are motivated by work in the Theoretical Computer Science community on Randomized Linear Algebra [DM16]. There, a common objective is the best rank-k approximation to $\mathbf{A}$ with respect to a unitarily invariant norm,

$$\mathbf{A}_k \equiv \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^\top.$$

The particular approximation $\hat{\mathbf{U}}_k$ computed by Proto-Algorithm 19 guarantees a strong optimality property in the projection $\hat{\mathbf{U}}_k\hat{\mathbf{U}}_k^\top\mathbf{A}$: It is the best rank-$k$ approximation to $\mathbf{A}$ from $\mathcal{K}_q$ with respect to the Frobenius norm (see, Lemma 25).

Algorithm 19 Proto-algorithm for a low-rank approximation of $\mathbf{A}$ from $\mathcal{K}_q$

```
Input: $\mathbf{A} \in \mathbb{R}^{m\times n}$, starting guess $\mathbf{X} \in \mathbb{R}^{n\times s}$
1:      Target rank $k < \mathrm{rank}\,(\mathbf{A})$, provided $\sigma_k > \sigma_{k+1}$
2:      Block dimension $q \geq 1$ with $k \leq (q+1)s \leq m$
Output: $\hat{\mathbf{U}}_k \in \mathbb{R}^{m\times k}$ with orthonormal columns
3: Set $\mathbf{K}_q = \begin{pmatrix}\mathbf{AX} & (\mathbf{AA}^\top)\mathbf{AX} & \cdots & (\mathbf{AA}^\top)^q\mathbf{AX}\end{pmatrix} \in \mathbb{R}^{m\times(q+1)s}$,
4: and assume that $\mathrm{rank}\,(\mathbf{K}_\mathrm{q}) = \mathrm{(q+1)s}$.
5: Compute an orthonormal basis $\mathbf{U}_K \in \mathbb{R}^{m\times(q+1)s}$ for $\mathrm{range}\,(\mathbf{K}_\mathrm{q})$.
6: Set $\mathbf{W} \equiv \mathbf{U}_K^\top\mathbf{A} \in \mathbb{R}^{(q+1)s\times n}$, and assume $\mathrm{rank}\,(\mathbf{W}) \geq \mathrm{k}$.
7: Compute an orthonormal basis $\mathbf{U}_{W,k} \in \mathbb{R}^{(q+1)s\times k}$ for the $k$ dominant left singular
   vectors of $\mathbf{W}$.
8: return $\hat{\mathbf{U}}_k = \mathbf{U}_K\mathbf{U}_{W,k} \in \mathbb{R}^{m\times k}$.
```

Theorem 15 presents a quality-of-approximation result for $\hat{\mathbf{U}}_k$. To this end we distinguish the orthonormal columns of $\hat{\mathbf{U}}_k = \begin{pmatrix}\hat{\mathbf{u}}_1 & \ldots & \hat{\mathbf{u}}_k\end{pmatrix} \in \mathbb{R}^{m\times k}$ and set

$$\hat{\mathbf{U}}_i \equiv \begin{pmatrix}\hat{\mathbf{u}}_1 & \ldots & \hat{\mathbf{u}}_i\end{pmatrix} \in \mathbb{R}^{m\times i}, \qquad 1 \leq i \leq k, \tag{6.16}$$

and

$$\Delta \equiv \|\phi(\mathbf{\Sigma}_{k,\perp})\|_2\, \|\mathbf{V}_{k,\perp}^\top\mathbf{X}\,(\mathbf{V}_k^\top\mathbf{X})^\dagger\|_F.$$

Theorem 15. *Let $\phi(x)$ be a polynomial of degree $2q+1$ with odd powers only, such that $\phi(\mathbf{\Sigma}_k)$ is nonsingular, and $\phi(\sigma_i) \geq \sigma_i$ for $1 \leq i \leq k$. If* $\mathrm{rank}\left(\mathbf{V}_\mathrm{k}^\top\mathbf{X}\right) = \mathrm{k}$*, then for $1 \leq i \leq k$,*

$$\|\mathbf{A} - \hat{\mathbf{U}}_i\hat{\mathbf{U}}_i^\top\mathbf{A}\|_F^2 \leq \|\mathbf{A} - \mathbf{A}_i\|_F^2 + \Delta \tag{6.17}$$

$$\|\mathbf{A} - \hat{\mathbf{U}}_i\hat{\mathbf{U}}_i^\top\mathbf{A}\|_2^2 \leq \|\mathbf{A} - \mathbf{A}_i\|_2^2 + \Delta \tag{6.18}$$

$$\sigma_i - \Delta \leq \|\hat{\mathbf{u}}_i^\top\mathbf{A}\|_2^2 \leq \sigma_i. \tag{6.19}$$

If, in addition, $\mathbf{X}$ has orthonormal columns, then

$$\Delta = \|\phi(\mathbf{\Sigma}_{k,\perp})\|_2\, \|\tan\left[\mathbf{\Theta}(\mathbf{X},\mathbf{V}_k)\right]\|_F.$$

Proof. This proof is more involved than the ones of Section 6.1, and requires two auxiliary results, an alternative expression for the error (Section 6.3), and a bound on its Frobenius norm (Section 6.3).

An alternative expression for the error Algorithm 19 approximates the dominant left singular vectors of $\mathbf{A}$ by the orthonormal matrix $\hat{\mathbf{U}}_k \in \mathbb{R}^{m\times k}$. Since bounding $\|\mathbf{A} - \hat{\mathbf{U}}_k\hat{\mathbf{U}}_k^\top\mathbf{A}\|_F$ seems hard, we present an alternative expression that is easier to analyze.

Lemma 25 (Lemma 8 in [BDMI14])**.** *Let $\mathbf{U}_K$ be an orthonormal basis for $\mathcal{K}_q$ and let $\hat{\mathbf{U}}_i$ be as in eqn;* (6.16)*, containing the top i columns of the output of Algorithm 19. Then*

$$\mathbf{A} - \hat{\mathbf{U}}_i\hat{\mathbf{U}}_i^\top\mathbf{A} = \mathbf{A} - \mathbf{U}_K\left(\mathbf{U}_K^\top\mathbf{A}\right)_i, \qquad 1 \le i \le k. \tag{6.20}$$

In addition, $\mathbf{U}_K\left(\mathbf{U}_K^\top\mathbf{A}\right)_i$ is a best rank-i approximation to $\mathbf{A}$ from $\mathcal{K}_q$ in the Frobenius norm,

$$\|\mathbf{A} - \mathbf{U}_K\left(\mathbf{U}_K^\top\mathbf{A}\right)_i\|_F^2 = \min_{\mathrm{rank}(\mathbf{Y})\le \mathrm{i}} \|\mathbf{A} - \mathbf{U}_K\mathbf{Y}\|_F^2, \qquad 1 \le i \le k. \tag{6.21}$$

Proof. Since the transition to best rank-i approximations is a key component, we illustrate how it comes about by proving the first assertion for the case $i = k$.

Algorithm 19 outputs $\hat{\mathbf{U}}_k = \mathbf{U}_K\mathbf{U}_{W,k}$, where $\mathbf{U}_{W,k}$ is the matrix of the dominant k left singular vectors of $\mathbf{W} = \mathbf{U}_K^\top\mathbf{A}$. This means $\mathbf{U}_{W,k}$ spans the same range as $\mathbf{W}_k$, the best rank-k approximation to $\mathbf{W}$. Therefore

$$\begin{aligned} \mathbf{A} - \hat{\mathbf{U}}_k\hat{\mathbf{U}}_k^\top\mathbf{A} &= \mathbf{A} - \mathbf{U}_K\mathbf{U}_{W,k}\mathbf{U}_{W,k}^\top\mathbf{U}_K^\top\mathbf{A} \\ &= \mathbf{A} - \mathbf{U}_K\mathbf{W}_k\mathbf{W}_k^\dagger\mathbf{W} = \mathbf{A} - \mathbf{U}_K\mathbf{W}_k = \mathbf{A} - \mathbf{U}_K\left(\mathbf{U}_K^\top\mathbf{A}\right)_k. \end{aligned}$$

The penultimate equality follows from $\mathbf{W}_k\mathbf{W}_k^\dagger$ being the orthogonal projector onto range$\,(\mathbf{W}_\mathrm{k})$. □

Lemma 25 shows that eqn. (6.17) in Theorem 15 can be proved by bounding $\|\mathbf{A}-\mathbf{U}_K\left(\mathbf{U}_K^\top\mathbf{A}\right)_i\|_F$. Next we transition from the best rank-i approximation of the "projected" matrix $(\mathbf{U}_K^\top\mathbf{A})_i$ to the best rank-i approximation $\mathbf{A}_i$ of the original matrix, by splitting for $1 \le i \le k$,

$$\mathbf{A} = \mathbf{A}_i + \mathbf{A}_{i,\perp} \qquad \text{where} \quad \mathbf{A}_i = \mathbf{U}_i\mathbf{\Sigma}_i\mathbf{V}_i^\top \text{ and } \mathbf{A}_{i,\perp} = \mathbf{U}_{i,\perp}\mathbf{\Sigma}_{i,\perp}\mathbf{V}_{i,\perp}^\top. \tag{6.22}$$

Lemma 26. *Let* $\mathbf{U}_K$ *be an orthonormal basis for* $\mathcal{K}_q$, *and* $\hat{\mathbf{U}}_i$ *in eqn.* (6.16) *the columns of the output of Algorithm 19. Then*

$$\|\mathbf{A}-\hat{\mathbf{U}}_i\hat{\mathbf{U}}_i^\top\mathbf{A}\|_F^2 \le \|\mathbf{A}_i-\mathbf{U}_K\mathbf{U}_K^\top\mathbf{A}_i\|_F^2 + \|\mathbf{A}_{i,\perp}\|_F^2.$$

Proof. The optimality of eqn. (6.21) in Lemma 25 implies

$$\begin{aligned}\|\mathbf{A}-\hat{\mathbf{U}}_i\hat{\mathbf{U}}_i^\top\mathbf{A}\|_F^2 &= \|\mathbf{A}-\mathbf{U}_K\left(\mathbf{U}_K^\top\mathbf{A}\right)_i\|_F^2 \\ &\le \|\mathbf{A}-\mathbf{U}_K\mathbf{U}_K^\top\mathbf{A}_i\|_F^2 \\ &= \|\mathbf{A}_i-\mathbf{U}_K\mathbf{U}_K^\top\mathbf{A}_i\|_F^2 + \|\mathbf{A}_{i,\perp}\|_F^2.\end{aligned}$$

The last equality follows from Lemma 1 □

Bounding the important part of the error We bound the term in Lemma 26 over which we have control, namely $\|\mathbf{A}_i-\mathbf{U}_K\mathbf{U}_K^\top\mathbf{A}_i\|_F^2$.

Let $\mathcal{P}_q$ be the orthogonal projector onto $\mathcal{K}_q$. For $\mathbf{\Phi}$ in eqn. (6.3) let $\mathbf{\Phi}\mathbf{\Phi}^\dagger$ be the orthogonal projector onto range(Φ), with range$\left(\Phi\Phi^\dagger\right) \subset$ range$(\mathcal{P}_\mathrm{q})$ due to eqn. (6.4). The leads to the obvious bound

$$\|\mathbf{A}_i-\mathbf{U}_K\mathbf{U}_K^\top\mathbf{A}_i\|_F = \|\mathbf{A}_i-\mathcal{P}_q\mathbf{A}_i\|_F \le \|\mathbf{A}_i-\mathbf{\Phi}\mathbf{\Phi}^\dagger\mathbf{A}_i\|_F, \qquad 1 \le i \le k. \tag{6.23}$$

We don't stop here, though, but go further and pursue a bound in terms of polynomials.

Lemma 27. *Let $\phi(x)$ be a polynomial of degree $2q+1$ with odd powers only that satisfies $\phi(\sigma_j) \geq \sigma_j$ for $1 \leq j \leq k$. Then*

$$\|\mathbf{A}_i - \mathbf{U}_K\mathbf{U}_K^\top\mathbf{A}_i\|_F \leq \|\mathbf{U}_i\phi\left(\mathbf{\Sigma}_i\right) - \mathbf{\Phi}\mathbf{\Phi}^\dagger\mathbf{U}_i\phi\left(\mathbf{\Sigma}_i\right)\|_F, \qquad 1 \leq i \leq k.$$

Proof. We use the abbreviation $\mathcal{P}_\phi^\perp \equiv \mathbf{I} - \mathbf{\Phi}\mathbf{\Phi}^\dagger$, to denote the orthogonal projector onto $\operatorname{range}(\Phi)^\perp$. From eqn. (6.22), eqn. (6.23) and the unitary invariance of the Frobenius norm follows

$$\|\mathbf{A}_i - \mathbf{U}_K\mathbf{U}_K^\top\mathbf{A}_i\|_F \;\leq\; \|\mathcal{P}_\phi^\perp\mathbf{A}_i\|_F = \|\mathcal{P}_q^\perp\,\mathbf{U}_i\mathbf{\Sigma}_i\|_F, \qquad 1 \leq i \leq k.$$

Expressing the squared Frobenius norm as a sum of squared column norms, and then applying the assumption $\sigma_j \leq \phi(\sigma_j)$ yields for $1 \leq i \leq k$,

$$\|\mathcal{P}_\phi^\perp\mathbf{U}_i\mathbf{\Sigma}_i\|_F^2 \;=\; \sum_{j=1}^{i}\sigma_j^2\|\mathcal{P}_\phi^\perp\mathbf{u}_j\|_2^2 \leq \sum_{j=1}^{i}\phi(\sigma_j)^2\|\mathcal{P}_\phi^\perp\mathbf{u}_j\|_2^2 = \|\mathcal{P}_\phi^\perp\mathbf{U}_i\phi(\mathbf{\Sigma}_i)\|_F^2.$$

□

From projections to least-squares residuals Now we are ready to apply the approach from Theorem 13 and view the result of Lemma 27 as a least squares residual.

Lemma 28. *Under the assumptions of Theorem 15,*

$$\|\mathbf{U}_i\phi\left(\mathbf{\Sigma}_i\right) - \mathbf{\Phi}\mathbf{\Phi}^\dagger\mathbf{U}_i\phi\left(\mathbf{\Sigma}_i\right)\|_F \leq \|\phi(\mathbf{\Sigma}_{k,\perp})\|_2 \;\|\mathbf{V}_{k,\perp}^\top\mathbf{X}(\mathbf{V}_k^\top\mathbf{X})^\dagger\|_F.$$

Proof. Based on the orthogonality $(\mathbf{I} - \mathbf{U}_k\mathbf{U}_k^\top)\mathbf{U}_i = \mathbf{0}$ for $1 \leq i \leq k$, we can deduce as in eqn. (6.13) that

$$\|\left(\mathbf{I} - \mathbf{\Phi}\mathbf{\Phi}^\dagger\right)\mathbf{U}_i\phi(\mathbf{\Sigma}_i)\|_F \leq \|\mathbf{\Phi}_{k,\perp}\mathbf{\Phi}_k^\dagger\,\mathbf{U}_i\phi(\mathbf{\Sigma}_i)\|_F.$$

The expressions for $\mathbf{\Phi}_{k,\perp}$ in eqn. (6.11) and $\mathbf{\Phi}_k^\dagger$ in eqn. (6.12), along with the strong submultiplicativity yield

$$\begin{aligned}\|\mathbf{\Phi}_{k,\perp}\mathbf{\Phi}_k^\dagger\mathbf{U}_i\phi(\mathbf{\Sigma}_i)\|_F &= \|\mathbf{U}_{k,\perp}\,\phi(\mathbf{\Sigma}_{k,\perp})\,\mathbf{V}_{k,\perp}^\top\mathbf{X}(\mathbf{V}_k^\top\mathbf{X})^\dagger\,\phi(\mathbf{\Sigma}_k)^{-1}\mathbf{U}_k^\top\mathbf{U}_i\phi(\mathbf{\Sigma}_i)\|_F \\ &\le \|\phi(\mathbf{\Sigma}_{k,\perp})\|_2\ \|\mathbf{V}_{k,\perp}^\top\mathbf{X}(\mathbf{V}_k^\top\mathbf{X})^\dagger\|_F.\end{aligned}$$

The above inequality is obtained by noting that for $i = k$ we have $\phi(\mathbf{\Sigma}_k)^{-1}\mathbf{U}_k^\top\mathbf{U}_i\phi\left(\mathbf{\Sigma}_i\right) = \mathbf{I}_k$, while for $1 \le i < k$,

$$\phi(\mathbf{\Sigma}_k)^{-1}\mathbf{U}_k^\top\mathbf{U}_i\phi\left(\mathbf{\Sigma}_i\right) = \begin{pmatrix}\mathbf{I}_i \\ \mathbf{0}_{(k-i)\times i}\end{pmatrix}.$$

□

We continue by proving each of the three inequalities in turn. Recall that

$$\Delta \equiv \|\phi(\mathbf{\Sigma}_{k,\perp})\|_2\ \|\mathbf{V}_{k,\perp}^\top\mathbf{X}\,(\mathbf{V}_k^\top\mathbf{X})^\dagger\|_F.$$

Proof of eqn. (6.17) Combining Lemmas 26, 27, and 28 and recognizing the expression for Δ yields

$$\begin{aligned}\|\mathbf{A} - \hat{\mathbf{U}}_i\hat{\mathbf{U}}_i^\top\|_F^2 &\le \|\mathbf{A}_{i,\perp}\|_F^2 + \|\phi(\mathbf{\Sigma}_{k,\perp})\|_2^2\ \|\mathbf{V}_{k,\perp}^\top\mathbf{X}(\mathbf{V}_k^\top\mathbf{X})^\dagger\|_F^2, \\ &= \|\mathbf{A}_{i,\perp}\|_F^2 + \Delta^2, \qquad 1 \le i \le k.\end{aligned} \tag{6.24}$$

Inserting $\|\mathbf{A}_{i,\perp}\|_F = \|\mathbf{A} - \mathbf{A}_i\|_F$ gives

$$\|\mathbf{A} - \hat{\mathbf{U}}_i\hat{\mathbf{U}}_i^\top\mathbf{A}\|_F^2 \le \|\mathbf{A} - \mathbf{A}_i\|_F^2 + \Delta^2, \qquad 1 \le i \le k. \tag{6.25}$$

Taking advantage of the inequality below for scalars $\alpha, \beta \ge 0$,

$$\sqrt{\alpha^2 + \beta^2} \le \sqrt{\alpha^2 + \beta^2 + 2\alpha\beta} = \sqrt{(\alpha + \beta)^2} = \alpha + \beta, \tag{6.26}$$

gives the weaker, but square-free bound of eqn. (6.17).

Bounds of the form of eqn. (6.19) were already proposed in [MM15, Theorem 1] as a finer, *vector-wise*, way to capture the quality of approximations to individual left singular vectors of $\mathbf{A}$. Empirical evidence [MM15] suggests that error metrics of the form of eqn. (6.17) and eqn. (6.18) indicate the quality of the *aggregate* approximation and are therefore coarser than those of eqn. (6.19).

Proof of eqn. (6.18) We use Lemma 29 below, which shows that an additive error bound for a low-rank approximation in the Frobenius norm implies the same in the two norm, and apply it to eqn. (6.24). This gives $\|\mathbf{A}-\hat{\mathbf{U}}_i\hat{\mathbf{U}}_i^\top\mathbf{A}\|_2^2 \leq \|\mathbf{A}-\mathbf{A}_i\|_2^2+\Delta^2$. Taking square roots based on eqn. (6.26) produces the desired bound of eqn. (6.18).

Lemma 29 (Theorem 3.4 in [Gu15])**.** *Given* $\mathbf{A}, \tilde{\mathbf{A}} \in \mathbb{R}^{m\times n}$ *with* $\operatorname{rank}\left(\tilde{\mathbf{A}}\right) = \mathrm{k} < \operatorname{rank}(\mathbf{A})$*. If* $\|\mathbf{A}-\tilde{\mathbf{A}}\|_F^2 \leq \|\mathbf{A}-\mathbf{A}_k\|_F^2+\delta$*, then*

$$\|\mathbf{A}-\tilde{\mathbf{A}}\|_2^2 \leq \|\mathbf{A}-\mathbf{A}_k\|_2^2+\delta.$$

Proof of eqn. (6.19) The upper bounds follow from the *minimax theorem for singular values* [GV13, Theorem 8.6.1].

This leaves the lower bounds. Recall the non-increasing ordering of the singular values $\sigma_1 \geq \cdots \geq \sigma_k$, and the fact that $\hat{\mathbf{U}}_i$ in eqn. (6.16) has orthonormal columns.

Case $i = 1$ Apply Lemma 1 to eqn. (6.24)

$$\|\mathbf{A}\|_F^2 - \|\hat{\mathbf{u}}_1^\top\mathbf{A}\|_F^2 = \|\mathbf{A}-\hat{\mathbf{u}}_1\hat{\mathbf{u}}_1^\top\mathbf{A}\|_F^2 \leq \|\mathbf{A}_{1,\perp}\|_F^2+\Delta^2.$$

From $\|\mathbf{A}\|_F^2 - \|\mathbf{A}_{1,\perp}\|_F^2 = \sigma_1^2$ follows $\sigma_1^2 \leq \|\mathbf{A}-\hat{\mathbf{u}}_1\hat{\mathbf{u}}_1^\top\mathbf{A}\|_F^2+\Delta^2$. Taking square roots based on eqn. (6.26) proves eqn. (6.19) for $i = 1$.

Case $2 \le i \le k$ Among all matrices of rank $i-1$, the matrix $\mathbf{A}_{i-1}$ is closest to $\mathbf{A}$ in the Frobenius norm. Hence

$$\|\mathbf{A}_{i-1,\perp}\|_F = \|\mathbf{A} - \mathbf{A}_{i-1}\|_F \le \|\mathbf{A} - \hat{\mathbf{U}}_{i-1}\hat{\mathbf{U}}_{i-1}^\top \mathbf{A}\|_F.$$

The above, together with the outer product representation $\hat{\mathbf{U}}_i\hat{\mathbf{U}}_i^\top = \hat{\mathbf{U}}_{i-1}\hat{\mathbf{U}}_{i-1}^\top +$ $\hat{\mathbf{u}}_i\hat{\mathbf{u}}_i^\top$, Lemma 1 and eqn. (6.24) gives

$$\begin{aligned} \|\mathbf{A}_{i-1,\perp}\|_F^2 - \|\hat{\mathbf{u}}_i\hat{\mathbf{u}}_i^\top \mathbf{A}\|_F^2 &\le \|\mathbf{A} - \hat{\mathbf{U}}_{i-1}\hat{\mathbf{U}}_{i-1}^\top \mathbf{A}\|_F^2 - \|\hat{\mathbf{u}}_i\hat{\mathbf{u}}_i^\top \mathbf{A}\|_F^2 \\ &= \|\mathbf{A} - \hat{\mathbf{U}}_i\hat{\mathbf{U}}_i^\top \mathbf{A}\|_F^2 \le \|\mathbf{A}_{i,\perp}\|_F^2 + \Delta^2. \end{aligned}$$

At last, applying $\|\mathbf{A}_{i-1,\perp}\|_F^2 - \|\mathbf{A}_{i,\perp}\|_F^2 = \sigma_i^2$, and taking square roots based on eqn. (6.26) proves eqn. (6.19) for $2 \le i \le k$.

This concludes the proof for general $\mathbf{X}$. orthonormal columns follows from Lemma 21.

$\square$

6.4 Judicious choice of polynomials

We show the existence of and present bounds for the polynomials in Theorems 13, 14, and 15. The strict inequality rank$\,(\mathbf{A}) >$ k in Algorithm 19 allows us to express the relative singular gap as

$$\frac{\sigma_k - \sigma_{k+1}}{\sigma_{k+1}} \ge \gamma > 0, \tag{6.27}$$

which is equivalent to $\sigma_k \ge (1+\gamma)\sigma_{k+1} > 0$.

Lemma 30. *If eqn.* (6.27) *holds, then there exists a polynomial* $\phi(x)$ *of degree* $2q+1$ *with odd powers only, such that* $\phi(\sigma_i) \ge \sigma_i > 0$ *for* $1 \le i \le k$*, and*

$$|\phi(\sigma_i)| \le \frac{4\sigma_{k+1}}{2^{(2q+1)\,\min\left\{\sqrt{\gamma},1\right\}}}, \qquad i \ge k+1.$$

Hence

$$\|\phi(\mathbf{\Sigma}_k)^{-1}\|_2 \leq \sigma_k^{-1} \qquad \text{and} \qquad \|\phi(\mathbf{\Sigma}_{k,\perp})\|_2 \leq \frac{4\sigma_{k+1}}{2^{(2q+1)\ \min\left\{\sqrt{\gamma},1\right\}}}.$$

Proof. Let $\phi(x)$ be the rescaled gap-amplifying polynomial in eqn. (6.9) with $\alpha = \sigma_{k+1}$ and γ in eqn. (6.27). The inequalities for $\phi(\sigma_i)$ follow from $q' = 2q+1$, Lemma 23 and Lemma 24.

From $\phi(\sigma_i) > 0$ for $1 \leq i \leq k$ and $\phi(\mathbf{\Sigma}_k)$ being a diagonal matrix follows

$$\|\phi(\mathbf{\Sigma}_k)^{-1}\|_2 = \max_{1\leq i\leq k} \phi(\sigma_i)^{-1} \leq \max_{1\leq i\leq k} {\sigma_i}^{-1} = \sigma_k^{-1}.$$

Furthermore

$$\|\phi(\mathbf{\Sigma}_{k,\perp})\|_2 = \max_{i\geq k+1} |\phi\left(\sigma_i\right)| \leq \frac{4\sigma_{k+1}}{2^{(2q+1)\ \min\left\{\sqrt{\gamma},1\right\}}}.$$

□

We apply Lemma 30 to the previous results, first for the special case when $\mathbf{X}$ has linearly independent columns. Abbreviate

$$\Gamma(\mathbf{\Theta}, \gamma, q) \equiv 4\, \frac{\|\tan\left[\mathbf{\Theta}(\mathbf{X}, \mathbf{V}_k)\right]\|_2}{2^{(2q+1)\ \min\left\{\sqrt{\gamma},1\right\}}}.$$

To keep things short, we consider only the two-norm bound for Theorem 13.

Corollary 1. *Let eqn.* (6.27) *hold and* $\operatorname{rank}\left(\mathbf{V}_{\mathrm{k}}^\top \mathbf{X}\right) = \mathrm{k}$*. If* $\mathbf{X}$ *has orthonormal columns, then*

$$\|\sin\left[\mathbf{\Theta}(\mathcal{K}_q, \mathbf{U}_k)\right]\|_2 \ \ \leq \ \ \Gamma(\mathbf{\Theta}, \gamma, q)\, \frac{\sigma_{k+1}}{\sigma_k} \leq \frac{\Gamma(\mathbf{\Theta}, \gamma, q)}{1+\gamma},$$

and

$$|\sin\left[\mathbf{\Theta}(\mathcal{K}_q, \mathbf{u}_i)\right]| \ \ \leq \ \ \Gamma(\mathbf{\Theta}, \gamma, q)\, \frac{\sigma_{k+1}}{\sigma_i}, \qquad 1 \leq i \leq k.$$

Proof. Apply Lemma 30 to Theorems 13 and 14. □

Corollary 2. *Let eqn.* (6.27) *hold and* $\operatorname{rank}\left(\mathbf{V}_{\mathrm{k}}^\top \mathbf{X}\right) = \mathrm{k}$*. If* $\mathbf{X}$ *has orthonormal columns, then Theorem 15 holds with*

$$\Delta \leq \Gamma(\boldsymbol{\Theta}, \gamma, q)\, \sigma_{k+1},$$

so that for $1 \leq i \leq k$

$$\begin{aligned}
\|\mathbf{A} - \hat{\mathbf{U}}_i \hat{\mathbf{U}}_i^\top \mathbf{A}\|_F^2 &\leq \|\mathbf{A} - \mathbf{A}_i\|_F^2 + \Gamma(\boldsymbol{\Theta}, \gamma, q)\, \sigma_{k+1}, \\
\|\mathbf{A} - \hat{\mathbf{U}}_i \hat{\mathbf{U}}_i^\top \mathbf{A}\|_2^2 &\leq \|\mathbf{A} - \mathbf{A}_i\|_2^2 + \Gamma(\boldsymbol{\Theta}, \gamma, q)\, \sigma_{k+1}, \\
\sigma_i - \Gamma(\boldsymbol{\Theta}, \gamma, q)\, \sigma_{k+1} &\leq \|\hat{\mathbf{u}}_i^\top \mathbf{A}\|_2^2 \leq \sigma_i.
\end{aligned}$$

Proof. Apply Lemma 30 to Theorem 15. □

To achieve an additive error of $\Gamma(\boldsymbol{\Theta}, \gamma, q) \leq \varepsilon$, set q to be the smallest integer that exceeds

$$q \geq \frac{1}{2 \min\left\{\sqrt{\gamma}, 1\right\}} \left(\log_2 4\, \|\tan\left[\boldsymbol{\Theta}(\mathbf{X}, \mathbf{V}_k)\right]\|_2 - \log_2 \varepsilon\right). \tag{6.28}$$

Thus, as the singular value gap γ decreases, the dimension of the space $\mathcal{K}_q$ increases. More specifically, q increases logarithmically with higher target accuracy ε and increasing distance of $\mathbf{X}$ from the dominant right singular vector space of $\mathbf{A}$.

If $\mathbf{X}$ is rank deficient then Corollaries 1 and 2 still hold with

$$\Gamma(\boldsymbol{\Theta}, \gamma, q) = 4\, \frac{\|\mathbf{V}_{k,\perp}^\top \mathbf{X} (\mathbf{V}_k^\top \mathbf{X})^\dagger\|_2}{2^{(2q+1)\, \min\left\{\sqrt{\gamma}, 1\right\}}}.$$

6.5 The initial guess

It remains to bound $\|\mathbf{V}_{k,\perp}^\top \mathbf{X}\, (\mathbf{V}_k^\top \mathbf{X})^\dagger\|_{2,F}$. The simplest way might be strong submultiplicativity,

$$\|\mathbf{V}_{k,\perp}^\top \mathbf{X}\, (\mathbf{V}_k^\top \mathbf{X})^\dagger\|_{2,F} \leq \|\mathbf{V}_{k,\perp}^\top \mathbf{X}\|_{2,F}\, \|(\mathbf{V}_k^\top \mathbf{X})^\dagger\|_2 = \frac{\|\mathbf{V}_{k,\perp}^\top \mathbf{X}\|_{2,F}}{\sigma_k(\mathbf{V}_k^\top \mathbf{X})},$$

followed by separate bounds for the individual factors.

Ideally, the starting guess $\mathbf{X}$ should be close to range $(\mathbf{V}_{\mathrm{k}})$ and far away from range $(\mathbf{V}_{\mathrm{k},\perp})$, so that $\sigma_k(\mathbf{V}_k^\top \mathbf{X})$ is large and $\|\mathbf{V}_{k,\perp}^\top \mathbf{X}\|_{2,F}$ is small. The assumption $\sigma_k(\mathbf{V}_k^\top \mathbf{X}) > 0$ is critical for our results, hence a necessary condition for the user-specified matrix $\mathbf{X} \in \mathbb{R}^{n\times s}$ is rank $(\mathbf{X}) \geq \mathrm{k}$, while trying to keep the column dimension $s \geq k$ small.

If $\mathbf{X}$ is a random Gaussian, then $\sigma_k(\mathbf{V}_k^\top \mathbf{X})$ is bounded away from zero with high probability even for $s = k$. However, there are many other choices for $\mathbf{X}$ that come with lower bounds for $\sigma_k(\mathbf{V}_k^\top \mathbf{X})$. They include random sign matrices [Ach01, MZ11], the fast randomized Hadamard transform [AC09, Sar06], the subsampled randomized Hadamard transform [DMMS11, Tro11], the fast randomized discrete cosine transform [NDT09], and input sparsity time embeddings [CW13, MM13, NN13].

In contrast, keeping $\|\mathbf{V}_{k,\perp}^\top \mathbf{X}\|_F$ small is relatively easy. For typical random matrices $\mathbf{X}$, one can show that, with high probability,

$$\|\mathbf{V}_{k,\perp}^\top \mathbf{X}\|_{2,F} \leq c\,\|\mathbf{V}_{k,\perp}\|_F \leq c\sqrt{n},$$

where c is a small constant.

For instance, if $s = 1$ and $\mathbf{X}$ is a Gaussian column vector, then

$$\mathbf{E}\left[\|\mathbf{V}_{k,\perp}^\top \mathbf{X}\|_F^2\right] = \|\mathbf{V}_{k,\perp}^\top\|_F^2 \leq n. \tag{6.29}$$

Markov's inequality guarantees that, with probability at least .9,

$$\|\mathbf{V}_{k,\perp}^\top \mathbf{X}\|_F \leq \sqrt{10n}.$$

Essentially all randomized embedding matrices satisfy variants of eqn. (6.29), and we expect the iteration count q in eqn. (6.28) to be logarithmic in n.

From a numerical point of view, a starting guess $\mathbf{X}$ with orthonormal columns is preferable. Thus one could pick a random matrix $\mathbf{X}$ and apply a thin QR decomposition $\mathbf{X} = \mathbf{QR}$. However, this significantly complicates the derivation of bounds for

$\|\mathbf{V}_{k,\perp}^\top \mathbf{X}\|_{2,F}$ and $\|(\mathbf{V}_k^\top \mathbf{X})^\dagger\|_2$, as most matrix concentration inequalities apply only to the original random matrix $\mathbf{X}$, not to its orthonormal basis $\mathbf{Q}$. For instance, if $\mathbf{X}$ is a random matrix whose entries are ± 1 with equal probability, then $\mathbf{Q}$ does not inherit this property. Fortunately, the subsampled Hadamard transform [DMMS11, Tro11] is one of a few random matrices with orthonormal columns, hence amenable to application of matrix concentration inequalities.

6.6 TeraPCA: Principal Component Analysis of genetic data at scale

This section presents a brief introduction into TeraPCA, a software package that approximates principal components using the power of the randomized subspace iteration [Saa11, HMT11]. We refer the interested reader in [BKK+19] for more information.

We selected to introduce this topic in Chapter 6 for two reasons; first the randomized subspace iteration is closely related to the block Krylov suspace. Indeed, the subspace considered in the randomized subspace iteration is the last element of $\mathcal{K}_q$. Second, the linear-algebraic objective of TeraPCA is the approximation of the top leading singular vectors of a covariance matrix (see, Section 2.2.3) and the main concern is how much informative these approximate vectors are (see, [Sai19] for an analysis of the randomized subspace iteration in terms of principal angles and low rank approximations).

Sections of the text that follows have been published in [BKK+19]

TeraPCA: a Fast and Scalable Software Package to Study Genetic Variation in Tera-scale Genotypes, A. Bose, V. Kalantzis, E-M. Kontopoulou, M. Elkady, P. Paschou and P. Drineas in Oxford Bioinformatics (2019), Vol. 35(19), pp. 3679-3683

6.6.1 Motivation and design

Motivated by the continuously increasing size of modern datasets as well by the popularity of PCA in the study of genetic variations, we built TeraPCA, a C++ multi-threaded library that combines out-of-core operations with the block Randomized Subspace Iteration (RSI) to accurately compute (up to a given accuracy) top principal components.

TeraPCA fetches row-wise blocks from the secondary to the main memory (see, Fig. 6.1) with the goal to reduce the communication between secondary memory and CPU as much as possible. Algorithm 20 provides a high level sketch of the procedure. Each product $\mathbf{C} = \mathbf{A}(\mathbf{A}^{\top}\mathbf{X})$ that appears in steps 0 and 3 is computed in an out-of-core manner that is described in Algorithm 21. The full software package can be downloaded from: `https://github.com/aritra90/TeraPCA`.

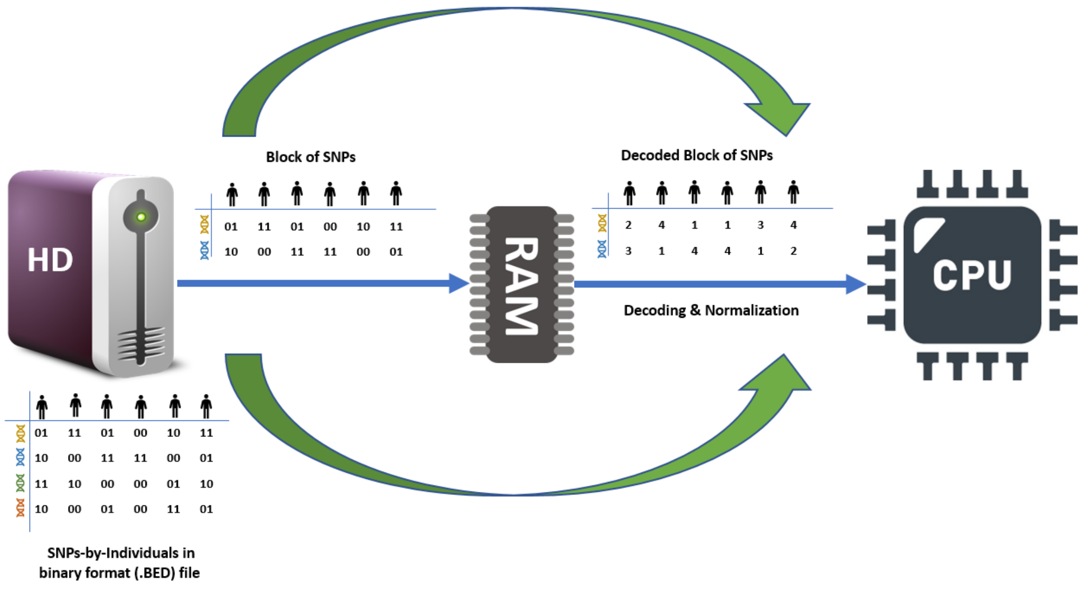


Figure 6.1.: TeraPCA procedure.

Algorithm 20 TeraPCA scheme for randomized subspace iteration

```
Input: $\mathbf{A}^\top \in \mathbb{R}^{n\times m}$, initial guess matrix $\mathbf{X}_0 \in \mathbb{R}^{m\times s}$ with elements drawn i.i.d. from
    the normal distribution $\mathcal{N}(0,1)$, $k \geq 1$, and $s \geq k$.
Output: The $k$ leading approximate left singular vectors of $\mathbf{A}$.
1: $\mathbf{C} = \mathbf{A}(\mathbf{A}^\top \mathbf{X}_0)$
2: repeat
3:     $\mathbf{Q} = \texttt{orth}(\mathbf{C})$
4:     $\mathbf{C} = \mathbf{A}\mathbf{A}^\top \mathbf{Q}$
5:     $\mathbf{M} = \mathbf{Q}^\top \mathbf{C}$
6:     Compute the eigenvalue decomposition $\mathbf{M} = \mathbf{X}\mathbf{D}\mathbf{X}^\top$
7:     $\mathbf{C} = \mathbf{Q}\mathbf{X}$
8: until convergence
9: return first $k$ columns of $\mathbf{Q}$
```

Algorithm 21 Out-of-core matrix-multiVector multiplication for $\mathbf{A}(\mathbf{A}^\top\mathbf{X})$

```
Input: $\zeta > 0$, $\mathbf{X} \in \mathbb{R}^{m\times s}$.
Output: $\mathbf{C} \in \mathbb{R}^{m\times s}$.
1: $\mathbf{C} = \mathbf{0}$
2: for $i = 1 : \zeta$ do
3:     Fetch the $i$-th row-block of $\mathbf{A}^\top$
4:     $\mathbf{C} = \mathbf{C} + \mathbf{A}_i(\mathbf{A}_i^\top \mathbf{X})$
5: end for
```

6.6.2 Empirical evaluation

The implementation of the randomized subspace iteration is based on the state-of-the-art dense linear algebra libraries, BLAS [BDD$^+$02] and LAPACK [ABB$^+$99]. All our experiments ran at Purdue's Brown cluster on a dedicated node which features an Intel Xeon Gold 6126 processor at 2.6 GHz with 96 GB of RAM.

We evaluated TeraPCA on real-world datasets as well as synthetic ones, that were generated using a genetic data generator we implemented and is heavily based on the R-script from [GHBS16]. The data generator can be downloaded from `https://github.com/eugeniamaria/DataSimulator`. Table 6.1 summarizes the datasets and their characteristics, which we used in our empirical evaluation.

We tested TeraPCA against its basic competitor, FlashPCA2 [AQI17]. Tables 6.2 and 6.3 indicate that TeraPCA is an appealing alternative to FlashPCA2.

Table 6.1.: Simulated and real datasets used for the experiments. The .PED and .BED filetypes represent compressed files and are ubiquitously used in genetics to describe genetic data. The number of samples and the number of SNPs (Single Nucleotide Polymorphism) correspond to the number of rows and the number of columns of the data matrix respectively.

Dataset	Size (.PED file)	Size (.BED file)	# Samples	# SNPs
S_1 (simulated)	19 GB	120 MB	5,000	1,000,000
S_2 (simulated)	38 GB	239 MB	10,000	1,000,000
S_3 (simulated)	373 GB	24 GB	100,000	1,000,000
S_4 (simulated)	1.9 TB	117 GB	500,000	1,000,000
S_5 (simulated)	3.7 TB	233 GB	1,000,000	1,000,000
S_6 (simulated)	38 GB	2.4 GB	100,000	100,000
S_7 (simulated)	150 GB	9.4 GB	2,000	20,000,000
HGDP	615 MB	39 MB	1,043	154,417
1000 Genomes	8.4 GB	483 MB	2,504	808,704
PRK	2 GB	126 MB	4,706	111,831
T2D	1.8 GB	111 MB	6,370	72,457

Table 6.2.: Time comparison between TeraPCA and FlashPCA2 when approximating the top ten principal components. The size of the initial subspace was set to s. In the table * indicates no convergence after 50 hrs. The maximum RAM size allowed for both libraries was 2GB.

Dataset	TeraPCA	FlashPCA2	Speed-up
S_1	26.2 mins	33.3 mins	1.27
S_2	39.3 mins	87.5 mins	2.22
S_3	7.9 hrs	35.6 hrs	4.50
S_4	7.3 hrs	n/a*	∞
S_5	13.2 hrs	n/a*	∞
S_6	39.5 mins	141.1 mins	3.57
S_7	37.3 mins	106.5 mins	2.86
HGDP	6.5 secs	7.7 secs	1.22
1000 Genomes	4.3 mins	3.5 mins	0.81
T2D	96 secs	119 secs	1.24
PRK	76 secs	73 secs	0.96

We further compared the principal components returned by TeraPCA against those returned by LAPACK. Figure 6.2 shows that the top principal components computed by TeraPCA are numerically close to those computed by LAPACK.

Table 6.3.: Accuracy of the ten leading eigenvalues computed by TeraPCA and by FlashPCA2.

eigenvalue index	relative error		eigenvalue index	relative error	
	TeraPCA	FlashPCA2		TeraPCA	FlashPCA2
1	9.91E-15	1.74E-03	6	3.01E-06	7.63E-04
2	1.02E-13	1.30E-03	7	3.36E-06	1.47E-03
3	5.65E-11	1.49E-03	8	1.04E-05	6.81E-04
4	2.18E-08	1.31E-03	9	7.11E-05	1.28E-03
5	2.65E-06	1.10E-03	10	1.74E-04	7.44E-04

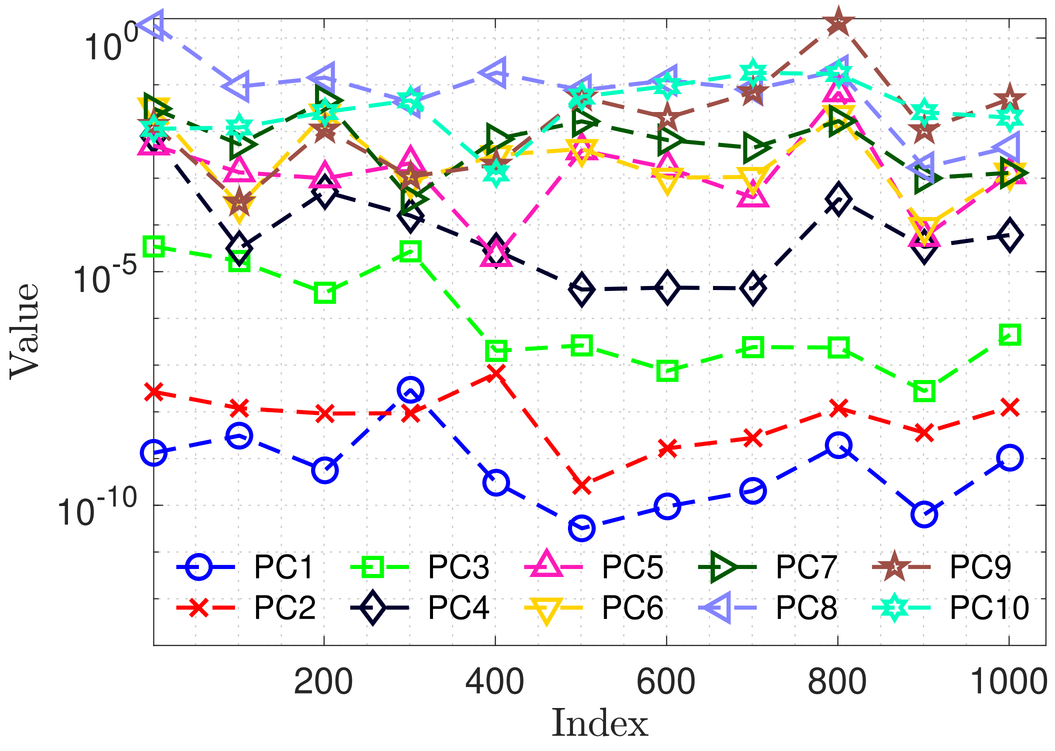


Figure 6.2.: Element-wise relative error of the ten leading principal components computed by TeraPCA against those computed by LAPACK for the HGDP dataset.

Finally, we evaluated the qualitative performance of TeraPCA. Figure 6.3 shows that the top-two principal components computed using TeraPCA can efficiently cluster the various populations.

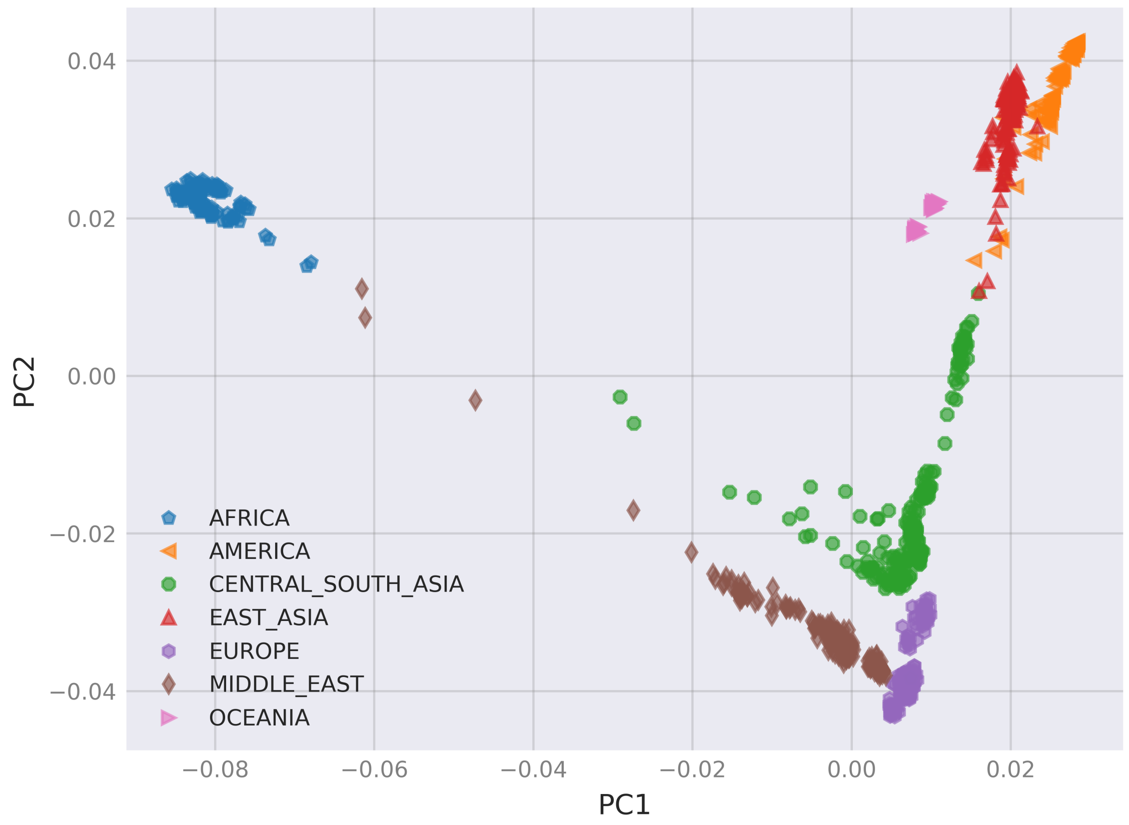


Figure 6.3.: The projection of the HGDP dataset along the two leading principal components computed by TeraPCA.

7 FUTURE DIRECTIONS

This chapter summarizes future directions for each of the problems we have discussed in earlier chapters. The structure of the chapter is as follows: Sections 7.1 and 7.2 describe future directions for the logarithm of the determinant of an SPD matrix and the Von Neumann entropy respectively. Sections 7.3 discusses future directions for sparse principal component analysis. Finally, Section 7.4 discusses future direction on the problem of approximating dominant singular spaces from block Krylov subspaces.

7.1 Approximation of the logarithm determinant of a symmetric positive definite matrix

There are no many open problems in the case of the logarithm of the determinant of a symmetric positive definite matrix. We can easily extend our approach to the Hermitian case, using the results from Section 4.4.

An interesting but difficult problem is the extension of methods likes ours on the approximation of the logarithm determinant of arbitrary nonsingular matrices. The difficulty lies in the fact that the logarithm of a negative singular value is complex. This can potentially be a useful extension for problems that need to approximate determinants of large matrices.

7.2 Estimation of the Von Neumann entropy of density matrices

An interesting open problem would be to consider the estimation of the cross entropy. The cross entropy is a measure between two probability distributions and is particularly important in information theory. Algebraically, it can be defined as $\mathcal{H}\left(\mathbf{S},\mathbf{R}\right) = -\mathbf{Tr}\left(\mathbf{S}\log\mathbf{R}\right)$, where $\mathbf{S} \in \mathbb{C}^{n\times n}$ and $\mathbf{R} \in \mathbb{C}^{n\times n}$ are density matrices with a full set of pure states. One can further extend our polynomial-based approaches using the Taylor expansion or the Chebyshev polynomials to approximate the matrix $\Gamma = \mathbf{S} \log \mathbf{R}$. The case where both or one of the density matrices have an incomplete set of pure states is an open problem: if $\mathbf{R}$ is low-rank, then our first two approaches would not work for the reasons discussed in Section 2.5.3. However, if the only low rank matrix is $\mathbf{S}$, then our first two approaches would still work: $\mathbf{S}$ is only appearing in the trace estimation part, and having eigenvalues equal to zero does not affect the positive semi-definiteness of Γ. When $\mathbf{R}$ is of low rank then one might be able to use our random projection approaches to reduce its dimensionality and/or the dimensionality of $\mathbf{S}$.

Another, interesting open problem would be to develop methods similar to ours for the Rényi entropy. The Rényi entropy is a generalization of mostly all known entropies like the Shannon entropy, the Von Neumann entropy, the min entropy, e.t.c. Given an order $\alpha \geq 0$ with $\alpha \neq 1$ and a random variable $\mathbf{X}$ with discrete outcomes $1, 2, \ldots, n$ and corresponding probabilities $p_1, p_2, \ldots, p_n$:

$$\mathcal{H}_\alpha \mathbf{X} = \frac{1}{1-\alpha} \log \left(\sum_{i=1}^{n} p_i^\alpha \right). \tag{7.1}$$

When $\alpha = 1$ then

$$\mathcal{H}_1(\mathbf{X}) = \lim_{\alpha \to 1} \mathcal{H}_\alpha(\mathbf{X}) = -\sum_{i=1}^{n} p_i \log(p_i)$$

which is the Shannon (or Von Neumann) entropy. The term $\log\left(\sum_{i=1}^{n} p_i^\alpha\right)$ of eqn. (7.1) can be defined as the $\mathbf{Tr}\left(\log\left[\alpha \mathbf{\Sigma}_p\left(\mathbf{X}\right)\right]\right) = \log\left(\|\mathbf{\Sigma}_p\|_\alpha^\alpha\right)$, where $\mathbf{\Sigma}_p$ is the diagonal matrix of the probabilities p_i, $i = 1, \ldots, n$ and $\|\mathbf{\Sigma}_p\|_\alpha$ is the α-Shatten norm. The last equality holds since the p_i's are probabilities and thus non-negative. A simple method would be to use one of the methods that have been described in [UCS17] and [HMAS17] for the approximation of traces of matrix functions, or methods that approximate Shatten p-norms like [LW16] or [Bra18].

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

Finally, it would be of increased interest, especially in the quantum mechanics community, if our third approach is extended to the Hermitian case. In reality most of the quantum systems are not having a full set of pure states; indeed this is an extremely rare phenomenon. Furthermore, the majority of quantum systems are described by complex density matrices. In order to extend our third approach to the Hermitian case, we need to extend the random projection theory to Hermitian matrices which requires extending matrix concentration inequalities to the complex space, with the latter being the most demanding step.

7.3 A randomized rounding algorithm for sparse principal component analysis (PCA)

From a theoretical perspective, it would be interesting to explore whether other relaxations of the sparse PCA problem of eqn. (1.14) e.g. semi-definite relaxations, combined with randomized rounding procedures, could improve our error bounds in Theorem 12. It would also be interesting to formally analyze the deflation algorithm that computes more than one sparse singular vectors in a randomized manner (Algorithm 18). Another interesting aspect would be to somehow enforce orthogonality of the sparse principal components. This is probably the hardest of the open problems. Finally, from a complexity theory perspective, we are not aware of any inapproximability results for the sparse PCA problem; to the best of our knowledge, it is not known whether relative error approximations are possible (without any assumptions on the input matrix).

7.4 Structural convergence results for approximation of dominant subspaces from block Krylov spaces

A very important issue in this work is the gap between low-rank approximations and dominant subspace computations, so an important future direction would have been towards bridging this gap. Furthermore, as we mentioned in Chapter 6, a singular value gap is demanded for well-posed dominant subspace computations but it is not necessary for certain starting guesses in the case of low-rank approximations. To the best of our knowledge, gap-independent results are not known for arbitrary $\mathbf{X}$.

A very interesting future direction would be to understand if it is possible to relax the full-rank assumption for $\mathbf{V}_k^\top \mathbf{X}$. Our proofs require rank $\left(\mathbf{V}_{\mathrm{k}}^\top \mathbf{X}\right) = \mathrm{k}$, which forces starting guesses to have at least $s \geq k$ columns. Thus, even in the presence of the requisite singular value gaps, our proofs collapse for starting guesses that consist of a single column.

Finally, it is important to understand if our bounds are tight enough to be informative and helpful in practical numerical implementations of block Krylov methods for low-rank approximation and dominant subspace reconstructions.

8 CONCLUSION

The problems we have selected to investigate appear as computational bottlenecks in many modern applications, including machine learning, computer graphics, deep learning, genetics, e.t.c. which directly benefit large scale data analytics. The primal goal of this work is to encourage the practical use of RandNLA approaches to solve Big Data bottlenecks at industrial level. The extensive evaluation tests are complemented by a thorough theoretical analysis that proves their accuracy and highlights their scalability as the volume of data increases. Finally, the targeted low computation time and the low memory consumption as well as the simple implementation scheme of the proposed methods (that can easily be extended in parallel or distributed environments) render them suitable for use in the development of highly efficient real-world software.

REFERENCES

[ABB$^+$99] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. SIAM, Philadelphia, PA, third edition, 1999.

[ABF16] F. Arrigo, M. Benzi, and C. Fenu. Computation of Generalized Matrix Functions. *SIAM J. Matrix Anal. Appl.*, 37(3):836–860, 2016.

[AC09] N. Ailon and B. Chazelle. The Fast Johnson–Lindenstrauss Transform and Approximate Nearest Neighbors. *SIAM J. Comput.*, 39(1):302–322, 2009.

[Ach01] D. Achlioptas. Database-friendly Random Projections. In *ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 274–281, 2001.

[AQI17] G. Abraham, Y. Qiu, and M. Inouye. FlashPCA2: Principal Component Analysis of Biobank-scale Genotype Datasets. *Bioinformatics*, 33(17):2776–2778, 2017.

[AT11] H. Avron and S. Toledo. Randomized Algorithms for Estimating the Trace of an Implicit Symmetric Positive Semi-definite Matrix. *J. ACM*, 58(2):8, 2011.

[AW08] A. A. Amini and M. J. Wainwright. High-dimensional Analysis of Semidefinite Relaxations for Sparse Principal Components. In *IEEE International Symposium on Information Theory (ISIT)*, pages 2454–2458, 2008.

[BD09] N. Bosner and Z. Drmač. Subspace Gap Residuals for Rayleigh-Ritz Approximations. *SIAM J. Matrix Anal. Appl.*, 31(1):54–67, 2009.

[BDD$^+$02] S. Blackford, J. Demmel, J. Drongarra, I. Duff, S. Hammarling, G. Henry, M. Heroux, L. Kaufman, A. Lumsdaine, A. Petitet, R. Pozo, K. Remington, and R. C. Whaley. An Updated Set of Basic Linear Algebra Subprograms (BLAS). *ACM Trans. Math. Softw.*, 28(2):135–151, 2002.

[BDHS11] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz. Minimizing Communication in Numerical Linear Algebra. *SIAM J. Matrix Anal. Appl.*, 32(3):866–901, 2011.

[BDK$^+$17] C. Boutsidis, P. Drineas, P. Kambadur, E.-M. Kontopoulou, and A. Zouzias. A Randomized Algorithm for Approximating the Log Determinant of a Symmetric Positive Definite Matrix. *Linear Algebra Appl.*, 533:95–117, 2017.

[BDMI11] C. Boutsidis, P. Drineas, and M. Magdon-Ismail. Near-optimal Column-based Matrix Reconstruction. In *Foundations of Computer Science (FOCS)*, pages 305–314, 2011.

[BDMI14] C. Boutsidis, P. Drineas, and M. Magdon-Ismail. Near-optimal Column-based Matrix Reconstruction. *SIAM J. Comput.*, 43(2):687–717, 2014.

[BER04] C. Beattie, M. Embree, and J. Rossi. Convergence of Restarted Krylov Subspaces to Invariant Subspaces. *SIAM J. Matrix Anal. Appl.*, 25(4):1074–1109, 2004.

[BES05] C. Beattie, M. Embree, and D. C. Sorensen. Convergence of Polynomial Restart Krylov Methods for Eigenvalue Computations. *SIAM Rev.*, 47(3):492–515, 2005.

[BGS06] S. L. Braunstein, S. Ghosh, and S. Severini. The Laplacian of a Graph as a Density Matrix: A Basic Combinatorial Approach to Separability of Mixed States. *Ann. Comb*, 10:291–317, 2006.

[Bjö15] A. Björck. *Numerical Methods in Matrix Computations.* Springer, 2015.

[BKK$^+$19] A. Bose, V. Kalantzis, E.-M. Kontopoulou, M. Elkady, P. Paschou, and P. Drineas. TeraPCA: a Fast and Scalable Software Package to Study Genetic Variation in Tera-scale Genotypes. *Bioinformatics*, 35(19):3679–3683, 2019.

[BP99] R. P. Barry and R. K. Pace. Monte Carlo Estimates of the Log Determinant of Large Sparse Matrices. *Linear Algebra Appl.*, 289(1):41–54, 1999.

[BR05] J. Baglama and L. Reichel. Augmented Implicitly Restarted Lanczos Bidiagonalization Methods. *SIAM J. Sci. Comput.*, 27(1):19–42, 2005.

[BR06] J. Baglama and L. Reichel. Restarted Block Lanczos Bidiagonalization Methods. *Numer. Alg.*, 43:251–272, 2006.

[Bra18] Vladimir Braverman. Approximations of Schatten Norms via Taylor Expansions. 2018.

[CCFC04] M. Charikar, K. Chen, and M. Farach-Colton. Finding Frequent Items in Data Streams. *Theor. Comput. Sci.*, 312(1):3–15, 2004.

[CHHS20] H. Choi, J. He, H. Hu, and Y. Shi. Fast Computation of Von Neumann Entropy for Large-Scale Graphs via Quadratic Approximations. *Linear Algebra Appl.*, 585:127–146, 2020.

[CJ95] J. Cadima and I. T. Jolliffe. Loading and Correlations in the Interpretation of Principle Compenents. *J. Appl. Stat.*, 22(2):203–214, 1995.

[Con07] The International HapMap Consortium. A Second Generation Human Haplotype Map of over 3.1 Million SNPs. *Nature*, 449:851–861, 2007.

[CSS18] H. Choi, X. Song, and Y. Shi. Randomized Method for Estimating the Von Neumann Entropy of Large Scale Density Matrices. In *IEEE Global Conference on Signal and Information Processing*, pages 296–300, 2018.

[Cur09] P. F. Curran. On a Variation of the Gershgorin Circle Theorem with Applications to Stability Theory. *IET Irish Signals and Systems Conference*, 2009.

[CW13] K. L. Clarkson and D. P. Woodruff. Low Rank Approximation and Regression in Input Sparsity Time. In *ACM Symposium on Theory of Computing (STOC)*, pages 81–90, 2013.

[CWLR19] P.-Y. Chen, L. Wu, S. Liu, and I. Rajapakse. Fast Incrimental Von Neumann Graph Entropy Computation: Theory, Algorithm and Applications. In *International Conference on Machine Learning (ICML)*, pages 1869–1886, 2019.

[Dav75] E. R. Davidson. The Iterative Calculation of a few of the Lowest Eigenvalues and Corresponding Eigenvectors of Large Real-symmetric Matrices. *J. Comput. Phys.*, 17(1):87–94, 1975.

[Dav06] T. A. Davis. *Direct Methods for Sparse Linear Systems*, volume 2. SIAM, 2006.

[dBEG08] A. d'Aspremont, O. Banerjee, and L. El Ghaoui. First-order Methods for Sparse Covariance Selection. *SIAM J. Matrix Anal. Appl.*, 30(1):56–66, 2008.

[dBG08] A. d'Aspremont, F. Bach, and L. E. Ghaoui. Optimal Solutions for Sparse Principal Component Analysis. *J. Mach. Learn. Res.*, 9:1269–1294, 2008.

[dBG14] A. d'Aspremont, F. Bach, and L. El Ghaoui. Approximation Bounds for Sparse Principal Component Analysis. *Math. Program., Ser. B*, pages 89–110, 2014.

[dGJ07] A. d'Aspremont, L. E. Ghaoui, and M. I. Jordan. A Direct Formulation for Sparse PCA using Semidefinite Programming. *SIAM Rev.*, pages 434–448, 2007.

[DH11] T. A. Davis and Y. Hu. The University of Florida Sparse Matrix Collection. *ACM Trans. Math. Softw.*, 38(1):1–25, 2011.

[DI19] P. Drineas and I. C. F. Ipsen. Low-Rank Matrix Approximations do not Need a Singular Value Gap. *SIAM J. Matrix Anal. Appl.*, 40(1):299–319, 2019.

[DIKMI18] P. Drineas, I. Ipsen, E.-M. Kontopoulou, and M. Magdon-Ismail. Structural Convergence Results for Approximation of Dominant Subspaces from Block Krylov Spaces. *SIAM J. Matrix Anal. Appl.*, 39:567–586, 2018.

[Dix83] J. D. Dixon. Estimating Extreme Eigenvalues and Condition Numbers of Matrices. *SIAM J. Numer. Anal.*, 20(4):812–814, 1983.

[DL20] R. Drikvandi and O. Lawal. Sparse Principal Component Analysis for Natural Language Processing. *Ann. Data. Sci.*, 2020.

[DM16] P. Drineas and M. W. Mahoney. RandNLA: Randomized Numerical Linear Algebra. *Commun. ACM*, 59(6):80–90, 2016.

[DM18] P. Drineas and M. W. Mahoney. Lectures on Randomized Numerical Linear Algebra. *The Mathematics of Data , IAS/Park City Math. Ser.*, 25, 2018.

[DMM08] P. Drineas, M. W. Mahoney, and S. Muthukrishnan. Relative Error CUR Matrix Decompositions. *SIAM J. Matrix Anal. Appl.*, 30(2):844–881, 2008.

[DMMS11] P. Drineas, M. W. Mahoney, S. Muthukrishnan, and T. Sarlós. Faster Least Squares Approximation. *Numer. Math.*, 117:219–249, 2011.

[Drm96] Z. Drmač. On Relative Residual Bounds for the Rigenvalues of a Hermitian Matrix. *Linear Algebra Appl.*, 244:155–163, 1996.

[DV92] J. Demmel and K. Veselic. Jacobi's Method is More Accurate than QR. *SIAM J. Matrix Anal. Appl.*, 13(4):1204–1245, 1992.

[EY36] C. Eckart and G. Young. The Approximation of one Matrix by Another of Lower Rank. *Psychometrika*, 1(3):1–211, 1936.

[EZM$^+$20] N. B. Erichson, P. Zheng, K. Manohar, S. L. Brunton, J. N. Kutz, and A. Y. Aravkin. Spare Principal Compenent Analysis via Variable Projection. *SIAM J. Appl. Math.*, 80(2):977–1002, 2020.

[FHT08] J. Friedman, T. Hastie, and R. Tibshirani. Sparse Inverse Covariance Estimation with the Graphical LASSO. *Biostatistics*, 9(3):432–441, 2008.

[FKKD17] K. Fountoulakis, A. Kundu, E.-M. Kontopoulou, and P. Drineas. A Randomized Rounding Algorithm for Sparce PCA. *ACM Trans. Knowl. Discov. Data*, 11(3):1–26, 2017.

[GB$^+$10] G. Guennebaud, J. Benoît, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[GHBS16] P. Gopalan, W. Hao, D. M. Blei, and J. Storey. Scaling Probabilistic Models of Genetic Variation to Millions of Humans. *Nature Genetics*, 48:1587–1590, 2016.

[GKK97] A. Gupta, G. Karypis, and V. Kumar. Highly Scalable Parallel Algorithms for Sparse Matrix Factorization. *IEEE Trans. Parallel Distrib. Syst.*, 8(5):502–520, 1997.

[Gu15] M. Gu. Subspace Iteration Randomization and Singular Value Problems. *SIAM J. Sci. Comput.*, 37(3):A1139–A1173, 2015.

[Gup00] A. Gupta. WSMP: Watson Sparse Matrix Package (Part-I: direct solution of symmetric sparse systems). 2000.

[GV96] G. H. Golub and C. F. Van Loan. *Matrix Computations (3rd Ed.)*. Johns Hopkins University Press, 1996.

[GV13] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, fourth edition, 2013.

[GVDG08] K. Goto and R. Van De Geijn. High-performance Implementation of the Level-3 BLAS. *ACM Trans. Math. Softw.*, 35(1):4, 2008.

[HAB14] T. Hunter, A. E. Alaoui, and A. Bayen. Computing the Log-determinant of Symmetric, Diagonally Dominant Matrices in Near-linear Time. *ArXiv*, 2014.

[HBI73] J. B. Hawkins and A. Ben-Israel. On Generalized Matrix Functions. *Linear and Multilinear Algebra*, 1(2):163–171, 1973.

[HHW18] J. Hass, C. Heil, and M. Weir. *Thoma's Calculus*. Pearson, 2018.

[Hig08] N. Higham. *Functions of Matrices: Theory and Computation*. SIAM, 2008.

[HJ91] R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1991.

[HMAS17] I. Han, D. Malioutov, H. Avron, and J. Shin. Approximating the Spectral Sums of Large-scale Matrices using Chebyshev Approximations. *SIAM J. Sci. Comput.*, 39(4):1558—1585, 2017.

[HMS15] I. Han, D. Malioutov, and J. Shin. Large-scale Log-determinant Computation through Stochastic Chebyshev Expansions. In *International Conference on Machine Learning (ICML)*, volume 37, pages 908–917, 2015.

[HMT11] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. *SIAM Rev.*, 53(2):217–288, 2011.

[HNO08] N. J. A. Harvey, J. Nelson, and K. Onak. Sketching and Streaming Entropy via Approximation Theory. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 489–498, 2008.

[HSD$^+$13] C.-J. Hsieh, M. A. Sustik, I. S. Dhillon, P. K. Ravikumar, and R. Poldrack. BIG & QUIC: Sparse Inverse Covariance Estimation for a Million Variables. In *Advances in Neural Information Processing Systems (NIPS)*, volume 2, pages 3165–3173, 2013.

[JI92] E. R. Jessup and I. C. F. Ipsen. Improving the Accuracy of Inverse Iteration. *SIAM J. Sci. Stat. Comput.*, 13(3):550–72, 1992.

[Jia17] Z. Jia. The Regularization Theory of the Krylov Iterative Solvers LSQR and CGLS for Linear Discrete Ill-posed Problems, Part I: the Simple Singular Value Case. *ArXiv*, 2017.

[JNRS10] M. Journée, Y. Nesterov, P. Richtárik, and R. Sepulchre. Generalized Power Method for Sparse Principal Component Analysis. *J. Mach. Learn. Res.*, 11:517–553, 2010.

[Joh16] N. Johnston. QETLAB: A Matlab Toolbox for Quantum Entanglement, Version 0.9. `http://qetlab.com`, 2016.

[Jol95] I. T. Jolliffe. Rotation of Principal Components: Choice of Normalization Constraints. *J. Appl. Stat.*, 22(1):29–35, 1995.

[JTU03] I. T. Jolliffe, N. T. Trendafilov, and M. Uddin. A Modified Principal Component Technique Based on the LASSO. *J. Comput. Graph. Stat.*, 12(3):531–547, 2003.

[KDS$^+$20] E. Kontopoulou, G. Dexter, W. Szpankowski, A. Grama, and P. Drineas. Randomized Linear Algebra Approaches to Estimate the Von Neumann Entropy of Density Matrices. *IEEE Trans. Inf. Theory*, 2020.

[KL13] P. Kambadur and A. Lozano. A Parallel, Block Greedy Method for Sparse Inverse Covariance Estimation for Ultra-high Dimensions. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 351–359, 2013.

[Kny87] A. V. Knyazev. Convergence Rate Estimates for Iterative Methods for a Mesh Symmetric Eigenvalue Problem. *Soviet J. Numer. Anal. Math. Modelling*, 2(5):371–396, 1987.

[KW92] J. Kuczyński and H. Woźniakowski. Estimating the Largest Eigenvalue by the Power and Lanczos Algorithms with a Random Start. *SIAM J. Matrix Anal. Appl.*, 13(4):1094–1122, 1992.

[KW94] J. Kuczyński and H. Woźniakowski. Probabilistic Bounds on the Extremal Eigenvalues and Condition Number by the Lanczos Algorithm. *SIAM J. Matrix Anal. Appl.*, 15(2):672–691, 1994.

[LAT$^+$08] J. Z. Li, D. M. Absher, H. Tang, A. M. Southwick, A. M. Casto, S. Ramachandran, H. M. Cann, G. S. Barsh, M. Feldman, L. L. Cavalli-Sforza, and R. M. Myers. Worldwide Human Relationships Inferred from Genome-Wide Patterns of Variation. *Science*, 319(5866):1100–1104, 2008.

[LP01] J. P LeSage and R. K. Pace. Spatial dependence in data mining. In *Data Mining for Scientific and Engineering Applications*, pages 439–460. Springer, 2001.

[LW16] Y. Li and D. Woodruff. On Approximating Functions of the Singular Values in a Stream. In *ACM Symposium on Theory of Computing (STOC)*, pages 726–739, 2016.

[LWGX16] Y. Luo, D. Wen, Y.and Tao, J. Gui, and C. Xu. Large Margin Multi-Modal Multi-Task Feature Extraction for Image Classification. *IEEE Trans. Image Process.*, 25(1):414–427, 2016.

[LZ15] R.-C Li and L.-H Zhang. Convergence of the Block Lanczos Method for Eigenvalue Clusters. *Numer. Math.*, 131:83–113, 2015.

[LZL05] W.E. Leithead, Y. Zhang, and D. J. Leith. Efficient Gaussian Process Based on BFGS Updating and Logdet Approximation. In *16th IFAC World Congress*, 2005.

[LZW$^+$16] W. Liu, Z. Zha, Y. Wang, K. Lu, and D. Tao. p-Laplacian Regularized Sparse Coding for Human Activity Recognition. *IEEE Trans. Ind. Electron.*, 63(8):5120–5129, 2016.

[Mah92] P. J. Maher. Some Norm Inequalities Concerning Generalized Inverses. *Linear Algebra Appl.*, 174:99–110, 1992.

[Mar92] R. J. Martin. Approximations to the Determinant Term in Gaussian Maximum Likelihood Estimation of some Spatial Models. *Commun. Stat. Theory Methods*, 22(1):189–205, 1992.

[MD09] M. W. Mahoney and P. Drineas. CUR Matrix Decompositions for Improved Data Analysis. *Proceedings of the National Academy of Sciences (PNAS)*, 106(3):697–702, 2009.

[MD16] M. W. Mahoney and P. Drineas. Structural Properties Underlying High-Quality Randomized Numerical Linear Algebra Algorithms. In *Handbook of Big Data*, pages 137–154. 2016.

[MM13] X. Meng and M. W. Mahoney. Low-Distortion Subspace Embeddings in Input-Sparsity Time and Applications to Robust Linear Regression. In *ACM Symposium on Theory of Computing (STOC)*, pages 91–100, 2013.

[MM15] C. Musco and C. Musco. Randomized Block Krylov Methods for Stronger and Faster Approximate Singular Value Decomposition. In *Advances in Neural Information Processing Systems (NIPS)*, volume 28, pages 1396–1404. 2015.

[MNS$^+$18] C. Musco, P. Netrapalli, A. Sidford, S. Ubaru, and D. P. Woodruff. Spectrum Approximation Beyond Fast Matrix Multiplication: Algorithms and Hardness. In *Innovations in Theoretical Computer Science (ITCS)*, 2018.

[MWA12] B. Moghaddam, Y. Weiss, and S. Avidan. Generalized Spectral Bounds for Sparse LDA. In *International Conference on Machine Learning (ICML)*, pages 641–648, 2012.

[MZ11] A. Magen and A. Zouzias. Low Rank Matrix-Valued Chernoff Bounds and Approximate Matrix Multiplication. In *ACM Symposium on Discrete Algorihtms (SODA)*, pages 1422–1436, 2011.

[NDT09] N. H. Nguyen, T. T. Do, and T. D. Tran. A Fast and Efficient Algorithm for Low-rank Approximation of a Matrix. In *ACM Symposium on Theory of Computing (STOC)*, pages 215–224, 2009.

[NN13] J. Nelson and H. L. Nguyen. OSNAP: Faster Numerical Linear Algebra Algorithms via Sparser Subspace Embeddings. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2013.

[PB97] R. K. Pace and R. Barry. Quick Computation of Spatial Autoregressive Estimators. *Geogr. Anal.*, 29(3):232–247, 1997.

[PBGS00] R. K. Pace, R. Barry, O. W. Gilley, and C. F. Sirmans. A Method for Spatial–Temporal Forecasting with an Application to Real Estate Prices. *Int. J. Forecast*, 16(2):229–246, 2000.

[PBMID13] S. Paul, C. Boutsidis, M. Magdon-Ismail, and P. Drineas. Random Projections and Support Vector Machines. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2013.

[PDK13] D. S. Papailiopoulos, A. G. Dimakis, and S. Korokythakis. Sparse PCA through Low-rank Approximations. In *International Conference on Machine Learning (ICML)*, volume 28, pages 747—755, 2013.

[PL04] R. K. Pace and J. P. LeSage. Chebyshev Approximation of Log-determinants of Spatial Weight Matrices. *Comput. Stat. Data Anal.*, 45(2):179–196, 2004.

[PLJD10] P. Paschou, J. Lewis, A. Javed, and P. Drineas. Ancestry Informative Markers for Fine-Scale Individual Assignment to Worldwide Populations. *J. Med. Genet.*, 47(12):835–847, 2010.

[PMVdG$^+$13] J. Poulson, . Marker, R. A. Van de Geijn, J. R. Hammond, and N. A. Romero. Elemental: A New Framework for Distributed Memory Dense Matrix Computations. *ACM Trans. Math. Softw.*, 39(2):13, 2013.

[PZ19] S. Park and H. Zhao. Sparse Principal Component Analysis with Missing Observations. *Ann. Appl. Stat.*, 13(2):1016–1042, 2019.

[Reu02] A. Reusken. Approximation of the Determinant of Large Sparse Symmetric Positive Definite Matrices. *SIAM J. Matrix Anal. Appl.*, 23(3):799–818, 2002.

[RKA15] F. Roosta-Khorasani and U. Ascher. Improved Bounds on Sample Size for Implicit Matrix Trace Estimators. *Found. Comput. Math.*, 15:1187–1212, 2015.

[Saa80] Y. Saad. On the Rates of Convergence of the Lanczos and the Block-Lanczos Methods. *SIAM J. Numer. Anal.*, 17(5):687–706, 1980.

[Saa11] Y. Saad. *Numerical Methods for Large Eigenvalue Problems.* Classics in Applied Mathematics. SIAM, revised edition, 2011.

[SAI17] A. K. Saibaba, A. Alexanderian, and I. C. F. Ipsen. Randomized Matrix-free Trace and Log-Determinant Estimators. *Numer. Math.*, 137:353—395, 2017.

[Sai19] A. K. Saibaba. Randomized Subspace Iteration: Analysis of Canonical Angles and Unitary Invariant Norms. *SIAM J. Matrix Anal. Appl.*, 40(1):23–48, 2019.

[Sar06] T. Sarlós. Improved Approximation Algorithms for Large Matrices via Random Projections. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 143–152, 2006.

[SCLE18] K. Sjöstrand, L.H. Clemmensen, R. Larsen, and B. Ersbøll. Spasm: A Matlab Toolbox for Sparse Statistical Modeling. *J. Stat. Softw.*, 84:1–37, 2018.

[SMDS11] J. K. Salmon, M. A. Moraes, R. O. Dror, and D. E. Shaw. Parallel Random Numbers: as easy as 1, 2, 3. In *IEEE Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pages 1–12, 2011.

[SS90] G. W. Stewart and J.-G. Sun. *Matrix Perturbation Theory.* Academic Press, 1990.

[SZ00] H. D. Simon and H. Zha. Low-Rank Matrix Approximation Using the Lanczos Bidiagonalization Process with Applications. *SIAM J. Sci. Comput.*, 21(6):2257–2274, 2000.

[TLWM07] D. Tao, X. Li, X. Wu, and S. Maybank. General Tensor Discriminant Analysis and Gabor Features for Gait Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(10):1700–1715, 2007.

[Tre11] L. Trevisan. Graph Partitioning and Expanders. Handout 7, 2011.

[Tre12] L. N. Trefethen. *Approximation Theory and Approximation Practice.* SIAM, 2012.

[Tro11] J. A. Tropp. Improved Analysis of the Subsampled Randomized Hadamard Transform. *Adv. Adapt. Data Anal.*, 3(1):115–126, 2011.

[UCS17] S. Ubaru, J. Chen, and Y. Saad. Fast Estimation of $tr(f(A))$ via Stochastic Lanczos Quadrature. *SIAM J. Matrix Anal. Appl.*, 38(4):1075–1099, 2017.

[VDTC$^+$19] K. Van Deun, L. Thorrez, M. Coccia, D. Hasdemir, J. A. Westerhuis, A. K. Smilde, and I. Van Mechelen. Weighted Sparse Principal Component Analysis. *Chemom. Intell. Lab. Syst.*, 195(103875), 2019.

[WBS14] T. P. Wihler, B. Bessire, and A. Stefanov. Computing the Entropy of a Large Matrix. *J. Phys. A: Math. Theor.*, 47(24):245201, 2014.

[Woo14] D. Woodruff. Sketching as a Tool for Numerical Linear Algebra. *Found. Trends Theor. Comput. Sci.*, 10:1–157, 2014.

[WZZ15] S. Wang, Z. Zhang, and T. Zhang. Improved Analyses of the Randomized Power Method and Block Lanczos Method. *ArXiv*, 2015.

[YZ13] X.-T. Yuan and T. Zhang. Truncated Power Method for Sparse Eigenvalue Problems. *J. Mach. Learn. Res.*, 14:899–925, 2013.

[ZG06] D. Zeimpekis and E. Gallopoulos. TMG: A MATLAB Toolbox for Generating Term Document Matrices from Text Collections. In *Grouping Multidimensional Data: Recent Advances in Clustering*, pages 187–210. Kogan, J. and Nicholas, C. and Teboulle, M., 2006.

[ZHT06] H. Zou, T. Hastie, and R. Tibshirani. Sparse Principal Component Analysis. *J. Comput. Graph. Stat.*, 15:265–286, 2006.

[ZK13] P. Zhu and A. V. Knyazev. Angles Between Subspaces and their Tangents. *J. Numer. Math.*, 21(4):325–340, 2013.

[ZL07] Y. Zhang and W. E. Leithead. Approximate Implementation of the Logarithm of the Matrix Determinant in Gaussian Process Regression. *J. Stat. Comput. Sim.*, 77(4):329–348, 2007.

[ZLLW08] Y. Zhang, W. E. Leithead, D. J. Leith, and L Walshe. Log-det Approximation Based on Uniformly Distributed Seeds and its Application to Gaussian Process Regression. *J. Comput. Appl. Math.*, 220(1):198–214, 2008.

VITA

Evgenia-Maria Kontopoulou is a Ph.D. candidate at the department of Computer Science, Purdue Univesity, Indiana, U.S.A. Prior to joining Purdue, she acquired her bachelor and master in computer engineering from the Department of Computer Engineering and Informatics, University of Patras, Greece. Her research interests lie in the general area of (Randomized) Numerical Linear Algebra with a focus on designing, analyzing and prototyping algorithms for the solution/approximation of computational bottlenecks especially in the case of Big Data. Except from her research work, she has been employed as a numerical software engineer for MathWorks. During her time in MathWorks she has extensively worked on the extension of MATLAB half precision package. After graduating Purdue she will be joining MathWorks as a senior software engineer.