Foundations and Trends® in
Theoretical Computer Science
Vol. 9, No. 2 (2013) 125–210

DOI: 10.1561/0400000065

now
the essence of knowledge

Faster Algorithms via Approximation Theory

Sushant Sachdeva
Yale University
sushant.sachdeva@yale.edu

Nisheeth K. Vishnoi
Microsoft Research
nisheeth.vishnoi@gmail.com

Contents

Abstract

This monograph presents ideas and techniques from approximation theory for approximating functions such as x^s, x^{-1} and e^{-x}, and demonstrates how these results play a crucial role in the design of fast algorithms for problems which are increasingly relevant. The key lies in the fact that such results imply faster ways to compute primitives such as $A^s v$, $A^{-1}v$, $\exp(-A)v$, eigenvalues, and eigenvectors, which are fundamental to many spectral algorithms. Indeed, many fast algorithms reduce to the computation of such primitives, which have proved useful for speeding up several fundamental computations such as random walk simulation, graph partitioning, and solving systems of linear equations.

S. Sachdeva and N. K. Vishnoi. *Faster Algorithms via Approximation Theory*. Foundations and Trends® in Theoretical Computer Science, vol. 9, no. 2, pp. 125–210, 2013.

DOI: 10.1561/0400000065.

Introduction

A Brief History of Approximation Theory

The area of approximation theory is concerned with the study of how well functions can be approximated by simpler ones. While there are several notions of *well* and *simpler*, arguably, the most natural notion is that of *uniform* approximations by *polynomials*: Given a function $f : \mathbb{R} \mapsto \mathbb{R}$ and an interval $\mathcal{I}$, what is the closest a degree d polynomial can remain to $f(x)$ throughout the entire interval? Formally, if Σ_d is the class of all univariate real polynomials of degree at most d, the goal is to understand

$$\varepsilon_{f,\mathcal{I}}(d) \stackrel{\text{def}}{=} \inf_{p \in \Sigma_d} \sup_{x \in \mathcal{I}} |f(x) - p(x)|.$$

This notion of approximation, called uniform approximation or Chebyshev approximation, is attributed to Pafnuty Chebyshev, who initiated this area in an attempt to improve upon the *parallel motion* invented by James Watt for his steam engine; see [13]. Chebyshev discovered the *alternation property* of the best approximating polynomial and found the best degree-$d-1$ polynomial approximating the monomial x^d; see [14]. Importantly, the study of this question led to the discovery of, what are now referred to as, Chebyshev polynomials (of the first kind). Chebyshev polynomials find applications in several different areas of science and mathematics and, indeed, repeatedly make an appearance in this monograph due to their extremal properties.[1]

[1] The Chebyshev polynomial of degree-d is the polynomial that arises when one writes $\cos(d\theta)$ as a polynomial in $\cos\theta$.

Despite Chebyshev's seminal results in approximation theory, including his work on best rational approximations, several foundational problems remained open. While it is obvious that $\varepsilon_{f,\mathcal{I}}(d)$ cannot increase as we increase d, it was Weierstrass [67] who later established that, for any continuous function f and a bounded interval $\mathcal{I}$, the error $\varepsilon_{f,\mathcal{I}}(d)$ tends to 0 as d goes to infinity. Further, it was Emile Borel [11] who proved that the best approximation is always *achieved* and is *unique*. Among other notable initial results in approximation theory, A. A. Markov [38], motivated by a question in chemistry posed by Mendeleev, proved that the absolute value of the derivative of a degree d polynomial that is bounded in absolute value by 1 in the interval $[-1,1]$ cannot exceed d^2. These among other results not only solved important problems motivated by science and engineering, but also significantly impacted theoretical areas such as mathematical analysis in the early 1900s.

With computers coming into the foray around the mid 1900s, there was a fresh flurry of activity in the area of approximation theory. The primary goal was to develop efficient ways to calculate mathematical functions arising in scientific computation and numerical analysis. For instance, to evaluate e^x for $x \in [-1,1]$, it is sufficient to store the coefficients of its best polynomial (or rational) approximation in this interval. For a fixed error, such approximations often provided a significantly more succinct representation of the function than the representation obtained by truncating the appropriate Taylor series.

Amongst this activity, an important development occurred in the 1960s when Donald Newman [43] showed that the best degree-d rational approximation to the function $|x|$ on $[-1,1]$ achieves an approximation error of $e^{-\Theta(\sqrt{d})}$, while the best degree-d polynomial approximation can only achieve an error of $\Theta(1/d)$. Though rational functions were also considered earlier, including by Chebyshev himself, it was Newman's result that revived the area of uniform approximation with rational functions and led to several rational approximation results where the degree-error trade-off was exponentially better than that achievable by polynomial approximations. Perhaps the problem that received the most attention, due to its implications to numerical methods for solving systems of partial differential equations (see [19]), was to understand the best rational approximation to e^{-x} over the interval $[0,\infty)$. Rational functions of degree d were shown to approximate e^{-x} on $[0,\infty)$ up to an error of c^d for some constant $c < 1$. This line of research culminated in a land-

mark result in this area by Gonchar and Rakhmanov [20] who determined the optimal c. Despite remarkable progress in the theory of approximation by rational functions, there seems to be no clear understanding as to why rational approximations are often significantly better than polynomial approximations of the same degree, and surprising results abound. Perhaps this is what makes the study of rational approximations promising and worth understanding.

Approximation Theory in Algorithms and Complexity

Two of the first applications of approximation theory in algorithms[2] were the Conjugate Gradient method (see [24, 31]) and the Lanczos method (see [36]), which are used to solve systems of linear equations $Ax = v$ where A is an $n \times n$ real, symmetric, and positive semi-definite (PSD) matrix. These results, which surfaced in the 1950s, resulted in what are called *Krylov subspace methods* and can also be used to speed up eigenvalue and eigenvector computations. These methods are iterative and reduce such computations to a small number of computations of the form Au for different vectors u. Thus, they are particularly suited for sparse matrices that are too large to handled by Gaussian elimination-based methods; see the survey [58] for a detailed discussion.

Until recently, the main applications of approximation theory in theoretical computer science have been in complexity theory. One of the most notable was by Beigel *et al.* [8] who used Newman's result to show that the complexity class PP is closed under intersections and unions.[3] Another important result where approximation theory, in particular Chebyshev polynomials, played a role is the *quadratic* speed-up for quantum search algorithms, initiated by a work by Grover [22]. The fact that one cannot speed up beyond Grover's result was shown by Beals *et al.* [7] which, in turn, relied on the use of Markov's theorem as inspired by Nisan and Szegedy's lower bound for the Boolean OR function [46]. For more on applications of approximation theory to complexity theory, communication complexity and computational learning theory, we refer the reader to [1, 33, 61, 65], and for applications to streaming algorithms to [23].

[2] More precisely, in the area of numerical linear algebra.

[3] PP is the complexity class that contains sets that are accepted by a polynomial-time bounded probabilistic Turing machine which accepts with probability strictly more than $1/2$.

Faster Algorithms via Approximation Theory

The goal of this monograph is to illustrate how classical and modern techniques from approximation theory play a crucial role in obtaining results that are relevant to the emerging theory of fast algorithms. For example, we show how to compute good approximations to matrix-vector products such as $A^s v$, $A^{-1}v$ and $\exp(-A)v$ for any matrix A and a vector v.[4] We also show how to speed up algorithms that compute the top few eigenvalues and eigenvectors of a symmetric matrix A. Such primitives are useful for performing several fundamental computations quickly, such as random walk simulation, graph partitioning, and solving linear system of equations. The algorithms for computing these primitives perform calculations of the form Bu where B is a matrix closely related to A (often A itself) and u is some vector. A key feature of these algorithms is that if the matrix-vector product for A can be computed quickly, e.g., when A is sparse, then Bu can also be computed in essentially the same time. This makes such algorithms particularly relevant for handling the problem of *big data*. Such matrices capture either numerical data or large graphs, and it is inconceivable to be able to compute much more than a few matrix-vector product on matrices of this size.

Roughly half of this monograph is devoted to the ideas and results from approximation theory that we think are central, elegant, and may have wider applicability in TCS. These include not only techniques relating to polynomial approximations but also those relating to approximations by rational functions and beyond. The remaining half illustrates a variety of ways we can use these results to design fast algorithms.

As a simple but important application, we show how to speed up the computation of $A^s v$ where A is a symmetric matrix with eigenvalues in $[-1,1]$, v is a vector and s is a large positive integer. The straightforward way to compute $A^s v$ takes time $O(ms)$ where m is the number of non-zero entries in A, *i.e.*, A's *sparsity*. We show how, appealing to a result from approximation theory, we can bring this running time down to essentially $O(m\sqrt{s})$. We start with a result on polynomial approximation for x^s over the interval $[-1,1]$. Using some of the earliest results proved by Chebyshev, it can be shown that

[4] Recall that the matrix exponential is defined to be $\exp(-A) \stackrel{\text{def}}{=} \sum_{k\geq 0} \frac{(-1)^k A^k}{k!}$.

there is a polynomial p of degree $d \approx \sqrt{s \log 1/\delta}$ that δ-approximates x^s over $[-1,1]$. Suppose $p(x)$ is $\sum_{i=0}^{d} a_i x^i$, then the candidate approximation to $A^s v$ is $\sum_{i=0}^{d} a_i A^i v$. The facts that all the eigenvalues of A lie in $[-1,1]$, and that p is close to x^s in the entire interval $[-1,1]$ imply that $\sum_{i=0}^{d} a_i A^i v$ is close to $A^s v$. Moreover, the time taken to compute $\sum_{i=0}^{d} a_i A^i v$ is easily seen to be $O(md) = O(m\sqrt{s \log 1/\delta})$, which gives us a saving of about $\sqrt{s}$.

When A is the random walk matrix of a graph and v is an initial distribution over the vertices, the result above implies that we can speed up the computation of the distribution after s steps by a quadratic factor. Note that this application also motivates why uniform approximation is the right notion for algorithmic applications, since all we know is the interval in which eigenvalues of A lie while v can be any vector and, hence, we would like the approximating polynomial to be close everywhere in that interval.

While the computation of $\exp(-A)v$ is of fundamental interest in several areas of mathematics, physics, and engineering, our interest stems from its recent applications in algorithms and optimization. Roughly, these latter applications are manifestations of the *multiplicative weights method* for designing fast algorithms, and its extension to solving semi-definite programs via the framework by Arora and Kale [6].[5] At the heart of all algorithms based on the matrix multiplicative weights update method is a procedure to quickly compute $\exp(-A)v$ for a symmetric, positive semi-definite matrix A and a vector v. Since exact computation of the matrix exponential is expensive, we seek an approximation. It suffices to approximate the function e^{-x} on a certain interval. A simple approach is to truncate the Taylor series expansion of e^{-x}. However, we can use a polynomial approximation result for e^{-x} to produce an algorithm that saves a quadratic factor (a saving similar to the application above). In fact, when A has more structure, we can go beyond the square-root.

For fast graph algorithms, often the quantity of interest is $\exp(-t\mathcal{L})v$, where $\mathcal{L}$ is the normalized Laplacian of a graph, $t \geq 0$ and v is a vector. The vector $\exp(-t\mathcal{L})v$ can also be interpreted as the resulting distribution of a t-length continuous-time random walk on the graph with starting distribution v. Appealing to a rational approximation to e^{-x} with some additional prop-

[5] See also [26, 27, 28, 29, 50, 51, 48, 66, 5].

erties, the computation of $\exp(-t\mathcal{L})v$ can be reduced to a small number of computations of the form $\mathcal{L}^{-1}u$. Thus, using the near-linear-time Laplacian solver[6] due to Spielman and Teng [62], this gives an $\widetilde{O}(m)$-time algorithm for approximating $\exp(-t\mathcal{L})v$ for graphs with m edges. In the language of random walks, continuous-time random walks on an undirected graph can be simulated essentially independent of time; such is the power of rational approximations.

A natural question which arises from our last application is whether the Spielman-Teng result (which allows us to perform computations of the form $\mathcal{L}^{-1}u$) is *necessary* in order to compute $\exp(-\mathcal{L})v$ in near-linear time. In our final application of approximation theory, we answer this question in the affirmative: We show that the inverse of a positive-definite matrix can be approximated by a weighted-sum of a small number of matrix exponentials. Roughly, we show that for a PSD matrix A, $A^{-1} \approx \sum_{i=1}^{k} w_i \exp(-t_i A)$ for a small k. Thus, if there happens to be an algorithm that performs computations of the form $\exp(-t_i A)v$ in time T (independent of t_i), then we can compute $A^{-1}v$ in essentially $O(Tk)$ time. Thus, we show that the disparate looking problems of inversion and exponentiation are really the same from a point of view of designing fast algorithms.

Organization

We first present the ideas and results from approximation theory and subsequently we present applications to the design of fast algorithms. While we have tried to keep the presentation self-contained, for the sake of clarity, we have sometimes sacrificed tedious details. This means that, on rare occasions, we do not present complete proofs or do not present theorems with optimal parameters.

In Section 1, we present some essential notations and results from approximation theory. We introduce Chebyshev polynomials in Section 2, and prove certain extremal properties of these polynomials which are used in this monograph. In Sections 3 and 4 we construct polynomial approximations to

[6] A Laplacian solver is an algorithm that (approximately) solves a given system of linear equations $\mathcal{L}x = v$, where $\mathcal{L}$ is a (normalized) graph Laplacian and $v \in \mathrm{Im}(\mathcal{L})$, *i.e.*, it (approximately) computes $\mathcal{L}^{-1}v$; see [66].

the monomial x^s over the interval $[-1,1]$ and e^{-x} over the interval $[0,b]$ respectively. Both results are based on Chebyshev polynomials. In Section 5 we prove a special case of Markov's theorem which is then used to show that these polynomial approximations are asymptotically optimal.

Sections 6–7 are devoted to introducing techniques for understanding rational approximations for the function e^{-x} over the interval $[0,\infty)$. In Section 6, we first show that degree d rational functions can achieve c^d error for some $0 < c < 1$. Subsequently we prove that this result is optimal up to the choice of constant c. In Section 7 we present a proof of the theorem that such geometrically decaying errors for the e^{-x} can be achieved by rational functions with an additional restriction that all its poles be real and negative. We also show how to bound and compute the coefficients involved in this rational approximation result; this is crucial for the application presented in Section 11.

Sections 8–11 contain the presentation of applications of the approximation theory results. In Section 8 we show how the results of Section 3 imply that we can quadratically speed up random walks in graphs. Here, we discuss the important issue of computing the coefficients of the polynomials in Section 3. In Section 9 we present the famous Conjugate Gradient method for iteratively solving symmetric PSD systems $Ax = v$, where the number of iterations depends on the square-root of the condition number of A. The square-root saving is shown to be due to the scalar approximation result for x^s from Section 2. In Section 10 we present the Lanczos method and show how it can be used to approximate the largest eigenvalue of a symmetric matrix. We show how the existence of a good approximation for x^s, yet again, allows a quadratic speedup over the *power method*.

In Section 11 we show how the polynomial and rational approximations to e^{-x} developed in Sections 6 and 7 imply the best known algorithms for computing $\exp(-A)v$. If A is a symmetric and diagonally dominant (SDD) matrix, then we show how to combine rational approximations to e^{-x} with negative poles with the powerful SDD (Laplacian) solvers of Spielman-Teng to obtain near-linear time algorithms for computing $\exp(-A)v$.

Finally, in 12, we show how x^{-1} can be approximated by a sparse sum of the form $\sum_i w_i e^{-t_i x}$ over the interval $(0,1]$. The proof relies on the Euler-

Maclaurin formula and certain bounds derived from the Riemann zeta function. Using this result, we show how to reduce computation of $A^{-1}v$ for a symmetric positive-definite (PD) matrix A to the computation of a small number of computations of the form $\exp(-tA)v$. Apart from suggesting a new approach to solving a PD system, this result shows that computing $\exp(-A)v$ inherently requires the ability to solve a system of equations involving A.

Acknowledgments

We would like to thank Elisa Celis with whom we developed the results presented in Sections 3 and 8. We thank Oded Regev for useful discussions regarding rational approximations to e^{-x}. Finally, many thanks to the anonymous reviewer(s) for several insightful comments which have improved the presentation of this monograph.

Part of this work was done when SS was at the Simons Institute for the Theory of Computing, UC Berkeley, and at the Dept. of Computer Science, Princeton University.

Sushant Sachdeva and Nisheeth K. Vishnoi
11 March 2014

Part I

APPROXIMATION THEORY

1

Uniform Approximations

In this section we introduce the notion of uniform approximations for functions. Subsequently, we prove Chebyshev's alternation theorem which characterizes the best uniform approximation.

Given an interval $\mathcal{I} \subseteq \mathbb{R}$ and a function $f : \mathbb{R} \mapsto \mathbb{R}$, we are interested in *approximations* for f over $\mathcal{I}$. The following notion of approximation will be of particular interest:

Definition 1.1. For $\delta > 0$, a function g is called a δ-approximation to a function f over an interval $\mathcal{I}$ if $\sup_{x \in \mathcal{I}} |f(x) - g(x)| \leq \delta$.

Both finite and infinite intervals $\mathcal{I}$ are considered. Such approximations are known as uniform approximations or Chebyshev approximations. We start by studying uniform approximations using polynomials. The quantity of interest, for a function f, is the best uniform error achievable over an interval $\mathcal{I}$ by a polynomial of degree d, namely, $\varepsilon_{f,\mathcal{I}}(d)$ as defined in the introduction. The first set of questions are:

1. Does $\lim_{d \to \infty} \varepsilon_{f,\mathcal{I}}(d) = 0$?

2. Does there always exist a degree-d polynomial p that achieves $\varepsilon_{f,\mathcal{I}}(d)$?

Interestingly, these questions were not addressed in Chebyshev's seminal work. Weierstrass [67] showed that, for a continuous function f on a bounded interval $[a,b]$, there exist arbitrarily good polynomial approximations, *i.e.*, for every $\delta > 0$, there exists a polynomial p that is a δ-approximation to f on $[a,b]$; see [54] for a proof. The existence and uniqueness of a degree-d polynomial that *achieves* the best approximation $\varepsilon_{f,\mathcal{I}}(d)$ was proved by Borel [11].

The trade-off between the degree of the approximating polynomial and the approximation error has been studied extensively, and is one of the main themes in this monograph.

In an attempt to get a handle on best approximations, Chebyshev showed that a polynomial p is the best degree-d approximation to f over an interval $[-1,1]$ if and only if the maximum error between f and p is achieved exactly at $d+2$ points in $[-1,1]$ with alternating signs, *i.e*, there are

$$-1 \leq x_0 < x_1 \cdots < x_{d+1} \leq 1$$

such that

$$f(x_i) - p(x_i) = (-1)^i \varepsilon$$

where $\varepsilon \stackrel{\text{def}}{=} \sup_{x\in[-1,1]} |f(x) - p(x)|$. We prove the following theorem, attributed to de La Vallee-Poussin which, not only implies the sufficient side of Chebyshev's alternation theorem but often, suffices for applications.

Theorem 1.1. Suppose f is a function over $[-1,1]$, p is a degree-d polynomial, and $\delta > 0$ is such that the *error function* $\varepsilon \stackrel{\text{def}}{=} f - p$ assumes alternately positive and negative signs at $d+2$ increasing points

$$-1 \leq x_0 < \cdots < x_{d+1} \leq 1,$$

and satisfies $|\varepsilon(x_i)| \geq \delta$ for all i. Then, for any degree-d polynomial q, we have $\sup_{x\in[-1,1]} |f(x) - q(x)| \geq \delta$.

Proof. Suppose, on the contrary, that there exists a degree-d polynomial q such that $\sup_{x\in[-1,1]} |f(x) - q(x)| < \delta$. This implies that for all i, we have

$$\varepsilon(x_i) - \delta < q(x_i) - p(x_i) < \varepsilon(x_i) + \delta.$$

Since $|\varepsilon(x_i)| \geq \delta$, the polynomial $q - p$ is non-zero at each of the x_is, and must have the same sign as ε. Thus, $q - p$ assumes alternating signs at the x_is, and hence must have a zero between each pair of successive x_is. This implies that the non-zero degree-d polynomial $q - p$ has at least $d + 1$ zeros, which is a contradiction. □

The above theorem easily generalizes to any finite interval. In addition to the conditions in the theorem, if we also have $\sup_{x\in[-1,1]} |f(x) - p(x)| = \delta$, then p is the best degree-d approximation. This theorem can be used to prove one of Chebyshev's results: The best degree-$(d-1)$ polynomial approximation to x^d over the interval $[-1,1]$ achieves an error of exactly 2^{-d+1} (as we shall see in Theorem 2.1).

Notes

Unlike Chebyshev's result on the best degree-$(d-1)$ approximation to x^d, it is rare to find the best uniform approximation. We present a short discussion on an approach which, often, gives good enough approximations. The idea is to *relax* the problem of finding the best uniform approximation over an interval $\mathcal{I}$ to that of finding the degree-d polynomial p that minimizes the ℓ_2-error $\int_{\mathcal{I}} (f(x) - p(x))^2 \, \mathrm{d}x$. For concreteness, let us restrict our attention to the case when the interval is $[-1,1]$. Algorithmically, we know how to solve the ℓ_2 problem efficiently: It suffices to have an *orthonormal* basis of degree-d polynomials $p_0(x), \ldots, p_d(x)$, *i.e.*, polynomials that satisfy

$$\int_{-1}^{1} p_i(x) p_j(x) \, \mathrm{d}x = \begin{cases} 0 \text{ if } i \neq j \\ 1 \text{ otherwise.} \end{cases}$$

Such an orthonormal basis can be constructed by applying Gram-Schmidt orthonormalization to the polynomials $1, x, \ldots, x^d$ with respect to the uniform measure on $[-1,1]$ [1]. Given such an orthonormal basis, the best degree-d ℓ_2-approximation is given by

$$p(x) = \sum_{i=0}^{d} \widehat{f}_i p_i(x), \quad \text{where} \quad \widehat{f}_i = \int_{-1}^{1} f(x) p_i(x) \, \mathrm{d}x.$$

[1] These orthogonal polynomials are given explicitly by $\left\{ \sqrt{(2d+1)/2} \cdot L_d(x) \right\}$, where $L_d(x)$ denotes the degree-d Legendre polynomials; see [63].

The question then is, if $p(x)$ is the best ℓ_2-approximation to the function $f(x)$, how does it compare to the best uniform approximation to $f(x)$? While we cannot say much in general for such an approximation, if we modify the relaxation to minimize the ℓ_2-error with respect to the weight function $w(x) \stackrel{\text{def}}{=} 1/\sqrt{1-x^2}$, *i.e.*, minimize $\int_{-1}^{1}(f(x)-p(x))^2 \frac{\mathrm{d}x}{\sqrt{1-x^2}}$, then, when f is continuous, the best degree-d ℓ_2-approximation with respect to w turns out be an $O(\log d)$ approximation for the best uniform approximation. Formally, if we let p be the degree-d polynomial that minimizes the ℓ_2-error with respect to w, and let $p^\star$ be the best degree-d uniform approximation, then

$$\sup_{x\in[-1,1]} |f(x)-p(x)| \leq O(\log d) \cdot \sup_{x\in[-1,1]} |f(x)-p^\star(x)|;$$

see [54, Section 2.4] for a proof.

The orthogonal polynomials obtained by applying the Gram-Schmidt process with the weight w defined above, turn out to be *Chebyshev Polynomials*, which are central to approximation theory due to their important extremal properties.

2

Chebyshev Polynomials

In this section we define the Chebyshev polynomials and study some of their important properties that are used extensively in the next several sections.

There are several ways to define Chebyshev polynomials.[1] For a non-negative integer d, if $T_d(x)$ denotes the Chebyshev polynomial of degree d, then they can be defined recursively as follows:

$$T_0(x) \stackrel{\text{def}}{=} 1, T_1(x) \stackrel{\text{def}}{=} x,$$

and for $d \geq 2$,

$$T_d(x) \stackrel{\text{def}}{=} 2xT_{d-1}(x) - T_{d-2}(x). \tag{2.1}$$

For convenience, we extend the definition of Chebyshev polynomials to negative integers by defining

$$T_d(x) \stackrel{\text{def}}{=} T_{|d|}(x)$$

for $d < 0$. It is easy to verify that with this definition, the recurrence given by (2.1) is satisfied for all integers d. Rearranging (2.1), we obtain the following:

[1] The polynomials introduced here are often referred to as the Chebyshev polynomials of the *first* kind.

Proposition 2.1. The Chebyshev polynomials $\{T_d\}_{d\in\mathbb{Z}}$ satisfy the following relation for all $d \in \mathbb{Z}$,

$$xT_d(x) = \frac{T_{d+1}(x) + T_{d-1}(x)}{2}.$$

An important property of Chebyshev polynomials, which is often used to define them, is given by the following proposition which asserts that the Chebyshev polynomial of degree d is exactly the polynomial that arises when one writes $\cos(d\theta)$ as a polynomial in $\cos\theta$.

Proposition 2.2. For any $\theta \in \mathbb{R}$, and any integer d, $T_d(\cos\theta) = \cos(d\theta)$.

This can be easily verified as follows. First, note that

$$T_0(\cos(\theta)) = 1 = \cos(0\cdot\theta) \text{ and } T_1(\cos(\theta)) = \cos(\theta) = \cos(1\cdot\theta).$$

Moreover, by induction,

$$\begin{aligned}\cos(d\theta) &= 2\cdot\cos\theta\cdot\cos((d-1)\theta) - \cos((d-2)\theta)\\ &= 2\cdot\cos\theta\cdot T_{d-1}(\cos\theta) - T_{d-2}(\cos\theta) = T_d(\cos\theta),\end{aligned}$$

and hence, the result follows. This proposition also immediately implies that over the interval $[-1,1]$, the value of any Chebyshev polynomials is bounded by 1 in magnitude.

Proposition 2.3. For any integer d, and $x \in [-1,1]$, we have $|T_d(x)| \le 1$.

In fact, Proposition 2.2 implies that, over the interval $[-1,1]$, the polynomial $T_d(x)$ achieves its extremal magnitude at exactly $d+1$ points $x = \cos(j\pi/d)$, for $j = 0,\ldots,d$, and the sign of $T_d(x)$ alternates at these points. (See Figure 2.1 for a graphical illustration of the first few Chebyshev polynomials.) We can now prove Chebyshev's result mentioned in the notes at the end of the previous section.

Theorem 2.1. For every positive integer d, the best degree-$(d-1)$ polynomial approximation to x^d over $[-1,1]$, achieves an approximation error of 2^{-d+1}, *i.e.*,

$$\inf_{p_{d-1}\in\Sigma_{d-1}} \sup_{x\in[-1,1]} |x^d - p_{d-1}(x)| = 2^{-d+1}.$$

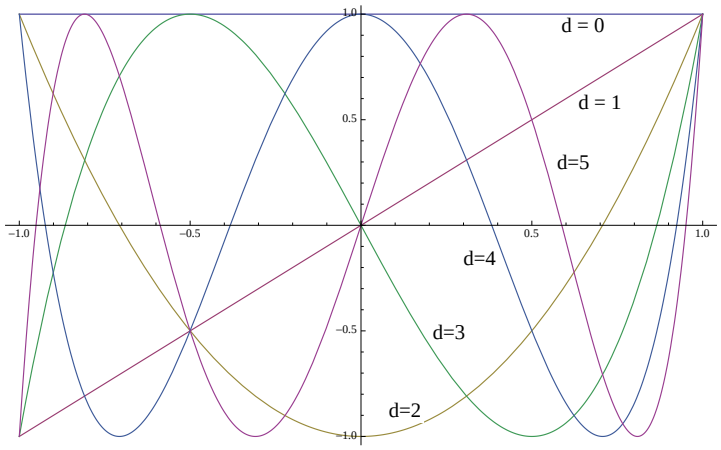

Figure 2.1: Graphs of Chebyshev polynomials $T_d(x)$ on $[-1,1]$ where d is in $\{0,1,\ldots,5\}$.

Proof. Observe that the leading coefficient of $T_d(x)$ is 2^{d-1} and, hence, $q_{d-1}(x) \stackrel{\text{def}}{=} x^d - 2^{-d+1}T_d(x)$ is a polynomial of degree $(d-1)$. The approximation error is given by $x^d - q_{d-1}(x) = 2^{-d+1}T_d(x)$, which is bounded in magnitude on $[-1,1]$ by 2^{-d+1}, and achieves the value $\pm 2^{-d+1}$ at $d+1$ distinct points with alternating signs. The result now follows from Theorem 1.1. □

The fact that $T_d(x)$ takes alternating ± 1 values $d+1$ times in $[-1,1]$, leads to another important property of the Chebyshev polynomials:

Proposition 2.4. For any degree-d polynomial $p(x)$ such that $|p(x)| \leq 1$ for all $x \in [-1,1]$, and any y such that $|y| > 1$, we have $|p(y)| \leq |T_d(y)|$.

Proof. Assume, on the contrary, that there is a y satisfying $|y| > 1$ such that $|p(y)| > |T_d(y)|$, and let

$$q(x) \stackrel{\text{def}}{=} \frac{T_d(y)}{p(y)} \cdot p(x).$$

Hence, $\big|q(x)\big| < |p(x)| \leq 1$ for all $x \in [-1,1]$, and $q(y) = T_d(y)$. Since $|q(x)| < 1$ in $[-1,1]$, between any two successive points where $T_d(x)$ alternates between $+1$ and -1, there must exist an x_i such that $T_d(x_i) = q(x_i)$. Hence, $T_d(x) - q(x)$ has at least d distinct zeros in the interval $[-1,1]$, and

another zero at y. Hence it is a non-zero polynomial of degree at most d with $d+1$ roots, which is a contradiction. □

Surprisingly, this proposition is used in the proof of a lower bound for *rational* approximations to e^{-x} in Section 6. Along with the proposition above, we also need an upper bound on the growth of Chebyshev polynomials outside the interval $[-1,1]$. This can be achieved using the following closed-form expression for $T_d(x)$ which can be easily verified using the recursive definition of Chebyshev polynomials.

Proposition 2.5. For any integer d, and x with $|x| \geq 1$, we have

$$T_d(x) = \frac{1}{2}\left(x+\sqrt{x^2-1}\right)^d + \frac{1}{2}\left(x-\sqrt{x^2-1}\right)^d.$$

Proof. Let $z = x+\sqrt{x^2-1}$. Note that $z^{-1} = x-\sqrt{x^2-1}$. Thus, we wish to prove that for all d, $T_d(x) = 1/2 \cdot \left(z^d + z^{-d}\right)$. It immediately follows that

$$\begin{aligned}
\frac{1}{2}\cdot\left(z^0+z^0\right) &= 1 = T_0(x), \quad \text{and,}\\
\frac{1}{2}\cdot\left(z^1+z^{-1}\right) &= x = T_1(x).
\end{aligned}$$

Moreover, for $d \leq 0$, $T_d(x) = T_{|d|}(x)$. Assuming $T_d(x) = 1/2 \cdot \left(z^d + z^{-d}\right)$ holds for $d = 0,\ldots,k$,

$$\begin{aligned}
T_{k+1}(x) &= 2x\cdot T_k(x) - T_{k-1}(x)\\
&= 2x\cdot\frac{1}{2}\cdot\left(z^k+z^{-k}\right) - \frac{1}{2}\cdot\left(z^{k-1}+z^{-(k-1)}\right)\\
&= \frac{1}{2}\cdot\left(z^{k-1}(2xz-1) + z^{-(k-1)}(2xz^{-1}-1)\right)\\
&= \frac{1}{2}\cdot\left(z^{k+1}+z^{-(k+1)}\right),
\end{aligned}$$

where in the last equality, we use $2xz = z^2+1$ and $2xz^{-1} = z^{-2}+1$, which follow from $1/2\cdot(z+z^{-1}) = x$. Thus, by induction, we get that the claim holds for all d. □

Notes

It is an easy exercise to check that Theorem 2.1 is equivalent to showing that the infimum, over a monic degree-d polynomial p, of $\sup_{x\in[-1,1]} |p(x)| = 2^{-d+1}$. The minimizer is an appropriately scaled Chebyshev polynomial. An interesting question is whether we can characterize polynomials that approximate x^d up to a larger error, *e.g.*, $2^{-d/2}$.

3

Approximating Monomials

In this section, we develop a simple approximation for the function x^s on $[-1,1]$ for a positive integer s using Chebyshev polynomials. We show that x^s is approximated well by a polynomial of degree roughly $\sqrt{s}$.

Recall from Proposition 2.1 that for any d, we can write

$$x \cdot T_d(x) = \frac{1}{2} \cdot (T_{d-1}(x) + T_{d+1}(x)).$$

If we let Y be a random variable that takes values 1 and -1 with probability $1/2$ each, we can write $xT_d(x) = \mathbb{E}_Y[T_{d+Y}(x)]$. This simple observation can be iterated to obtain an expansion of the monomial x^s for any positive integer s in terms of the Chebyshev polynomials. Throughout this section, let $Y_1, Y_2, \ldots$ be i.i.d. variables taking values 1 and -1 each with probability $1/2$. For any integer $s \geq 0$, define the random variable $D_s \stackrel{\text{def}}{=} \sum_{i=1}^{s} Y_i$ where $D_0 \stackrel{\text{def}}{=} 0$.

Theorem 3.1. For any integer $s \geq 0$, and $Y_1, \ldots, Y_s$ random variables as defined above, we have, $\mathbb{E}_{Y_1,\ldots,Y_s}[T_{D_s}(x)] = x^s$.

Proof. We proceed by induction. For $s = 0$, we know $D_s = 0$ and, hence,

$\mathbb{E}[T_{D_s}(x)] = T_0(x) = 1 = x^0$. Moreover, for any $s \geq 0$,

$$\begin{aligned} x^{s+1} \overset{\text{Induction}}{=} x \cdot \mathop{\mathbb{E}}_{Y_1,\ldots,Y_s} T_{D_s}(x) &= \mathop{\mathbb{E}}_{Y_1,\ldots,Y_s}[x \cdot T_{D_s}(x)] \\ &\overset{\text{Prop. 2.1}}{=} \mathop{\mathbb{E}}_{Y_1,\ldots,Y_s}\left[\frac{T_{D_s+1}(x) + T_{D_s-1}(x)}{2}\right] \\ &= \mathop{\mathbb{E}}_{Y_1,\ldots,Y_s,Y_{s+1}}[T_{D_{s+1}}(x)]. \end{aligned}$$

□

Theorem 3.1 allows us to obtain polynomials that approximate x^s, but have degree close to $\sqrt{s}$. The main observation is that Chernoff bounds imply that the probability that $|D_s| \gg \sqrt{s}$ is small. We now state a version of Chernoff bounds that we need (see [40, Chapter 4]).

Theorem 3.2 (Chernoff Bound)**.** For independent random variables $Y_1, \ldots, Y_s$ such that $\mathbb{P}[Y_i = 1] = \mathbb{P}[Y_i = -1] = 1/2$ and for any $a \geq 0$, we have

$$\mathbb{P}\left[\sum_{i=1}^{s} Y_i \geq a\right] = \mathbb{P}\left[\sum_{i=1}^{s} Y_i \leq -a\right] \leq e^{-a^2/2s}.$$

The above theorem implies that the probability that $|D_s| > \sqrt{2s\log(2/\delta)} \overset{\text{def}}{=} \widehat{d}$ is at most δ. Moreover, since $|T_{D_s}(x)| \leq 1$ for all $x \in [-1,1]$, we can ignore all terms with degree greater than $\widehat{d}$ without incurring an error greater than δ.

We now prove this formally. Let $\mathbb{1}_{|D_s| \leq d}$ denote the indicator variable for the event that $|D_s| \leq d$. Our polynomial of degree d approximating x^s is obtained by truncating the above expansion to degree d, *i.e.*,

$$p_{s,d}(x) \overset{\text{def}}{=} \mathop{\mathbb{E}}_{Y_1,\ldots,Y_s}\left[T_{D_s}(x) \cdot \mathbb{1}_{|D_s| \leq d}\right]. \tag{3.1}$$

Since $T_{D_s}(x)$ is a polynomial of degree $|D_s|$, and the indicator variable $\mathbb{1}_{|D_s| \leq d}$ is zero whenever $|D_s| > d$, we obtain that $p_{s,d}$ is a polynomial of degree at most d.

Theorem 3.3. For any positive integers s and d, the degree-d polynomial $p_{s,d}$ defined by Equation (3.1) satisfies

$$\sup_{x \in [-1,1]} |p_{s,d}(x) - x^s| \leq 2e^{-d^2/2s}.$$

Hence, for any $\delta > 0$, and $d \geq \left\lceil \sqrt{2s \log(2/\delta)} \right\rceil$, we have $\sup_{x \in [-1,1]} |p_{s,d}(x) - x^s| \leq \delta$.

This theorem allows us to obtain polynomials approximating x^s with degree roughly $\sqrt{s}$.

Proof. Using Theorem 3.2, we know that

$$\mathop{\mathbb{E}}_{Y_1,\ldots,Y_s}\left[\mathbb{1}_{|D_s|>d}\right] = \mathop{\mathbb{P}}_{Y_1,\ldots,Y_s}[|D_s| > d] = \mathop{\mathbb{P}}_{Y_1,\ldots,Y_s}\left[\left|\sum_{i=1}^{s} Y_i\right| > d\right] \leq 2e^{-d^2/2s}.$$

Now, we can bound the error in approximating x^s using $p_{s,d}$.

$$\begin{aligned}
\sup_{x\in[-1,1]} |p_{s,d}(x) - x^s| &\overset{\text{Thm. 3.1}}{=} \sup_{x\in[-1,1]} \left| \mathop{\mathbb{E}}_{Y_1,\ldots,Y_s}\left[T_{D_s}(x) \cdot \mathbb{1}_{|D_s|>d}\right] \right| \\
&\leq \sup_{x\in[-1,1]} \mathop{\mathbb{E}}_{Y_1,\ldots,Y_s}\left[\left|T_{D_s}(x)\right| \cdot \mathbb{1}_{|D_s|>d}\right] \\
&\leq \mathop{\mathbb{E}}_{Y_1,\ldots,Y_s}\left[\mathbb{1}_{|D_s|>d} \cdot \sup_{x\in[-1,1]} \left|T_{D_s}(x)\right|\right] \\
&\overset{\text{Prop. 2.3}}{\leq} \mathop{\mathbb{E}}_{Y_1,\ldots,Y_s}\left[\mathbb{1}_{|D_s|>d}\right] \leq 2e^{-d^2/2s},
\end{aligned}$$

which is smaller than δ for $d \geq \left\lceil \sqrt{2s \log(2/\delta)} \right\rceil$. □

Over the next several sections, we explore several interesting consequences of this seemingly simple approximation. In Section 4, we use this approximation to give improved polynomial approximations to the exponential function. In Sections 9 and 10, we use it to give fast algorithms for solving linear systems and computing eigenvalues. We prove that the $\sqrt{s}$ dependence is optimal in Section 5.

Notes

In this section we developed polynomial approximations to x^s over the interval $[-1,1]$. We now point out how such a result can be lifted to a different

interval. Suppose a degree-d polynomial $p(x)$ approximates $f(x)$ on the interval $[a,b]$ up to error δ. Applying the transformation $x \stackrel{\text{def}}{=} \alpha z + \beta$, with $\alpha \neq 0$, we obtain

$$\delta \geq \sup_{x \in [a,b]} |f(x) - p(x)| = \sup_{z \in \left[(a-\beta)/\alpha, (b-\beta)/\alpha\right]} |f(\alpha z + \beta) - p(\alpha z + \beta)|.$$

Hence, the degree-d polynomial $p(\alpha z + \beta)$ approximates the function $f(\alpha z + \beta)$ on the interval $[(a-\beta)/\alpha, (b-\beta)/\alpha]$ up to an error of δ. In order to map the interval $[a,b]$ to $[c,d]$, we choose $\alpha \stackrel{\text{def}}{=} (a-b)/(c-d)$ and $\beta \stackrel{\text{def}}{=} (bc-ad)/(c-d)$.

Combining such linear transformations with the polynomials $p_{s,d}(x)$ approximating x^s as described in this section, we can construct good approximations for x^s over other symmetric intervals: *e.g.*, applying the transformation $x \stackrel{\text{def}}{=} z/2$ to the main result from this section, we obtain that for $d \geq \left\lceil \sqrt{2s \log(2/\delta)} \right\rceil$, the polynomial $p_{s,d}(z/2)$ approximates $(z/2)^s$ on $[-2,2]$ up to an error of δ, or equivalently, $2^s \cdot p_{s,d}(z/2)$ approximates z^s on $[-2,2]$ up to an error of $2^s \cdot \delta$.

4

Approximating the Exponential

In this section we construct polynomial approximations for the exponential function using approximations for x^s developed in Section 3.

We consider the problem of approximating the exponential function e^x. This is a natural and fundamental function and approximations to it are important in theory and practice. For applications to be introduced later in the monograph, we will be interested in approximations to e^x over intervals of the form $[-b,0]$ for $b \geq 0$. This is equivalent to approximating e^{-x} on the interval $[0,b]$ for $b \geq 0$.

A simple approach to approximating e^{-x} on the interval $[0,b]$ is to truncate the Taylor series expansion of e^{-x}. It is easy to show that using roughly $b + \log 1/\delta$ terms in the expansion suffices to obtain a δ-approximation. We prove the following theorem that provides a quadratic improvement over this simple approximation.

Theorem 4.1. For every $b > 0$, and $0 < \delta \leq 1$, there exists a polynomial

$r_{b,\delta}$ that satisfies

$$\sup_{x\in[0,b]} |e^{-x} - r_{b,\delta}(x)| \leq \delta,$$

and has degree $O\left(\sqrt{\max\{b, \log 1/\delta\} \cdot \log 1/\delta}\right)$.

Equivalently, the above theorem states that the polynomial $r_{b,\delta}(-x)$ approximates e^x up to δ on the interval $[-b, 0]$. As discussed in the notes at the end of the previous section, we can apply linear transformations to obtain approximations over other intervals. For example, the above theorem implies that the polynomial $r_{b-a,\delta}(-x+b)$ approximates e^{x-b} on $[a,b]$ up to error δ for $b > 1$, or equivalently, the polynomial $e^b \cdot r_{b-a,\delta}(-x+b)$ approximates e^x on $[a,b]$ up to error $e^b \cdot \delta$.

For a proof to the above theorem, we move from the interval $[0,b]$ to the familiar interval $[-1,1]$ via a linear transformation. After the transformation, it suffices to approximate the function $e^{-\lambda x-\lambda}$ over the interval $[-1,1]$, where $\lambda = b/2$ (for completeness, a proof is given at the end of this section).

We now outline the strategy for constructing the approximating polynomials for $e^{-\lambda x-\lambda}$ over $[-1,1]$. As mentioned before, if we truncate the Taylor expansion of $e^{-\lambda x-\lambda}$, we obtain

$$e^{-\lambda} \sum_{i=0}^{t} \frac{(-\lambda)^i}{i!} x^i$$

as a candidate approximating polynomial. The candidate polynomial is obtained by a general strategy that approximates each monomial x^i in this truncated series by the polynomial $p_{i,d}$ from Section 3. Formally,

$$q_{\lambda,t,d}(x) \stackrel{\text{def}}{=} e^{-\lambda} \sum_{i=0}^{t} \frac{(-\lambda)^i}{i!} p_{i,d}(x).$$

Since $p_{i,d}(x)$ is a polynomial of degree at most d, the polynomial $q_{\lambda,t,d}(x)$ is also of degree at most d. We now prove that for d roughly $\sqrt{\lambda}$, the polynomial $q_{\lambda,t,d}(x)$ gives a good approximation to $e^{-\lambda x}$ (for an appropriate choice of t).

Lemma 4.2. For every $\lambda > 0$ and $\delta \in (0, 1/2]$, we can choose $t = O(\max\{\lambda, \log 1/\delta\})$ and $d = O\left(\sqrt{t \log 1/\delta}\right)$ such that the polynomial $q_{\lambda,t,d}$

defined above, δ-approximates the function $e^{-\lambda-\lambda x}$ over the interval $[-1,1]$, *i.e.*,

$$\sup_{x\in[-1,1]}\left|e^{-\lambda-\lambda x}-q_{\lambda,t,d}(x)\right|\leq\delta.$$

Proof. We first expand the function $e^{-\lambda-\lambda x}$ via its Taylor series expansion around 0, and then split it into two parts, one containing terms with degree at most t, and the remainder.

$$\begin{aligned}
&\sup_{x\in[-1,1]}\left|e^{-\lambda-\lambda x}-q_{\lambda,t,d}(x)\right| \\
&\qquad\leq \sup_{x\in[-1,1]}\left|e^{-\lambda}\sum_{i=0}^{t}\frac{(-\lambda)^i}{i!}(x^i-p_{i,d}(x))\right|+\sup_{x\in[-1,1]}\left|e^{-\lambda}\sum_{i=t+1}^{\infty}\frac{(-\lambda)^i}{i!}x^i\right| \\
&\qquad\leq e^{-\lambda}\sum_{i=0}^{t}\frac{\lambda^i}{i!}\sup_{x\in[-1,1]}\left|x^i-p_{i,d}(x)\right|+e^{-\lambda}\sum_{i=t+1}^{\infty}\frac{\lambda^i}{i!}.
\end{aligned}$$

From Theorem 3.3, we know that $p_{i,d}$ is a good approximation to x^i, and we can use it to bound the first error term.

$$\begin{aligned}
e^{-\lambda}\sum_{i=0}^{t}\frac{\lambda^i}{i!}\sup_{x\in[-1,1]}\left|x^i-p_{i,d}(x)\right| &\leq e^{-\lambda}\sum_{i=0}^{t}\frac{\lambda^i}{i!}\cdot 2e^{-d^2/2i} \\
&\leq 2e^{-d^2/2t}\cdot e^{-\lambda}\sum_{i=0}^{\infty}\frac{\lambda^i}{i!} \\
&= 2e^{-d^2/2t}.
\end{aligned}$$

For the second term, we use the lower bound $i!\geq (i/e)^i$, and assume that $t\geq\lambda e^2$ to obtain

$$e^{-\lambda}\sum_{i=t+1}^{\infty}\frac{\lambda^i}{i!}\leq e^{-\lambda}\sum_{i=t+1}^{\infty}\left(\frac{\lambda e}{i}\right)^i\leq e^{-\lambda}\sum_{i=t+1}^{\infty}e^{-i}\leq e^{-\lambda-t}.$$

Thus, if $t=\left\lceil\max\{\lambda e^2,\log^2\!/\delta\}\right\rceil$ and $d=\left\lceil\sqrt{2t\log^4\!/\delta}\right\rceil$, combining the above and using $\lambda>0$, we obtain

$$\sup_{x\in[-1,1]}\left|e^{-\lambda-\lambda x}-q_{\lambda,t,d}(x)\right|\leq 2e^{-d^2/2t}+e^{-\lambda-t}\leq\frac{\delta}{2}+\frac{\delta}{2}\leq\delta.$$

□

Now, we can complete the proof of Theorem 4.1.

Proof. (of Theorem 4.1) Let $\lambda \stackrel{\text{def}}{=} b/2$, and let t and d be given by Lemma 4.2 for the given value of δ. Define $r_{b,\delta} \stackrel{\text{def}}{=} q_{\lambda,t,d}\left(1/\lambda \cdot (x-\lambda)\right)$, where $q_{\lambda,t,d}$ is the polynomial given by Lemma 4.2. Then,

$$\begin{aligned}\sup_{x\in[0,b]} \left|e^{-x} - r_{b,\delta}(x)\right| &= \sup_{x\in[0,b]} \left|e^{-x} - q_{\lambda,t,d}\left(1/\lambda \cdot (x-b/2)\right)\right| \\ &= \sup_{z\in[-1,1]} \left|e^{-\lambda z-\lambda} - q_{\lambda,t,d}(z)\right| \leq \delta,\end{aligned}$$

where the last inequality follows from the guarantee of Lemma 4.2. The degree of $r_{b,\delta}(x)$ is the same as that of $q_{\lambda,t,d}(x)$, *i.e.*, $d = O\left(\sqrt{\max\{b, \log 1/\delta\} \cdot \log 1/\delta}\right)$. □

Notes

A weaker version of Theorem 4.1 was proved by Orecchia, Sachdeva, and Vishnoi in [49]. A similar result is also implicit in a paper by Hochbruck and Lubich [25]. The approach used in this section can be extended in a straightforward manner to construct improved polynomial approximations for other functions. Roughly, a quadratic improvement in the degree can be obtained if the Taylor series of the function converges rapidly enough.

5

Lower Bounds for Polynomial Approximations

In this section we prove that the polynomial approximations obtained in the last couple of sections are essentially optimal. Specifically, we show that polynomial approximations to x^s on $[-1,1]$ require degree $\Omega(\sqrt{s})$, and that polynomials approximations to e^{-x} on $[0,b]$ require degree $\Omega(\sqrt{b})$. These bounds are derived from a special case of Markov's theorem, which we also include.

A useful tool for proving lower bounds on the degree of approximating polynomials is the following well known theorem by Markov.

Theorem 5.1 (Markov's Theorem; see [16])**.** Let p be a degree-d polynomial such that $|p(x)| \leq 1$ for any $x \in [-1,1]$. Then $p^{(1)}$, the derivative of p, satisfies $|p^{(1)}(x)| \leq d^2$ for all $x \in [-1,1]$.

In fact, the above theorem is another example of an extremal property of the Chebyshev polynomials since they are a tight example for this theorem. This theorem also generalizes to higher derivatives, where, if $p^{(k)}$ denotes the k^{th} derivative of p, it states that for any p as in Theorem 5.1, we have

$$|p^{(k)}(x)| \leq \sup_{y \in [-1,1]} |T_d^{(k)}(y)|,$$

for all k and $x \in [-1,1]$. This was proved by V. A. Markov [39]; see [54, Section 1.2] for a proof.

At the end of this section, we sketch a proof of the following special case of Markov's theorem, based on the work by Bun and Thaler [12], which bounds the derivative only at 1 instead of the complete interval and suffices for proving our lower bounds.

Lemma 5.2. For any degree-d polynomial q such that $|q(x)| \leq 1$ for all $x \in [-1,1]$, we have $|q^{(1)}(1)| \leq d^2$.

We now present the main idea behind the lower bound proofs for x^s and e^{-x}. Say $p(x)$ is an approximating polynomial. Since $p(x)$ is a good approximation to the function of interest, the range of p must be essentially the same as that of the function. The crux of both the proofs is to show that there exists a point t in the approximation interval such that $|p^{(1)}(t)|$ is large. Once we have such a lower bound on the derivative of p, a lower bound on the degree of p follows by applying the above lemma to a polynomial q obtained by a linear transformation of the input variable that maps t to 1. In order to show the existence of a point with a large derivative, we use the fact that our function value changes by a large amount over a small interval. Since p is a good approximation, p also changes by a large amount over the same interval; the Mean value theorem then implies that there exists a point in the interval where the derivative of p is large. We now use this strategy to show that any polynomial that approximates e^{-x} on $[0,b]$ to within $1/8$ must have degree at least $\sqrt{b}/3$.

Theorem 5.3. For every $b \geq 5$ and $\delta \in (0, 1/8]$, any polynomial $p(x)$ that approximates e^{-x} uniformly over the interval $[0,b]$ up to an error of δ, must have degree at least $1/3 \cdot \sqrt{b}$.

Proof. Suppose p is a degree-d polynomial that is a uniform approximation to e^{-x} over the interval $[0,b]$ up to an error of δ. Thus, for all $x \in [0,b]$, we have

$$e^{-x} - \delta \leq p(x) \leq e^{-x} + \delta.$$

Hence, $\sup_{x\in[0,b]} p(x) \leq 1+\delta$ and $\inf_{x\in[0,b]} p(x) \geq -\delta$.

Assume that $\delta \leq 1/8$, and $b \geq 5 > 3\log_e 4$. Applying the Mean Value theorem (see [55, Chapter 5]) on the interval $[0, \log_e 4]$, we know that there

exists a $t \in [0, \log_e 4]$, such that

$$\begin{aligned} |p^{(1)}(t)| &= \frac{1}{\log_e 4} \cdot \left| p(\log_e 4) - p(0) \right| \\ &\geq \frac{1}{\log_e 4} \cdot ((1-\delta) - (e^{-\log_e 4} + \delta)) \geq \frac{1}{2\log_e 4}. \end{aligned}$$

We define the following polynomial $q(x)$ which is obtained by applying p to a linear transformation of the input x such that -1 gets mapped to b, and 1 gets mapped to t. We also apply a linear transformation to the resultant so that the range of q is contained in $[-1,1]$, over the domain $[-1,1]$.

$$q(x) \stackrel{\text{def}}{=} \frac{1}{1+2\delta} \left(2p \left(\frac{t(1+x) + b(1-x)}{2} \right) - 1 \right).$$

Since

$$p([0,b]) \subseteq [-\delta, 1+\delta],$$

we obtain $|q(x)| \leq 1$ for all $x \in [-1,1]$. Thus, using Lemma 5.2, we have $|q^{(1)}(1)| \leq d^2$. This implies that

$$|q^{(1)}(1)| = \frac{(b-t)|p^{(1)}(t)|}{(1+2\delta)} \leq d^2.$$

Plugging in the lower bound on $|p^{(1)}(t)|$ proved above and rearranging, it follows that

$$d \geq \sqrt{\frac{b-t}{2 \cdot 5/4 \cdot \log_e 4}} \geq \frac{1}{3} \cdot \sqrt{b},$$

where the last step uses $t \leq \log_e 4 \leq b/3$. □

A similar proof strategy shows the tightness of the $\sqrt{s}$ bound for approximating x^s on the interval $[-1,1]$. In this case, we show that there exists a $t \in [1 - 1/s, 1]$ such that $|p^{(1)}(t)| \geq \Omega(s)$ (assuming δ small enough). The lower bound now follows immediately by applying Lemma 5.2 to the polynomial $1/1+\delta \cdot p(tx)$. Now, we give a proof of the special case of Markov's theorem given by Lemma 5.2.

Proof. (of Lemma 5.2) If we expand the polynomial q around $x = 1$ as follows,

$$q(x) = c_0 + c_1(x-1) + \ldots + c_d(x-1)^d,$$

we have $q^{(1)}(1) = c_1$. Hence, we can express the upper bound on $q^{(1)}(1)$ as the optimum of the following linear program where the c_is are variables and there are an infinite number of constraints:

$$\max\ c_1\ \text{ s.t. }\ \left|\sum_{i=0}^{d} c_i(x-1)^i\right| \le 1 \quad \forall x \in [-1,1].$$

Since $(-c_i)_i$ is a feasible solution whenever $(c_i)_i$ is a feasible solution, it suffices to maximize c_1 instead of $|c_1|$.

Now, we relax this linear program and drop all constraints except for $x = \cos(k\pi/d)$ for integral k [1] between 0 and d:

$$\max\ c_1 \text{ s.t. } \sum_{i=0}^{d} c_i(x-1)^i \le 1 \qquad \text{for } x = \cos\left(\frac{k\pi}{d}\right) \text{ with even } k,$$
$$\sum_{i=0}^{d} c_i(x-1)^i \ge -1 \qquad \text{for } x = \cos\left(\frac{k\pi}{d}\right) \text{ with odd } k.$$

It suffices to show that the optimum of this linear program is bounded above by d^2. We show this by constructing a feasible solution to its dual program and write the dual as follows:

$$\min \sum_{i=0}^{d} y_i \ \text{ s.t. } \ Ay = e_1 \ \text{ and } \ y_j \ge 0 \ \ \forall j.$$

Here $e_1 \in \mathbb{R}^{d+1}$ is the vector $(0,1,0,\ldots,0)^\top$, and A is the matrix defined by

$$A_{ij} \stackrel{\text{def}}{=} (-1)^j \left(\cos\left(\frac{j\pi}{d}\right) - 1\right)^i,$$

where $i = 0,\ldots,d$, and $j = 0,\ldots,d$. One can show that

$$y = \left(\frac{2d^2+1}{6}, \csc^2\frac{\pi}{2d}, \csc^2\frac{\pi}{d}, \ldots, \csc^2\frac{(d-1)\pi}{2d}, \frac{1}{2}\right)^\top$$

is, in fact, the unique solution to $Ay = e_1$, and satisfies $\sum y_i = d^2$. The proof requires the following elementary trigonometric identities (see [12]):

$$\sum_{j=0}^{d} (-1)^j \sin^{2i}\left(\frac{j\pi}{2d}\right) = \frac{1}{2}\cdot(-1)^d,$$

[1] Though these particular values seem magical, they are exactly the extremal points of the Chebyshev polynomial $T_d(x)$, which is known to be a tight example for Markov's theorem.

$$\sum_{j=1}^{d-1} \csc^2\left(\frac{j\pi}{2d}\right) = \frac{1}{6}\cdot(4d^2-4) \text{ and}$$

$$\sum_{j=1}^{d-1} (-1)^j \csc^2\left(\frac{j\pi}{2d}\right) = -\frac{d^2}{3} - \frac{1}{6} - \frac{1}{2}(-1)^d,$$

where the first equality holds for $2i < d$. Trivially, y satisfies the positivity constraints and, by weak duality, implies an upper bound of d^2 on the optimum value of the primal linear program. □

Notes

In the paper by Bun and Thaler [12], the authors also prove a lemma similar to Lemma 5.2 that bounds the derivative of the polynomial at 0. The two lemmas together can be used to obtain a proof of Markov's theorem (where the upper bound is tight up to a constant).

The lower bounds presented in this section do not depend on the approximation error δ. An interesting question here is to tighten the lower bound presented in this section to also incorporate a dependence on δ. For instance, for approximating the function e^{-x} on $[0,b]$, can we match the upper bound given by Theorem 4.1?

6

Approximating the Exponential using Rational Functions

In this section we highlight the power of rational functions by showing that there exist rational functions of the form $1/p(x)$, where p is a low degree polynomial, that approximate e^{-x} over $[0,\infty)$, up to an approximation error that decays exponentially with the degree of the approximation. We also show that no rational approximation of the form $1/p(x)$ can do much better.

6.1 Upper Bound

In the last section, we showed that the partial sums of the Taylor series expansion of e^{-x} require a large degree in order to provide a good approximation over a large interval. We now show that if we instead truncate the Taylor series expansion of $e^x = 1/e^{-x}$ to degree d and take its reciprocal, we can approximate e^{-x} on $[0,\infty)$ up to $2^{-\Omega(d)}$ error. We let

$$S_d(x) \stackrel{\text{def}}{=} \sum_{k=0}^{d} \frac{x^k}{k!}.$$

Theorem 6.1. For all integers $d \geq 0$,

$$\sup_{x\in[0,\infty)} \left| \frac{1}{S_d(x)} - e^{-x} \right| \leq 2^{-\Omega(d)}.$$

Hence, for any $\delta > 0$, we have a rational function of degree $O(\log 1/\delta)$ that is a δ-approximation to e^{-x}.

Proof. First, observe that for all d, and all $x \in [0,\infty)$, we have $S_d(x) \leq e^x$ and, hence, $1/S_d(x) - e^{-x} \geq 0$. We divide $[0,\infty)$ into three intervals:

$$\left[0, \frac{d+1}{3}\right), \left[\frac{d+1}{3}, \frac{2(d+1)}{3}\right), \text{ and } \left[\frac{2(d+1)}{3}, \infty\right),$$

and show a bound on the approximation error on each of these intervals. If $x \geq 2(d+1)/3$, both the terms are exponentially small. Using $S_d(x) \geq x^d/d!$ and $d! \leq (\frac{d+1}{2})^d$, we obtain

$$\forall x \in \left[\tfrac{2(d+1)}{3}, \infty\right), \qquad \left| \frac{1}{S_d(x)} - e^{-x} \right| \leq \frac{1}{S_d(x)} \leq \frac{d!}{x^d} \leq \left(\frac{d+1}{2x}\right)^d \leq \left(\frac{3}{4}\right)^d = 2^{-\Omega(d)},$$

Now, assume that $x < 2(d+1)/3$. We have,

$$\begin{aligned} \left| \frac{1}{S_d(x)} - e^{-x} \right| &= \frac{e^{-x}}{S_d(x)} \left(\frac{x^{d+1}}{(d+1)!} + \frac{x^{d+2}}{(d+2)!} + \dots \right) \\ &\leq \frac{e^{-x}}{S_d(x)} \cdot \frac{x^{d+1}}{(d+1)!} \left(1 + \frac{x}{d+1} + \frac{x^2}{(d+1)^2} + \dots \right) \\ &\leq 3 \frac{e^{-x}}{S_d(x)} \cdot \frac{x^{d+1}}{(d+1)!}. \end{aligned} \tag{6.1}$$

If $x \in [d+1/3, 2(d+1)/3)$, we use that e^{-x} is exponentially small, and show that the numerator is not much larger than $S_d(x)$. We use $S_d(x) \geq x^d/d!$ in (6.1) to obtain

$$\forall x \in \left[\tfrac{d+1}{3}, \tfrac{2(d+1)}{3}\right), \qquad \left| \frac{1}{S_d(x)} - e^{-x} \right| \leq 3e^{-\frac{d+1}{3}} \cdot \frac{x}{d+1} \leq 2e^{-d/3} = 2^{-\Omega(d)}.$$

Finally, if $x < {}^{(d+1)}/_{3}$, we use that $S_d(x)$ is an exponentially good approximation of e^x in this range. Using $(d+1)! \geq ({}^{(d+1)}/_{e})^{d+1}$ and $S_d(x) \geq 1$ in (6.1) to obtain

$$\forall x \in \left[0, \tfrac{d+1}{3}\right), \qquad \left|\frac{1}{S_d(x)} - e^{-x}\right| \leq 3\left(\frac{xe}{d+1}\right)^{d+1} \leq 3\left(\frac{e}{3}\right)^{d+1} = 2^{-\Omega(d)}.$$

□

A more careful argument by Cody, Meinardus, and Varga [19] shows that, in fact, ${}^{1}/_{S_d(x)}$ approximates e^{-x} up to an error of 2^{-d}.

6.2 Lower Bound

We now show that polynomials cannot do much better. We give a simple proof that shows that for any rational function of the form ${}^{1}/_{p_d(x)}$ which approximates e^{-x} on $[0,\infty)$ where $p_d(x)$ is a degree-d polynomial, the error cannot decay faster than exponentially in the degree.

Theorem 6.2. For every degree-d polynomial $p_d(x)$ with d large enough, $\sup_{x\in[0,\infty)} \left|e^{-x} - {}^{1}/_{p_d(x)}\right| \geq 50^{-d}$.

Proof. Assume, on the contrary, that for some large enough d there exists a degree-d polynomial $p_d(x)$ such that ${}^{1}/_{p_d(x)}$ approximates e^{-x} up to an error of 50^{-d} on $[0,\infty)$. Thus, for all $x \in [0,d]$, we have

$$\frac{1}{p_d(x)} \geq e^{-d} - 50^{-d} \geq \frac{1}{2}\cdot e^{-d},$$

i.e., $|p_d(x)| \leq 2e^d$. Hence, the degree-d polynomial ${}^{1}/_{2}\cdot e^{-d}\cdot p_d\left({}^{d}/_{2} + {}^{d}/_{2}\cdot y\right)$ is bounded by 1 in absolute value over the interval $[-1,1]$. Using Proposition 2.4, which implies that the Chebyshev polynomials have the fastest growth amongst such polynomials, we obtain

$$\left|\frac{1}{2}\cdot e^{-d}\cdot p_d\left(\frac{d}{2} + \frac{d}{2}\cdot y\right)\right| \leq |T_d(y)|.$$

Using the closed-form expression for Chebyshev polynomials given in Proposition 2.5, we have

$$\forall y \text{ s.t. } |y| \geq 1, \quad T_d(y) = \frac{1}{2}\left(y + \sqrt{y^2-1}\right)^d + \frac{1}{2}\left(y - \sqrt{y^2-1}\right)^d.$$

Thus, for $y = 7$, we have

$$p_d(4d) \leq 2e^d \cdot T_d(7) \leq 2e^d \cdot 14^d.$$

This implies that for $x = 4d$, we obtain

$$\left| e^{-x} - \frac{1}{p_d(x)} \right| \geq \frac{1}{p_d(x)} - e^{-x} \geq \frac{1}{2}(14e)^{-d} - e^{-4d},$$

which is larger than 50^{-d} for d large enough. This contradicts the assumption that $1/p_d(x)$ approximates e^{-x} for all $x \in [0,\infty)$ up to an error of 50^{-d}. □

Notes

In this section we constructed a rational approximation for the exponential function by taking the inverse of the truncation of a Taylor expansion of the inverse of the exponential function. This approach can be applied to other functions as well. Newman [45] used this approach to construct a rational approximation to x^s for $x \in [0,1]$. Specifically, he chose the Taylor expansion of the function x^{-s} around 1 (instead of 0), and truncated it to degree d, obtaining a candidate approximation for x^s of the form $1/q(x)$ where $q(x) = \sum_{i=0}^{d} \binom{s+i-1}{i}(1-x)^i$. Newman [45] proved that for all $x \in [0,1]$, we have $\left|1/q(x) - x^s\right| \leq 2/d \cdot \left((2s-2)/(2s+d)\right)^{s-1}$, implying that $d = \Theta(\log 1/\delta)$ suffices for achieving error δ.

The exact rate of decay of the best approximation for e^{-x} using rational functions was a central problem in approximation theory for more than 15 years. Cody, Meinardus, and Varga [19] were the first to prove a lower bound of $6^{-d+o(d)}$ for rational functions of the form $1/p_d(x)$ where p_d is a degree-d polynomial. Schönhage [60] proved that the best approximation of the form $1/p_d(x)$ achieves an approximation error of $3^{-d+o(d)}$. Newman [44] showed that even for an arbitrary degree-d rational function, *i.e.*, $p_d(x)/q_d(x)$ approximating e^{-x}, where both $p_d(x)$ and $q_d(x)$ are polynomials of degree at most

d, the approximation error cannot be smaller than 1280^{-d}. The question was settled by Gonchar and Rakhmanov [20] who finally proved that the smallest approximation error achieved by arbitrary degree-d rational functions is $c^{-d(1+o(1))}$, where c is the solution to an equation involving elliptic integrals.

7

Approximating the Exponential using Rational Functions with Negative Poles

Motivated by applications to speeding up certain algorithms, in this section we study rational approximations for e^{-x} with negative poles and approximation error that decays exponentially in the degree.

In the previous section, we constructed degree-d rational approximations for e^{-x} on $[0,\infty)$ with the approximation error decaying exponentially with d. For some applications, such as approximating the matrix exponential (discussed in Section 11) and numerical solutions to differential equations, it is desirable to have approximations that satisfy an additional condition: all their poles (*i.e.*, roots of the denominator) are negative (see [19, 59]). Further, such rational approximations have been used, in combination with powerful Laplacian solvers, to design near-linear time algorithms that compute approximations to $\exp(-L)v$ when L is a graph Laplacian; see Section 11. Approximations to e^{-x} by rational functions with negative poles were first studied by Saff, Schönhage, and Varga [59], who showed that there exist such approximations which still have the property that the approximation error decreases exponentially with the degree. In this section, we present a proof of their result.

We start by arguing why the rational approximation for e^{-x} constructed in the previous section, namely $1/S_d(x)$, does not already give us the desired result. Towards this, we need to understand the zeros of $S_d(x)$ which have been well studied (see [68] for a survey). It is fairly simple to show that $S_d(x)$ has exactly one real zero $x_d \in [-d,-1]$ if d is odd, and no real zeros if d is even. It is also known that the zeros of $S_d(x)$ grow linearly in magnitude with d. In fact, it was proved by Szegö [63] that if all the (complex) zeros of S_d are scaled down by d, as d goes to infinity they converge to a point on the curve $|ze^{1-z}| = 1$ on the complex plane. Thus, we can rule out the approximation $1/S_d(x)$.

How about the approximation $(1+x/d)^{-d}$? Trivially, it is a simple rational function where the denominator has only negative zeros, and converges to e^{-x} uniformly over $[0,\infty)$. However, the convergence rate of this approximation is slow with respect to d and it is easily seen that the approximation error at $x = 1$ is $\Theta(1/d)$. Saff, Schönhage, and Varga [59] showed that for every rational function of the form $1/p_d(x)$, where p_d is a degree-d polynomial with real roots,

$$\sup_{x\in[0,\infty)} |e^{-x} - 1/p_d(x)| = \Omega(1/d^2).$$

Surprisingly, the authors of [59] showed that rational functions of the form $p_d\left(\frac{1}{1+x/d}\right)$ can approximate e^{-x} up to $O(d2^{-d})$ for some degree-d polynomial $p_d(\cdot)$; see also [4]. Formally, the authors of [59] proved the following.

Theorem 7.1. For every d, there exists a degree-d polynomial p_d such that,

$$\sup_{x\in[0,\infty)} \left| e^{-x} - p_d\left(\frac{1}{1+x/d}\right)\right| \le O(d\cdot 2^{-d}).$$

Moreover, the coefficients of p_d are bounded by $d^{O(d)}$, and can be approximated up to an error of $d^{-\Theta(d)}$ using $\mathrm{poly}(d)$ arithmetic operations, where all intermediate numbers can be expressed using $\mathrm{poly}(d)$ bits.

The proof starts with the variable transformation

$$y \stackrel{\text{def}}{=} 1 - 2(1+x/d)^{-1}$$

which maps the infinite interval $[0,\infty)$ to a finite one, namely to $[-1,1]$. Thus, e^{-x} can be written as the following function of y:

$$f_d(y) \stackrel{\text{def}}{=} \exp\left(-d\cdot(1+y)/(1-y)\right) = e^{-x}.$$

We define $f_d(1) = 0$ and observe that y now varies over the interval $[-1,1]$. Thus, if we find a degree-d polynomial which is δ-close to $f_d(y)$ throughout the interval $[-1,1]$, then transforming back to x, we obtain a rational function of degree-d that δ-approximates e^{-x} over $[0,\infty)$.

One approach to finding such a polynomial would be to use the approximation obtained from truncating the Taylor series for $f_d(y)$. Concretely, start with the polynomial $Q_d(y)$ obtained by truncating, up to degree d, the Taylor series expansion of the function $f_1(y) = \exp\left(-{}^{(1+y)}/_{(1-y)}\right)$ around $y = -1$. Then, consider the degree-d^2 polynomial $Q_d^d(y)$. This polynomial can be shown to provide us with a $2^{-\Theta(d)}$-approximation (with degree d^2 instead of d) for the function $f_d(y)$. However, the proof of this is complicated, and it is not clear how to improve this approach to achieve a similar approximation using a degree-d rational function, as in Theorem 7.1.

In the rest of this section, we present a simplification of the proof of Theorem 7.1 from [59]. The proof, again, considers $f_d(y)$ and constructs a degree-d polynomial which achieves the desired approximation. Unlike the approach in the previous paragraph, this polynomial is chosen to be an optimizer to an ℓ_2-minimization problem as follows. We start by arguing that in order to find a good degree-d *uniform* approximation to $f_d(y)$, it is sufficient to solve an ℓ_1-optimization problem. Since $f_d(1) = 0$, for any polynomial $q(y)$ which satisfies $q(1) = 0$,[1] we can write the error at a point y as

$$|f_d(y) - q(y)| = \left| \int_y^1 (f_d^{(1)}(t) - q^{(1)}(t))\,\mathrm{d}t \right|,$$

where $f_d^{(1)}$ and $q^{(1)}$ denote the respective derivatives. Applying the triangle inequality, we can bound the above by $\int_{-1}^1 |f_d^{(1)}(t) - q^{(1)}(t)|\,\mathrm{d}t$ for all $y \in [-1,1]$. Since

$$\int_{-1}^1 |f_d^{(1)}(t) - q^{(1)}(t)|\,\mathrm{d}t \leq \sqrt{2}\sqrt{\int_{-1}^1 \left(f_d^{(1)}(t) - q^{(1)}(t)\right)^2 \mathrm{d}t}$$

by the Cauchy-Schwarz inequality, we can further reduce our task to the ℓ_2-

[1] Observe that this assumption can result in the approximation error increasing by at most a factor of 2.

problem of finding the degree-$(d-1)$ polynomial $r(t)$ which minimizes

$$\int_{-1}^{1} \left(f_d^{(1)}(t) - r(t) \right)^2 \mathrm{d}t.$$

The minimizer for this problem can be characterized exactly in terms of Legendre polynomials. Recall (from the notes at the end of Section 1) that the (normalized) Legendre polynomials, namely, $\sqrt{(2k+1)/2} \cdot L_k(x)$, are orthonormal for $k = 0, 1, \ldots$ with respect to the uniform weight on $[-1, 1]$. Thus, the ℓ_2-error of the minimizer is exactly $\sum_{k=d}^{\infty} (2k+1)\gamma_k^2$, where $\gamma_k \stackrel{\text{def}}{=} \int_{-1}^{1} f_d^{(1)}(t) L_k(t)\,\mathrm{d}t$ is the inner product of $f_d^{(1)}$ with L_k.

The remainder of the proof consists of algebraic manipulations to show that $\sum_{k=d}^{\infty} (2k+1)\gamma_k^2$ is of the order of $d^2 \cdot 4^{-d}$, giving us the theorem. We also end up using properties of Laguerre polynomials, which naturally show up while understanding higher order derivatives of $f_d(y)$. The proof, in all its detail, is quite involved and can be skipped in the first reading.

Proof. (of Theorem 7.1) We prove the theorem in two parts. We first prove the existence of a good polynomial p_d, and then revisit the proof carefully to show that the coefficients of p_d are bounded and can be computed efficiently up to the required approximation error.

Reduction to an ℓ_2-approximation problem. Let $f_d^{(k)}$ denote the k-th derivative of f_d, *i.e.*,

$$f_d^{(k)}(t) \stackrel{\text{def}}{=} \frac{\mathrm{d}^k}{\mathrm{d}t^k} f_d(t).$$

As outlined above, we first reduce our problem to an ℓ_2-approximation problem.

$$\begin{aligned}
\inf_{q_d \in \Sigma_d} \sup_{y \in [-1,1]} \left| f_d(y) - q_d(y) \right| &\leq \inf_{r_{d-1} \in \Sigma_{d-1}} \sup_{y \in [-1,1]} \left| \int_y^1 \left(f_d^{(1)}(t) - r_{d-1}(t) \right) \mathrm{d}t \right| \\
&\leq \inf_{r_{d-1} \in \Sigma_{d-1}} \int_{-1}^{1} \left| f_d^{(1)}(t) - r_{d-1}(t) \right| \mathrm{d}t \\
&\leq \sqrt{2} \inf_{r_{d-1} \in \Sigma_{d-1}} \sqrt{\int_{-1}^{1} \left(f_d^{(1)}(t) - r_{d-1}(t) \right)^2 \mathrm{d}t}.
\end{aligned} \tag{7.1}$$

The first inequality holds if we take the infimum over all degree-d polynomials q_d, and all degree-$(d-1)$ polynomials r_{d-1}. We know how to write an explicit solution to the optimization problem in the last expression. We require orthogonal polynomials on $[-1,1]$ under the uniform (constant) weight function, which are given by Legendre polynomials

$$L_k(t) \stackrel{\text{def}}{=} \frac{1}{2^k \cdot k!} \frac{\mathrm{d}^k}{\mathrm{d}t^k} [(t^2-1)^k],$$

and satisfy

$$\int_{-1}^{1} L_i(t) L_j(t)\, \mathrm{d}t = \begin{cases} 0 & \text{if } i \neq j \\ \frac{2}{2i+1} & \text{otherwise;} \end{cases}$$

see [64]. Hence, we can write the last expression explicitly to obtain

$$\inf_{q_d \in \Sigma_d} \sup_{y \in [-1,1]} \left| f_d(y) - q_d(y) \right| \leq \sqrt{\sum_{k \geq d} (2k+1)\gamma_k^2}, \tag{7.2}$$

where, as before, the infimum is taken over all degree-d polynomials q_d, and γ_k denotes the inner product of $f_d^{(1)}$ with the k-th Legendre polynomial $\gamma_k \stackrel{\text{def}}{=} \int_{-1}^{1} f_d^{(1)}(t) L_k(t)\, \mathrm{d}t$. We now bound the coefficients γ_k.

Bounding the coefficients γ_k. Plugging in the definition of Legendre polynomials, and using integration by parts successively, we obtain

$$\begin{aligned} \gamma_k &= \frac{1}{2^k \cdot k!} \int_{-1}^{1} f_d^{(1)}(t) \frac{\mathrm{d}^k}{\mathrm{d}t^k} [(t^2-1)^k]\, \mathrm{d}t \\ &= \frac{(-1)^k}{2^k \cdot k!} \int_{-1}^{1} (t^2-1)^k f_d^{(k+1)}(t)\, \mathrm{d}t. \end{aligned} \tag{7.3}$$

If we let $v \stackrel{\text{def}}{=} \frac{2d}{(1-t)}$, we obtain, $f_d(t) = e^{d-v}$ and $f_d^{(1)}(t) = \frac{-1}{(1-t)} v e^{d-v}$. A simple induction argument generalizes this to give

$$(1-t)^{k+1} f_d^{(k+1)}(t) = -e^d \frac{\mathrm{d}^k}{\mathrm{d}v^k} [v^{k+1} e^{-v}].$$

We now invoke the generalized Laguerre polynomials of degree k which are orthogonal with respect to the weight function ve^{-v}, defined to be

$$G_k(v) \stackrel{\text{def}}{=} \frac{1}{k!} \cdot \frac{1}{ve^{-v}} \cdot \frac{\mathrm{d}^k}{\mathrm{d}v^k} [v^{k+1} e^{-v}];$$

see [64]. Hence, simplifying (7.3), we obtain

$$\gamma_k = \frac{-e^d}{2^k}\int_{-1}^{1}(t+1)^k \frac{ve^{-v}}{(1-t)}G_k(v)\,\mathrm{d}t = -e^d\int_d^\infty \left(1-\frac{d}{v}\right)^k e^{-v}G_k(v)\,\mathrm{d}v.$$

Squaring the above equality, and applying Cauchy-Schwarz, we obtain

$$\gamma_k^2 \le e^{2d}\int_d^\infty ve^{-v}(G_k(v))^2\,\mathrm{d}v \cdot \int_d^\infty \frac{1}{v}\left(1-\frac{d}{v}\right)^{2k} e^{-v}\,\mathrm{d}v.$$

Now, we use the fact that $\int_0^\infty ve^{-v}(G_k(v))^2\,\mathrm{d}v = k+1$ (see [64]), and substitute $v = d(1+z)$ to obtain

$$\gamma_k^2 \le e^d(k+1)\int_0^\infty \frac{z^{2k}}{(z+1)^{2k+1}}e^{-dz}\,\mathrm{d}z. \tag{7.4}$$

Obtaining the final error bound. Plugging this back in (7.2), we obtain

$$\left(\inf_{q_d\in\Sigma_d}\sup_{y\in[-1,1]}\left|f_d(y)-q_d(y)\right|\right)^2 \le e^d\int_0^\infty \sum_{k\ge d}(k+1)(2k+1)\frac{z^{2k}}{(z+1)^{2k+1}}e^{-dz}\,\mathrm{d}z.$$

We can sum up the series in the above equation for any $z \ge 0$ to obtain

$$\sum_{k\ge d}(k+1)(2k+1)\frac{z^{2k}}{(z+1)^{2k+1}} \lesssim \left(\frac{z}{z+1}\right)^{2d-2}(d^2+dz+z^2).$$

Here $\lesssim$ means that the inequality holds up to an absolute constant. This implies that

$$\left(\inf_{q_d\in\Sigma_d}\sup_{y\in[-1,1]}\left|f_d(y)-q_d(y)\right|\right)^2 \lesssim \int_0^\infty \left(\frac{z}{z+1}\right)^{2d-2}(d^2+dz+z^2)e^{d-dz}\,\mathrm{d}z.$$

It is a simple exercise to show that for all $z \ge 0$, the expression $e^{d-dz+z}\left(\frac{z}{z+1}\right)^{2d-2}$ is maximized for $z=1$ and, hence, this expression is bounded by $4e\cdot 4^{-d}$. Thus,

$$\left(\inf_{q_d\in\Sigma_d}\sup_{y\in[-1,1]}\left|f_d(y)-q_d(y)\right|\right)^2 \lesssim 4^{-d}\int_0^\infty (d^2+dz+z^2)e^{-z}\,\mathrm{d}z \lesssim d^2\cdot 4^{-d},$$

which concludes the proof of existence of a good p_d.

Computing the coefficients of p_d. We now show that the coefficients of p_d are bounded in magnitude, and can be computed efficiently.

It suffices to compute them to a precision of $2^{-\mathrm{poly}(d)}$ and we present the main steps. Recall that in the proof of Theorem 7.1, the polynomial $r_{d-1}(t)$ which minimizes $\int_{-1}^{1}\left(f_d^{(1)}(t)-r_{d-1}(t)\right)^2 \mathrm{d}t$ (see Equation (7.1)) is given by

$$r_{d-1}(t)=\sum_{k=0}^{d-1}\frac{\sqrt{2k+1}}{\sqrt{2}}\cdot\gamma_k\cdot L_k(t).$$

The Legendre polynomials can be written as (see [2, Chapter 22])

$$L_k(t)=2^{-k}\sum_{i=0}^{\lfloor k/2\rfloor}(-1)^i\,x^{k-2i}\cdot\binom{k}{i}\binom{2k-2i}{k}.$$

Thus, assuming we know $\{\gamma_k\}_{k=0}^{d-1}$, we can compute the coefficients of r_{d-1} in $\mathrm{poly}(d)$ operations, and the sizes of the coefficients of r_{d-1} can be $2^{O(d)}$ larger. Since $q_d(y)=\int_y^1 r_{d-1}(t)\,\mathrm{d}t$, given the coefficients of r_{d-1}, we can simply integrate in order to find the coefficients of q_d. The approximating polynomial $p_d(x)$ is given by $p_d(x)\stackrel{\mathrm{def}}{=}q_d(1-2x)$. Hence, given the coefficients of q_d, those of p_d can be calculated in $\mathrm{poly}(d)$ operations, and again can only be at most $2^{O(d)}$ larger. Hence, it suffices to show how to compute $\{\gamma_k\}_{k=0}^{d-1}$.

With the substitution $z=d(1+v)$ in Equation (7.3), we have

$$\gamma_k=-d\int_0^\infty\left(\frac{z}{z+1}\right)^k e^{-dz}G_k(d(1+z))\,\mathrm{d}z.$$

The Laguerre polynomials G_k (of order 1) are explicitly given to be

$$G_k(t)=\sum_{i=0}^{k}(-1)^i\binom{k+1}{k-i}\frac{t^i}{i!},$$

see [2, Chapter 22]. After using this expansion for G_k, it suffices to compute the integrals $\int_0^\infty\frac{z^i}{(z+1)^j}e^{-dz}\,\mathrm{d}z$ for $0\le j\le i\le d$. If we know the values of these integrals, we can compute the γ_ks in $\mathrm{poly}(d)$ operations, although the coefficients may increase by a factor of $d^{O(d)}$. For any $0\le j\le i\le d$, by substituting $w=z+1$, we obtain

$$\int_0^\infty\frac{z^i}{(z+1)^j}e^{-dz}\,\mathrm{d}z=e^d\int_1^\infty\frac{(w-1)^i}{w^j}e^{-dw}\,\mathrm{d}w.$$

Since we can expand $(w-1)^i$ using the Binomial theorem, it suffices to compute integrals of the form $\int_1^\infty w^{-j} e^{-dw}\,\mathrm{d}w$ for $-d \le j \le d$, where again we lose at most $2^{O(d)}$ in the magnitude of the numbers. For $j \le 0$, this is a simple integration. For $j \ge 1$, the integral can be expressed using the Exponential Integral. Hence, it has the following rapidly convergent power series for $d > 1$, which can be used both to compute and to bound $E_j(d)$s easily:

$$E_j(d) \stackrel{\text{def}}{=} \int_1^\infty w^{-j} e^{-dw}\,\mathrm{d}w = \frac{e^{-d}}{d} \sum_{k=0}^{\infty} \frac{(-1)^k (j+k-1)!}{(j-1)!\,d^k};$$

see [30]. Combining everything, the coefficients of p_d can be approximated up to $d^{-\Theta(d)}$ error in time $\mathrm{poly}(d)$ using $\mathrm{poly}(d)$ sized registers.

□

Notes

The main result in this section is, undoubtedly, not as transparent as we would have liked it to be. Given the importance of the result, it remains an interesting problem to determine an intuitive reasoning as to why such a result should hold. An answer may be useful in understanding to what extent the poles of a degree-d rational approximation for a given function can be restricted while fixing the error.

Part II

APPLICATIONS

Notation: Matrices and Graphs

The following sections shall primarily deal with $n \times n$ symmetric matrices over the reals. A fundamental theorem in linear algebra (see [66, Chapter 1]) asserts that every symmetric matrix $A \in \mathbb{R}^{n \times n}$ has n real eigenvalues along with eigenvectors that can be chosen to be orthogonal. Thus, A can be written as $U \Lambda U^\top$ where the columns of U are the eigenvectors of A and Λ is the diagonal matrix corresponding to its eigenvalues. Hence, we also have $U^\top U = I$. A is said to be positive semidefinite (PSD) if all its eigenvalues are non-negative and positive definite (PD) if all its eigenvalues are strictly positive. We will use the notation $A \succeq 0$ (respectively $A \succ 0$) to denote that A is PSD (respectively PD). The notation $A \succeq B$ (respectively $A \succ B$) is equivalent to $A - B \succeq 0$ (respectively $A - B \succ 0$). The spectral norm of a matrix A, also sometimes called its $2 \to 2$ norm, is defined to be $\sup_{x \neq \mathbf{0}} \frac{\|Ax\|_2}{\|x\|_2}$. Thus, all eigenvalues of A are bounded in absolute value by the spectral norm of A. For a PSD matrix, its norm is equal to its largest eigenvalue. Henceforth, $\|\cdot\|$ will be used to denote the ℓ_2 norm for vectors and the spectral norm for matrices. For a PD matrix A, its condition number $\kappa(A)$ is the ratio of the largest eigenvalue to the smallest eigenvalue. Finally, for a vector v and a PSD matrix A, we define $\|v\|_A \stackrel{\text{def}}{=} \sqrt{v^\top A v}$.

For a matrix A, and a degree-d polynomial $p(x) \stackrel{\text{def}}{=} \sum_{i=0}^{d} c_i x^i$, we define $p(A)$ to be $\sum_{i=0}^{d} c_i A^i$. Similarly for a function $f : \mathbb{R} \mapsto \mathbb{R}$ defined by the power

series $f(x) = \sum_{i \geq 0} c_i x_i$, we define $f(A)$ to be $\sum_{i \geq 0} c_i A^i$. Thus, $\exp(A)$ or e^A is $\sum_{k \geq 0} \frac{A^k}{k!}$. If A is a real symmetric matrix with the spectral decomposition $U \Lambda U^\top$, this is equivalent to the definition $f(A) \stackrel{\text{def}}{=} U f(\Lambda) U^\top$, where $f(\Lambda)$ is the diagonal matrix with the (i,i)-th entry being equal to $f(\Lambda_{(i,i)})$.

For an $n \times n$ matrix A and a vector v, often we are interested in the solution to the system of equations $Ax = v$. We only consider the case when either A is invertible or v lies in the span of the columns of A. In either case, with a slight abuse of notation, we denote the solution by $x = A^{-1}v$. The all 1s and all 0s vectors are denoted by $\mathbf{1}$ and $\mathbf{0}$ respectively.

Finally, we will work with undirected graphs $G = (V, E)$ with $n \stackrel{\text{def}}{=} |V|$ vertices and $m \stackrel{\text{def}}{=} |E|$ edges. The edges of the graph may have positive weights and this is captured by the *adjacency* matrix A of the graph; an $n \times n$ matrix where $A_{i,j}$ is the weight of the edge between i and j. We assume that the graph has no self-loops and, hence, $A_{i,i} = 0$ for all i. Since the graph is undirected, A is symmetric and has $2m$ non-zero entries. Let e_i denote the vector with a 1 in the i-th coordinate and 0s elsewhere. The matrix $L \stackrel{\text{def}}{=} \sum_{i,j} A_{i,j}(e_i - e_j)(e_i - e_j)^\top$ is called the *combinatorial Laplacian* of G. If D is the diagonal matrix with $D_{i,i} \stackrel{\text{def}}{=} \sum_{j \neq i} A_{i,j}$, then $L = D - A$. D is called the degree matrix of G. The Laplacian L of a graph G is always PSD; $L \succeq 0$.

8

Simulating Random Walks

We begin our set of applications with a simple consequence of the polynomial approximations to x^s developed in Section 3: Random walks on graphs can be simulated quadratically faster.

Consider a connected and undirected graph $G = (V, E)$ with $|V| = n, |E| = m$, and let A and D respectively denote its adjacency matrix and the diagonal matrix of degrees. The simple random walk on such a graph corresponds to the process where, starting at a vertex i, one selects a vertex j with probability proportional to its weight $A_{i,j}$, and then repeats with j as the starting vertex. Suppose we select an initial vertex from the probability distribution $v \in \mathbb{R}^n$ and perform an s-step random walk. The probability distribution of the vertex after s steps of this random walk is given by $\widetilde{W}^s v$, [1] where $\widetilde{W} \stackrel{\text{def}}{=} AD^{-1}$. It is well-known that when v is a probability distribution over V and G is connected, as s tends to infinity, $\widetilde{W}^s v$ tends to the vector where the i-th entry is

[1] The convention in the literature on Markov chains is to express the probability distribution as a row vector $v^\top$ instead, giving the probability after s steps as $v^\top \widetilde{W}^s$. We will use the column vector convention. The only resulting change is that the walk matrix is replaced by its transpose everywhere.

proportional to the degree of the i-th vertex. This limit is the vector $D\mathbf{1}$ up to scaling and is independent of the starting vector.

In this section we consider the problem of computing products of the form

$$\widetilde{W}^s v$$

where v could be an arbitrary real vector (of ℓ_2-norm 1) rather than a probability vector. This problem is not only useful in the design of fast algorithms for the *Sparsest Cut* problem (refer to the notes at the end of this section), it is also of independent interest. The straightforward algorithm for this problem, which computes $\widetilde{W}^s$ by repeatedly multiplying by $\widetilde{W}$, runs in time roughly $O(ms)$. However, in applications, s could be $\Omega(n)$ and, thus, the question we ask here is how much can this dependence on s be improved. The main result of this section shows how we can improve the dependence on s to roughly $\sqrt{s}$.

Theorem 8.1. Let $\widetilde{W}$ be the random walk matrix for a graph G with n vertices and m edges. There is an algorithm that, given any positive integer s, a unit vector v, and $\delta \in (0, 1/2]$, computes a vector w such that $\left\| \widetilde{W}^s v - w \right\| \leq \delta$ in $O\left((m+n)\sqrt{s \log 1/\delta}\right)$ arithmetic operations.

8.1 Quadratically Faster Random Walks: Proof of Theorem 8.1

As mentioned earlier, a simple way to compute $\widetilde{W}^s v$ is to multiply the matrix $\widetilde{W}$ with v a total of s times, which requires $O(ms)$ operations. We now show that, as an immediate application of the polynomial approximations to x^s that we developed in Section 3, we can approximate this distribution using roughly $\sqrt{s}$ multiplications with $\widetilde{W}$. First, we extend Theorem 3.3 to matrices.

Theorem 8.2 (Corollary to Theorem 3.3)**.** For a symmetric M with $\|M\| \leq 1$, a positive integer s, and any $\delta > 0$, define $d \stackrel{\text{def}}{=} \left\lceil \sqrt{2s \log 2/\delta} \right\rceil$. Then, the degree-$d$ polynomial $p_{s,d}(M)$, defined by (3.1) satisfies $\left\| M^s - p_{s,d}(M) \right\| \leq \delta$.

Proof. Let $\{\lambda_i\}_i$ be the eigenvalues of M with $\{u_i\}_i$ as a set of corresponding orthogonal eigenvectors. Since M is symmetric and $\|M\| \leq 1$, we have $\lambda_i \in$

$[-1,1]$ for all i. Thus, Theorem 3.3 implies that for all i, $|\lambda_i^s - p_{s,d}(\lambda_i)| \leq \delta$. Note that if λ_i is an eigenvalue of M, then $\lambda_i^s - p_{s,d}(\lambda_i)$ is the corresponding eigenvalue of $M^s - p_{s,d}(M)$ with respect to the same eigenvector. Hence, we have

$$\left\|M^s - p_{s,d}(M)\right\| = \max_i |\lambda_i^s - p_{s,d}(\lambda_i)| \leq \delta.$$

□

When we try to apply this theorem to $\widetilde{W}$ we face the obvious problem that $\widetilde{W}$ is not necessarily symmetric. This can be handled by considering the matrix $W \stackrel{\text{def}}{=} D^{-1/2}\widetilde{W}D^{1/2} = D^{-1/2}AD^{-1/2}$, which is symmetric. Thus, $\widetilde{W}^s v = D^{1/2}W^s D^{-1/2}v$. We focus on proving Theorem 8.1 for the case when G is regular. In this case $\widetilde{W} = W$. The non-regular case follows from a straightforward extension of this proof and we omit the details. Note that $\|W\| \leq 1$ since W is a doubly stochastic matrix.

Note that if we can compute the coefficients of $p_{s,d}$ efficiently, then we can quickly compute $p_{s,d}(W)v$ for $d = \left\lceil \sqrt{2s\log 2/\delta} \right\rceil$. Thus, appealing to the theorem above, we obtain an efficient δ approximation to $W^s v$, *i.e.*, $\left\|W^s v - p_{s,d}(W)v\right\| \leq \delta \|v\| \leq \delta$. In order to compute the coefficients, first observe that we do not need to explicitly compute the coefficients of the polynomial $p_{s,d}$ since we can use the expansion of $p_{s,d}$ in terms of Chebyshev polynomials as in (3.1) and the recursive definition of Chebyshev polynomials from (2.1) to compute the vectors $T_0(W)v, \ldots, T_d(W)v$ using only d multiplications with the matrix W.

The expansion of $p_{s,d}$ in terms of Chebyshev polynomials given by (3.1) implies that the non-zero coefficients are binomial coefficients up to a scaling factor. For instance, assuming that s is even, the coefficient of $T_{2j}(\cdot)$ for $j \neq 0$ is $2^{-s+1}\binom{s}{s/2+j}$. *Prima facie*, computing each of these binomial coefficients requires $O(s)$ multiplications and divisions, which is worse than the trivial $O(ms)$ time algorithm to compute $W^s v$. However, observe that since the non-zero coefficients are scaled binomial coefficients, if c_i is the coefficient of T_i, the ratios c_i/c_0 are rational numbers that we can compute explicitly and quickly. Define α to be the sum of the coefficients of $T_0(\cdot), \ldots, T_d(\cdot)$ in $p_{s,d}$,

i.e.,

$$\alpha \stackrel{\text{def}}{=} \sum_{i=0}^{d} c_i = \mathop{\mathbb{P}}_{Y_1,\ldots,Y_s}[|D_s| \leq d],$$

where we use the notation from Section 3. We know that α lies between 1 and $1-\delta$. Express c_i as,

$$c_i = \alpha \cdot \frac{c_i}{\sum_{j=0}^{d} c_j} = \alpha \cdot \frac{c_i/c_0}{\sum_{j=0}^{d} c_j/c_0}.$$

Since we can explicitly compute the ratios c_i/c_0, and hence also $\sigma \stackrel{\text{def}}{=} \sum_{i=0}^{d} c_i/c_0$, we can explicitly compute $1/\sigma \cdot c_i/c_0 = c_i/\alpha$ (which is approximately c_i). Hence, we can compute the coefficients in the Chebyshev expansion of the polynomial $\alpha^{-1} \cdot p_{s,d}(\cdot)$, and it satisfies

$$\sup_{x\in[-1,1]} \left|\alpha^{-1} \cdot p_{s,d}(x) - x^s\right| \leq \frac{\delta}{(1-\delta)} = O(\delta).$$

This completes the proof of Theorem 8.1.[2]

Notes

Theorem 8.1 can be easily generalized to a reversible irreducible Markov chain. An open problem is to obtain a version of Theorem 8.1 where the dependence on s is better than $\sqrt{s}$. (See the notes at the end of Section 11 for an approach.)

Finally, Theorem 8.1 can be used, along with the connection between random walks and sparse cuts, to show how speeding up random walks allows us to quadratically speed up finding sparse cuts in undirected graphs. For a graph $G = (V,E)$ with adjacency matrix A, a set $S \subseteq V$ is said to have sparsity

[2] An important issue we need to note is the bit length of the numbers involved. Even though it is not possible to store these numbers precisely, here we show that a small number of bits are sufficient to store these numbers. Assume that we store each of the numbers in b-bit registers. All the numbers involved in computing the ratios of successive coefficients are $O(s)$, thus we need $b = \Omega(\log s)$. Each of these ratios can be computed to an accuracy of $O(2^{-b})$, and since there are $O(d)$ multiplications/divisions involved, we can compute all of c_i/c_0 up to an accuracy of $O(d2^{-b})$. Hence, the absolute error in σ is at most $O(d^2 2^{-b})$. This implies that if $d^2 2^{-b} = O(\delta)$, the error in the estimate u is at most $O(\delta)\|v\|$. Thus, $b = \Theta(\log s/\delta)$ suffices.

or conductance

$$\phi(S) \stackrel{\text{def}}{=} \frac{\sum_{i\in S}\sum_{j\notin S}A(i,j)}{\min\left(\sum_{i\in S}\sum_{j\in V}A(i,j),\sum_{i\notin S}\sum_{j\in V}A(i,j)\right)}.$$

Thus, for a set $S \subseteq V$, $\Phi(S)$ is the ratio of the edges going across the cut $(S, V \setminus S)$, to the total number of edges incident on the vertices in S or $V \setminus S$, whichever is less. The conductance of a graph is defined to be $\phi \stackrel{\text{def}}{=} \min_{S\subseteq V}\phi(S)$. The *Sparsest Cut* problem is to compute a cut which minimizes conductance. This problem is NP-hard and is an extremely important problem in both theory and practice; see [66, Chapter 5] for a detailed discussion on this problem. A celebrated result of Cheeger [15] and Alon and Milman [3] relates the second smallest eigenvalue of the Laplacian $L \stackrel{\text{def}}{=} D - A$ of G, denoted $\lambda_2(L)$, to the conductance of the graph. Often referred to as Cheeger's inequality, the result, stated here for d-regular graphs, asserts that

$$\phi \leq O\left(\sqrt{\frac{\lambda_2}{d}}\right).$$

Let $\lambda \stackrel{\text{def}}{=} \lambda_2/d$ be the normalized *spectral gap* and $\mathcal{L} \stackrel{\text{def}}{=} \frac{1}{d}L$ be the normalized Laplacian. For a connected graph, $\lambda \in (0,2]$ and the interesting case is when λ is small. The results of this section can be used to prove the following theorem whose dependence on λ is roughly $1/\sqrt{\lambda}$ as opposed to the $1/\lambda$ that can be obtained by the Power method; see also Section 10.

Theorem 8.3. Given an undirected graph G with normalized spectral gap λ, we can find a cut of conductance $O(\sqrt{\lambda})$ with probability at least $1/3$ using $O\left(m/\sqrt{\lambda}\cdot\log n/\lambda\right)$ operations.

9

Solving Linear Equations via the Conjugate Gradient Method

In this section we discuss the problem of solving a system of linear equations. We first present the Conjugate Gradient method which works when the corresponding matrix is positive definite. We then give a simple proof of the rate of convergence of this method by using the polynomial approximations for x^s developed in Section 3.

Given a matrix $A \in \mathbb{R}^{n\times n}$ and a vector $v \in \mathbb{R}^n$, our goal is to find a vector $x \in \mathbb{R}^n$ such that $Ax = v$. The exact solution $x^\star \stackrel{\text{def}}{=} A^{-1}v$ can be computed by Gaussian elimination, but the fastest known implementation requires the same time as matrix multiplication (currently $O(n^{2.737})$). For many applications, the number of non-zero entries in A (denoted by m), or its *sparsity*, is much smaller than n^2 and, ideally, we would like linear solvers which run in time $\widetilde{O}(m)$ [1], roughly the time it takes to multiply a vector with A. While we are far from this goal for general matrices, *iterative methods*, based on techniques such as gradient descent or the Conjugate Gradient method reduce the problem of solving a system of linear equations to the computation of a small number of matrix-vector products with the matrix A when A is symmetric and

[1] The $\widetilde{O}$ notation hides polynomial factors in $\log n$.

positive definite (PD). These methods often produce only approximate solutions. However, these approximate solutions suffice for most applications. While the running time of the gradient descent-based method varies linearly with the condition number of A, that of the Conjugate Gradient method depends on the square-root of the condition number; the quadratic saving occurring precisely because there exist $\sqrt{s}$- degree polynomials approximating x^s. The guarantees of the Conjugate Gradient method are summarized in the following theorem:

Theorem 9.1. Given an $n \times n$ symmetric matrix $A \succ 0$, and a vector $v \in \mathbb{R}^n$, the Conjugate Gradient method can find a vector x such that $\|x - A^{-1}b\|_A \leq \delta \|A^{-1}b\|_A$ in time $O((t_A + n) \cdot \sqrt{\kappa(A)} \log 1/\delta)$, where t_A is the time required to multiply A with a given vector, and $\kappa(A)$ is the condition number of A.

9.1 A Gradient Descent Based Linear Solver

The gradient descent method is a general method to solve convex programs; here we only focus on its application to linear systems. The PD assumption on A allows us to formulate the problem of solving $Ax = v$ as a convex programming problem: For the function

$$f_A(x) \stackrel{\text{def}}{=} \left\|x - x^\star\right\|_A^2 = (x - x^\star)^\top A(x - x^\star) = x^\top Ax - 2x^\top v + x^{\star\top} Ax^\star,$$

find the vector x that minimizes $f_A(x)$. Since A is symmetric and PD, this is a convex function, and has a unique minimizer $x = x^\star$.

When minimizing f_A, each iteration of the gradient descent method is as follows: Start from the current estimate of $x^\star$, say x_t, and move along the direction of maximum rate of decrease of the function f_A, *i.e.*, against its gradient, to the point that minimizes the function along this line. We can explicitly compute the gradient of f_A to be $\nabla f_A(x) = 2A(x - x^\star) = 2(Ax - v)$. Thus, we have

$$x_{t+1} = x_t - \alpha_t \nabla f_A(x_t) = x_t - 2\alpha_t (Ax_t - v)$$

for some α_t. Define the *residual* $r_t \stackrel{\text{def}}{=} v - Ax_t$. Thus, we have

$$\begin{aligned} f_A(x_{t+1}) &= f_A(x_t + 2\alpha_t r_t) = (x_t - x^\star + 2\alpha_t r_t)^\top A(x_t - x^\star + 2\alpha_t r_t) \\ &= (x_t - x^\star)^\top A(x_t - x^\star) + 4\alpha_t (x_t - x^\star)^\top A r_t + 4\alpha_t^2 r_t^\top A r_t. \\ &= (x_t - x^\star)^\top A(x_t - x^\star) - 4\alpha_t r_t^\top r_t + 4\alpha_t^2 r_t^\top A r_t \end{aligned}$$

is a quadratic function in α_t. We can analytically compute the α_t that minimizes f_A and find that it is $\frac{1}{2} \cdot \frac{r_t^\top r_t}{r_t^\top A r_t}$. Substituting this value of α_t, and using $x^\star - x_t = A^{-1} r_t$, we obtain

$$\|x_{t+1} - x^\star\|_A^2 = \|x_t - x^\star\|_A^2 - \frac{(r_t^\top r_t)^2}{r_t^\top A r_t} = \|x_t - x^\star\|_A^2 \left(1 - \frac{r_t^\top r_t}{r_t^\top A r_t} \cdot \frac{r_t^\top r_t}{r_t^\top A^{-1} r_t}\right).$$

Now we relate the rate of convergence to the optimal solution to the condition number of A. Towards this, note that for any z, we have

$$z^\top A z \leq \lambda_1 z^\top z \quad \text{and} \quad z^\top A^{-1} z \leq \lambda_n^{-1} z^\top z,$$

where λ_1 and λ_n are the smallest and the largest eigenvalues of A respectively. Thus,

$$\|x_{t+1} - x^\star\|_A^2 \leq (1 - \kappa^{-1}) \|x_t - x^\star\|_A^2,$$

where $\kappa \stackrel{\text{def}}{=} \kappa(A) = \lambda_1/\lambda_n$ is the condition number of A. Hence, assuming we start with $x_0 = \mathbf{0}$, we can find an x_t such that $\|x_t - x^\star\|_A \leq \delta \|x^\star\|_A$ in approximately $\kappa \log 1/\delta$ iterations, with the cost of each iteration dominated by $O(1)$ multiplications of the matrix A with a given vector (and $O(1)$ dot product computations). Thus, this gradient descent-based method allows us to compute a δ approximate solution to $x^\star$ in time $O((t_A + n)\kappa \log 1/\delta)$.

9.2 The Conjugate Gradient Method

Observe that at any step t of the gradient descent method, we have $x_{t+1} \in \mathsf{Span}\{x_t, Ax_t, v\}$. Hence, for $x_0 = \mathbf{0}$, it follows by induction that for any positive integer k,

$$x_k \in \mathsf{Span}\{v, Av, \ldots, A^k v\}.$$

The running time of the gradient descent-based method is dominated by the time required to compute a basis for this subspace. However, this vector x_k

may not be a vector from this subspace that minimizes f_A. On the other hand, the essence of the Conjugate Gradient method is that it finds the vector in this subspace that minimizes f_A, in essentially the same amount of time required by k iterations of the gradient descent-based method. We must address two important questions about the Conjugate Gradient method: (1) Can the best vector be computed efficiently? and (2) What is the approximation guarantee achieved after k iterations? We show that the best vector can be found efficiently, and prove, using the polynomial approximations to x^k from Section 3, that the Conjugate Gradient method achieves a quadratic improvement over the gradient descent-based method in terms of its dependence on the condition number of A.

Finding the best vector efficiently. Let $\{v_0,\ldots,v_k\}$ be a basis for $\mathcal{K} \stackrel{\text{def}}{=} \mathsf{Span}\{v,Av,\ldots,A^kv\}$ (called the *Krylov subspace of order k*). Hence, any vector in this subspace can be written as $\sum_{i=0}^{k}\alpha_i v_i$. Our objective then becomes

$$\|x^\star - \sum_i \alpha_i v_i\|_A^2 = (\sum_i \alpha_i v_i)^\top A(\sum_i \alpha_i v_i) - 2(\sum_i \alpha_i v_i)^\top v + \left\|x^\star\right\|_A^2.$$

Solving this optimization problem for α_i requires matrix inversion, the very problem we set out to mitigate. The crucial observation is that if the v_is are A-orthogonal, *i.e.*, $v_i^\top A v_j = 0$ for $i \neq j$, then all the cross-terms disappear. Thus,

$$\|x^\star - \sum_i \alpha_i v_i\|_A^2 = \sum_i (\alpha_i^2 v_i^\top A v_i - 2\alpha_i v_i^\top v) + \left\|x^\star\right\|_A^2,$$

and, as in the gradient descent-based method, we can analytically compute the set of values α_i that minimize the objective to be given by $\alpha_i = \frac{v_i^\top v}{v_i^\top A v_i}$.

Hence, if we can construct an A-orthogonal basis $\{v_0,\ldots,v_k\}$ for $\mathcal{K}$ efficiently, we do at least as well as the gradient descent-based method. If we start with an arbitrary set of vectors and try to A-orthogonalize them via the Gram-Schmidt process (with inner products with respect to A), we need to compute k^2 inner products and, hence, for large k, it is not more efficient than the gradient descent-based method. An efficient construction of such a basis is one of the key ideas here. We proceed iteratively, starting with $v_0 = v$. At the i^{th} iteration, we compute Av_{i-1} and A-orthogonalize it with respect

to $v_0, \ldots, v_{i-1}$, to obtain v_i. It is trivial to see that the vectors $v_0, \ldots, v_k$ are A-orthogonal. Moreover, it is not difficult to see that for every i, we have

$$\mathsf{Span}\{v_0, \ldots, v_i\} = \mathsf{Span}\{v, Av, \ldots, A^i v\}.$$

Now, since $Av_j \in \mathsf{Span}\{v_0, \ldots, v_{j+1}\}$ for every j, and A is symmetric, A-orthonormality of the vectors implies $v_i^\top A(Av_j) = v_j^\top A(Av_i) = 0$ for all j such that $j+1 < i$. This implies that we need to A-orthogonalize Av_i only to vectors v_i and v_{i-1}. Hence, the time required for constructing this basis is dominated by $O(k)$ multiplications of the matrix A with a given vector, and $O(k)$ dot-product computations. Hence, we can find the best vector in the Krylov subspace efficiently enough.

Approximation guarantee. We now analyze the approximation guarantee achieved by this vector. Note that the Krylov subspace $\mathcal{K} = \mathsf{Span}\{v, Av, \ldots, A^k v\}$ consists of exactly those vectors which can be expressed as $\sum_{i=0}^{k} \beta_i A^i v = p(A)v$, where p is a degree-k polynomial defined by the coefficients β_i. Let Σ_k denote the set of all degree-k polynomials. Since the output vector x_k is the vector in the subspace that achieves the best possible error guarantee, we have

$$\|x_k - x^\star\|_A^2 = \inf_{x \in \mathcal{K}} \|x^\star - x\|_A^2 = \inf_{p \in \Sigma_k} \|x^\star - p(A)v\|_A^2 \leq \|x^\star\|_A^2 \cdot \inf_{p \in \Sigma_k} \|I - p(A)A\|^2.$$

Observe that the last expression can be written as

$$\|x^\star\|_A^2 \cdot \inf_{q \in \Sigma_{k+1}, q(0)=1} \|q(A)\|^2,$$

where the minimization is now over degree-$(k+1)$ polynomials q that evaluate to 1 at 0. Since A is symmetric and, hence, diagonalizable, we know that

$$\|q(A)\|^2 = \max_i |q(\lambda_i)|^2 \leq \sup_{\lambda \in [\lambda_n, \lambda_1]} |q(\lambda)|^2,$$

where $0 < \lambda_n \leq \cdots \leq \lambda_1$ denote the eigenvalues of the matrix A. Hence, in order to prove that an error guarantee of $\|x_k - x^\star\|_A \leq \delta \|x^\star\|_A$ is achieved after k rounds, it suffices to show that there exists a polynomial of degree $k+1$ that takes value 1 at 0, and whose magnitude is less than δ on the interval $[\lambda_n, \lambda_1]$.

As a first attempt, we consider the degree-s polynomial

$$q_0(x) \stackrel{\text{def}}{=} \left(1 - \frac{2x}{(\lambda_1 + \lambda_n)}\right)^s.$$

The maximum value attained by q_0 over the interval $[\lambda_n, \lambda_1]$ is $\left({}^{(\kappa-1)}/_{(\kappa+1)}\right)^s$. Hence, $d_0 \stackrel{\text{def}}{=} \lceil \kappa \log {}^1/_\delta \rceil$ suffices for this value to be less than δ. Or equivalently, approximately $\kappa \log {}^1/_\delta$ rounds suffice for error guarantee $\|x - x^\star\|_A \leq \delta \|x^\star\|_A$, recovering the guarantee provided by the gradient descent-based method.

However, for a better guarantee, we can apply the polynomial approximation to x^{d_0} developed in Section 3. Let $z \stackrel{\text{def}}{=} 1 - {}^{2x}/_{(\lambda_1+\lambda_n)}$. Hence, $q_0(x) = z^s$. As x ranges over $[0, \lambda_n + \lambda_1]$, the variable z varies over $[-1, 1]$. Theorem 3.3 implies that for $d \stackrel{\text{def}}{=} \left\lceil \sqrt{2 d_0 \log {}^2/_\delta} \right\rceil$, the polynomial $p_{d_0,d}(z)$ approximates the polynomial z^{d_0} up to an error of δ over $[-1, 1]$. Hence, the polynomial $q_1(x) \stackrel{\text{def}}{=} p_{d_0,d}(z)$ approximates $q_0(x)$ up to δ for all $x \in [0, \lambda_1 + \lambda_n]$. Combining this with the observations from the previous paragraph, $q_1(x)$ takes value at most 2δ on the interval $[\lambda_n, \lambda_1]$, and at least $1 - \delta$ at 0. Thus, the polynomial ${}^{q_1(x)}/_{q_1(0)}$ is a polynomial of degree $d = O(\sqrt{\kappa} \log {}^1/_\delta)$ that takes value 1 at 0, and at most ${}^{2\delta}/_{(1-\delta)} = O(\delta)$ on the interval $[\lambda_n, \lambda_1]$. Or equivalently, $O(\sqrt{\kappa} \log {}^1/_\delta)$ rounds suffice for an error guarantee $\|x - x^\star\|_A \leq O(\delta)\|x^\star\|_A$, which gives a quadratic improvement over the guarantee provided by the gradient descent-based method. This completes the proof of Theorem 9.1.

Notes

The above proof of the guarantee of the Conjugate Gradient method is different from the traditional proof, which directly shows that the polynomial $T_d(1 - {}^{2x}/_{(\lambda_1+\lambda_n)})$ for $d = O(\sqrt{\kappa} \log {}^1/_\delta)$ takes value 1 at 0 and is less than δ on the interval $[\lambda_n, \lambda_1]$ (see, *e.g.*, [66]). ($T_d(\cdot)$ is the Chebyshev polynomial of order d.)

Another intriguing question here concerns the Accelerated Gradient Descent method for optimizing convex functions, described by Nesterov [41, 42]. When applied to a quadratic function, Nesterov's method reduces to a Conjugate Gradient-type method. It would be interesting if the proof for Conjugate Gradient presented in this section can be extended to (the more general) Nesterov's method.

10

Computing Eigenvalues via the Lanczos Method

In this section we present the Lanczos method for approximating the largest eigenvalue of a matrix and prove that it achieves a quadratic improvement over the number of iterations required by the Power method for approximating the largest eigenvalue.

The Conjugate Gradient method is one of several methods that work with the Krylov subspace, collectively called *Krylov subspace methods*. Another Krylov subspace method of particular interest is the Lanczos method, which is typically employed for approximating the eigenvalues and eigenvectors of a symmetric matrix; see [53] for an extensive discussion. We present the Lanczos method and show how the existence of good polynomial approximations to x^s allow us to easily improve upon the Power method. Concretely, we prove the following theorem about the Lanczos method.

Theorem 10.1. Given a symmetric PSD matrix A, and a parameter $\delta > 0$, the Lanczos method after k iterations, for $k = O(1/\sqrt{\delta} \cdot \log n/\delta)$, outputs a value $\mu \in [(1-\delta)\lambda_1(A), \lambda_1(A)]$ with constant probability. The total number of operations required is $O((t_A + n)k + k^2)$, where t_A is the number of operations required to multiply A with a given vector.

The key thing is the dependence of $1/\sqrt{\delta}$ on δ as opposed to that $1/\delta$ as in the Power method; a quadratic improvement. Such a result was proven by Kuczyński and Woźniakowski [35]. Our proof is simpler and utilizes the $\sqrt{s}$ degree approximation for x^s presented in Section 3.

10.1 Proof of Theorem 10.1

We start by recalling the following variational characterization of the largest eigenvalue of a symmetric matrix.

Proposition 10.1. For a symmetric matrix A, its largest eigenvalue $\lambda_1(A)$ is characterized as follows

$$\lambda_1(A) = \max_{w \neq \mathbf{0}} \frac{w^\top A w}{w^\top w}.$$

For a vector w, the value $\frac{w^\top A w}{w^\top w}$ is called its Rayleigh quotient w.r.t. A. Thus, to compute $\lambda_1(A)$, it suffices to find a non-zero vector which maximizes the Rayleigh quotient. The Power method (see [66, Chapter 8]) tells us that for a unit vector v chosen uniformly at random, with constant probability, the vector $A^s v$ satisfies

$$\frac{(A^s v)^\top A (A^s v)}{(A^s v)^\top (A^s v)} \geq (1-\delta)\lambda_1(A)$$

for s roughly $1/\delta$. The Lanczos method *essentially* finds the vector in the Krylov subspace $\mathcal{K} \stackrel{\text{def}}{=} \mathsf{Span}\{v, Av, \ldots, A^k v\}$ that maximizes the Rayleigh quotient. It is sufficient to let $k = s \approx 1/\delta$ as dictated by the Power method. We prove below, again using the polynomial approximations to x^s from Section 3, that in order to find a vector with Rayleigh quotient at least $(1-\delta)\lambda_1$, it suffices to choose k to be approximately $1/\sqrt{\delta}$. Finding a vector in the Krylov subspace of order k that maximizes the Rayleigh quotient requires us to compute a basis for this subspace followed by computing the largest eigenvalue of a tridiagonal matrix of size $k+1$. The basis computation is identical to that in the Conjugate Gradient method and computing the largest eigenvalue of the tridiagonal matrix can be done in $O(k^2)$ time.

Formally, let $\lambda_1 \geq \cdots \geq \lambda_n$ be the eigenvalues of A, and let $u_1, \ldots, u_n$ be the corresponding eigenvectors. Let $\delta > 0$ be a specified error parameter. Let v be a unit vector and let $\{v_0, \ldots, v_k\}$ be any orthonormal basis

for the Krylov subspace constructed starting with the vector v, *i.e.*, $\mathcal{K} = \mathsf{Span}\{v, Av, \ldots, A^k v\}$.[1] Let V denote the $n \times (k+1)$ matrix whose i-th column is v_i. Thus, $V^\top V = I_{k+1}$ and $VV^\top$ is the orthogonal projection onto $\mathcal{K}$. Let $T \stackrel{\text{def}}{=} V^\top AV$. The $(k+1) \times (k+1)$ matrix T denotes the operator A restricted to $\mathcal{K}$, expressed in the basis $\{v_i\}_{i=0}^k$. Thus, for every $w \in \mathcal{K}$, we have $w = VV^\top w$, and hence,

$$w^\top Aw = (w^\top VV^\top)A(VV^\top w) = w^\top V(V^\top AV)V^\top w = (w^\top V)T(V^\top w). \tag{10.1}$$

The above equality says that for any vector $w \in \mathcal{K}$, the Rayleigh quotient of the vector w with respect to A is the same as the Rayleigh quotient of the vector $V^\top w$ with respect to T. Thus, combining (10.1) with Proposition 10.1, it follows that $\lambda_1(T) \leq \lambda_1$. In the other direction, we show that for a carefully chosen v and k, $\lambda_1(T)$ is nearly as large as λ_1. Thus, the Lanczos method computes the largest eigenvalue of T, $\lambda_1(T)$ for such a choice of v and k, and outputs it as an approximation to λ_1. We proceed to describe how to choose k and v.

Since $\mathcal{K}$ is a $k+1$ dimensional subspace with the columns of V as an orthonormal basis, as z ranges over $\mathcal{K}$, the vector $V^\top z$ ranges over $\mathbb{R}^{k+1}$. Also, for any vector $z \in \mathcal{K}$, we know that $z^\top V^\top TVz = z^\top Az$. Combining these two observations, we obtain the following:

$$\lambda_1(T) = \max_{w \in \mathbb{R}^{k+1}} \frac{w^\top Tw}{w^\top w} = \max_{z \in \mathcal{K}} \frac{z^\top VTV^\top z}{z^\top z} = \max_{z \in \mathcal{K}} \frac{z^\top Az}{z^\top z}. \tag{10.2}$$

Since every $z \in \mathcal{K}$ can be expressed as $p(A)v$ for some $p \in \Sigma_k$, we obtain, using (10.2), that,

$$\lambda_1(T) = \max_{z \in \mathcal{K}} \frac{z^\top Az}{z^\top z} = \max_{p \in \Sigma_k} \frac{v^\top p(A)Ap(A)v}{v^\top p(A)^2 v}.$$

Writing $v = \sum_{i=1}^n \alpha_i u_i$, where u_i are the eigenvectors of A, and hence also the eigenvectors of $p(A)^2$ and $Ap(A)^2$, we get,

$$\lambda_1(T) = \max_{p \in \Sigma_k} \frac{v^\top p(A)Ap(A)v}{v^\top p(A)^2 v} = \max_{p \in \Sigma_k} \frac{\sum_i \lambda_i p(\lambda_i)^2 \alpha_i^2}{\sum_i p(\lambda_i)^2 \alpha_i^2}.$$

[1] Later on we show how to construct such a basis quickly, similar to the case of the Conjugate Gradient method.

Here λ_i is the eigenvalue of A corresponding to u_i. Thus, for any $p \in \Sigma_k$, we can bound how well $\lambda_1(T)$ relatively approximates λ_1 as follows:

$$\frac{\lambda_1 - \lambda_1(T)}{\lambda_1} \leq \frac{\sum_{i=1}^n (1 - \lambda_i/\lambda_1) p(\lambda_i)^2 \alpha_i^2}{\sum_{i=1}^n p(\lambda_i)^2 \alpha_i^2}.$$

Here we have used $\lambda_1 > 0$. In the remainder of this section we assume that $\lambda_i > 0$ for all i which follows from the assumption that A is PSD.

We choose v to be a uniformly random unit vector. A standard concentration bound then implies that with probability at least $1/2$, we have $\alpha_1^2 \geq 1/9n$. Thus, assuming that $\alpha_1^2 \geq 1/9n$, we split the sum in the numerator depending on whether $\lambda \geq (1-\delta)\lambda_1$, or otherwise.

$$\begin{aligned}\frac{\sum_{i=1}^n (1 - \lambda_i/\lambda_1) p(\lambda_i)^2 \alpha_i^2}{\sum_{i=1}^n p(\lambda_i)^2 \alpha_i^2} &\leq \delta + \frac{\sum_{\lambda_i < (1-\delta)\lambda_1} p(\lambda_i)^2 \alpha_i^2}{p(\lambda_1)^2 \alpha_1^2} \\ &\leq \delta + 9n \sup_{\lambda \in [0,(1-\delta)\lambda_1]} \frac{p(\lambda)^2}{p(\lambda_1)^2}.\end{aligned}$$

Observe that if we choose the polynomial $p(\lambda) = (\lambda/\lambda_1)^s$ for $s \stackrel{\text{def}}{=} \lceil 1/2\delta \cdot \log 9n/\delta \rceil$ in the above bounds, we can bound the relative error to be

$$\frac{\lambda_1 - \lambda_1(T)}{\lambda_1} \leq \delta + 9n \cdot \sup_{\lambda \in [0,(1-\delta)\lambda_1]} \left(\frac{\lambda}{\lambda_1}\right)^{2s} = \delta + 9n \cdot (1-\delta)^{2s} \leq 2\delta.$$

Hence, the Lanczos method after $k = O(1/\delta \cdot \log n/\delta)$ iterations finds a vector with Rayleigh quotient at least $(1 - O(\delta))\lambda_1$ with constant probability, essentially matching the guarantee of the Power method.

However, we use the polynomial approximations $p_{s,d}$ to x^s from Section 3 to show that the Lanczos method does better. We use the polynomial $p(\lambda) = p_{s,d}(\lambda/\lambda_1)$ for $s = \lceil 1/2\delta \cdot \log 9n/\delta \rceil$ as above, and $d = \left\lceil \sqrt{2s \cdot \log 2n/\delta} \right\rceil$. In this case, we know that for all λ such that $|\lambda/\lambda_1| \leq 1$, we have $|p(\lambda) - (\lambda/\lambda_1)^s| \leq \delta/n$. Hence, $p(\lambda_1) \geq 1 - \delta/n$, and

$$\sup_{\lambda \in [0,(1-\delta)\lambda_1]} p(\lambda)^2 \leq \sup_{\lambda \in [0,(1-\delta)\lambda_1]} (\lambda/\lambda_1)^{2s} + \delta/n = O(\delta/n),$$

giving the following bound on the relative error

$$\frac{\lambda_1 - \lambda_1(T)}{\lambda_1} \leq \delta + 9n \cdot \frac{O(\delta/n)}{1 - \delta/n} \leq O(\delta).$$

Since the degree of this polynomial is d, we obtain that $k = d = O(1/\sqrt{\delta} \cdot \log n/\delta)$ iterations of Lanczos method suffice to find a vector with Rayleigh quotient at least $(1 - O(\delta))\lambda_1$.

Running time. It remains to analyze the time taken by this algorithm to estimate λ_1. Let t_A denote the number of operations required to compute Au, given a vector u. We first describe how to quickly compute an orthonormal basis for $\mathcal{K}$. The procedure is essentially the same as the one used in the Conjugate Gradient method. We iteratively compute Av_i, orthogonalize it with respect to $v_i, \ldots, v_0$, and scale it to norm 1 in order to obtain v_{i+1}. As in the Conjugate Gradient method, we have $Av_j \in \mathsf{Span}\{v_0, \ldots, v_{j+1}\}$ for all $j < k$ and, hence, using the symmetry of A, we obtain $v_j^\top (Av_i) = v_i^\top (Av_j) = 0$ for $j+1 < i$. Thus, we need to orthogonalize Av_i only with respect to v_i and v_{i-1}. This also implies that T is tridiagonal. Hence, we can construct V and T using $O((t_A + n)k)$ operations. (Note the subtle difference with respect to the Conjugate Gradient method – here, by using usual inner products instead of A-inner products, we ensure that the basis vectors are orthonormal instead of A-orthogonal.) The only remaining step is to compute the largest eigenvalue of T, which can be found via an eigendecomposition of T. Since T is tridiagonal, this step can be upper bounded by $O(k^2)$ (see [52]). The eigenvector w of T which achieves $\lambda_1(T)$ can be used to give a candidate for the largest eigenvector of A, *i.e.*, the vector Vw. This completes the proof of Theorem 10.1.

Notes

Theorem 10.1 can be easily generalized to the case when A is symmetric and not necessarily PSD. Further, the Lanczos method can also be used to approximate several large eigenvalues of A. The algorithm is essentially the same, except that we choose a Krylov subspace of higher order k, and output the top eigenvalues of the matrix T. Using techniques similar to the ones presented above, we can achieve a similar speed-up when the largest eigenvalues of A are well-separated. Such results were obtained by Kaniel [31] and Saad [56] (see [57, Chapter 6] and the notes therein).

Finally, the Lanczos method can in fact be used more generally to obtain a fast approximation to $f(A)v$ for any function f, and any vector v. If

we work with the Krylov subspace $\{v, Av, \ldots, A^k v\}$, it can be shown that for any polynomial $p \in \Sigma_k$, we have $V p(T) V^\top v = p(A)v$. Hence, a natural approximation for $f(A)v$ is $V f(T) V^\top v$. Moreover, using the method above, the number of operations required is $O((t_A + n)k)$ plus those required to compute $f(\cdot)$ on the $(k+1) \times (k+1)$ tridiagonal matrix T, which can usually be upper bounded by $O(k^2)$ via diagonalization (see [52]). Letting $\mathcal{I} \stackrel{\text{def}}{=} [\lambda_n(A), \lambda_1(A)]$, the error in the approximation can be upper bounded by the uniform approximation error achieved by the best degree k polynomial approximating f on $\mathcal{I}$; see [66, Chapter 19]. This method derives its power from the fact that in order to compute a good approximation, just the *existence* of a good polynomial that approximates f on $\mathcal{I}$ is sufficient, and we do not need to *know* the coefficients of the polynomial.

11

Computing the Matrix Exponential

In this section we use the rational approximation for e^{-x} from Section 7 to give a fast algorithm to compute an approximation to $\exp(-A)v$, given a symmetric PSD matrix A and a vector v. When A is a symmetric and diagonally dominant (SDD) matrix, the running time is near-linear in the number of non-zero entries in A, and depends only logarithmically on the spectral norm of A.

We consider the problem of computing $\exp(-A)v$ for an $n \times n$ PSD matrix A and a vector v. Recall that

$$\exp(-A) = \sum_{k=0}^{\infty} \frac{(-1)^k A^k}{k!}.$$

Of particular interest is the special case

$$\exp(-s(I - \widetilde{W})) = e^{-s} \sum_{k \geq 0} \frac{s^k}{k!} \widetilde{W}^k,$$

where $\widetilde{W}$ is the random walk matrix associated with a graph $G = (V, E)$ defined in Section 8, since this matrix corresponds to the transition matrix of a *continuous-time* random walk of length s on G, also called the *heat-kernel*

walk on G; see [17, 37]. In terms of the normalized Laplacian $\mathcal{L} = I - \widetilde{W}$, the above matrix can be expressed as $\exp(-s\mathcal{L})$. Note that this walk can be interpreted as the distribution of a discrete-time random walk after a Poisson-distributed number of steps with mean s since $\exp(-s\mathcal{L}) = e^{-s}\sum_{k\geq 0}\frac{s^k\widetilde{W}^k}{k!}$. These random walks are of importance in probability and algorithms, and the ability to simulate them in time near-linear in the number of edges in the graph results in near-linear time algorithms for problems including a version of the Sparsest Cut problem (described in the notes at the end of Section 8) with an additional *balance* constraint which requires that the two sides of the cut be linear in size; see [49]. More generally, fast computation of $\exp(-A)v$ plays a crucial role, via the *matrix multiplicative weights update* method, in obtaining fast algorithms to solve semi-definite programs; see [6, 47, 5].

In this section we prove the following theorem which gives a near-linear time algorithm that allows us to approximate the matrix exponential $\exp(-s\mathcal{L})v$ for a given s, a vector v, and the normalized Laplacian $\mathcal{L}$ for a regular graph. More generally, the algorithm allows us to approximate $\exp(-A)v$ where A is any *symmetric and diagonally dominant* (SDD) matrix. Recall that a matrix A is said to be SDD if it is symmetric and for all i we have $A_{ii} \geq \sum_{j\neq i}|A_{ij}|$. The normalized Laplacian for a regular graph is SDD. As in Section 8, the result can be generalized to irregular graphs in a straightforward manner; see the notes at the end of this section for details.

Theorem 11.1. There is an algorithm that, given an $n \times n$ SDD matrix A with m non-zero entries, a vector v, and $\delta \in (0,1]$, computes a vector u such that $\left\|\exp(-A)v - u\right\| \leq \delta\|v\|$ in time $\widetilde{O}((m+n)\log(2+\|A\|)\text{polylog } 1/\delta)$.

The most natural way to compute $\exp(-A)v$ is to approximate the matrix exponential using the Taylor series approximation for the exponential, or to use the improved polynomial approximations constructed in Section 4. Indeed, Theorem 4.1 can be used to compute a δ approximation to $\exp(-A)v$ in time $O\left((t_A+n)\sqrt{\|A\|\log 1/\delta}\right)$; similar to Theorem 8.1. However, Theorem 5.3 implies that no polynomial approximation can get rid of the dependence on $\sqrt{\|A\|}$ in the running time above.

What about rational approximations to e^{-x} proved in Section 6? Indeed, we can use the rational approximation from Theorem 6.1 to obtain

$\| \exp(-A) - \left(S_d(A)\right)^{-1} \| \leq 2^{-\Omega(d)}$, where $\left(S_d(A)\right)^{-1}$ is the approximation to $\exp(-A)$ defined by $S_d(x) = \sum_{k=0}^{d} \frac{x^k}{k!}$. For most applications, an error of $\delta = 1/\mathrm{poly}(n)$ suffices, so we can choose $d = O(\log n)$. How do we compute $\left(S_d(A)\right)^{-1} v$? Clearly, inverting $S_d(A)$ is not a good idea since that would be at least as inefficient as matrix inversion. The next natural idea is to factor $S_d(x) = \alpha_0 \prod_{i=1}^{d} (x - \beta_i)$ and then calculate $(S_d(A))^{-1} v = \alpha_0 \prod_{i=1}^{d} (A - \beta_i I)^{-1} v$. Since d is small, namely $O(\log n)$, the cost of computing $(S_d(A))^{-1} v$ reduces to the cost of computing $(A - \beta_i I)^{-1} u$. Thus, it suffices to do a computation of this form. The first problem is that β_is could be complex, as is indeed the case for the polynomial S_d as discussed in Section 5. However, since S_d has real coefficients, its complex roots appear as conjugates. Hence, we can combine the factors corresponding to the conjugate pairs and reduce the task to computing $(A^2 - (\beta_i + \bar{\beta}_i)A + |\beta_i|^2 I)^{-1} u$. The matrix $(A^2 - (\beta_i + \bar{\beta}_i)A + |\beta_i|^2 I)$ is easily seen to be PSD and we can try to apply the Conjugate Gradient method to compute $(A^2 - (\beta_i + \bar{\beta}_i)A + |\beta_i|^2 I)u$. However, the condition number of this matrix can be comparable to that of A, which gives no significant advantage over $\sqrt{\|A\|}$.[1]

Similarly, the rational approximations to e^{-x} constructed in Section 7 suggest the vector $p_d((I + A/d)^{-1})v$ as an approximation to $\exp(-A)v$, where p_d is the polynomial given by Theorem 7.1. Once again, for any PSD matrix A, though the matrix $(I + A/d)$ is PSD, the condition number of $(I + A/d)$ could be comparable to that of A. Hence for arbitrary PSD matrices, the rational approximations to e^{-x} seem insufficient for obtaining improved algorithms for approximating the matrix exponential. Indeed, $O\left((t_A + n)\sqrt{\|A\| \log 1/\delta}\right)$ is the best result known for computing the matrix-exponential-vector product for a general PSD matrix A; see [49].

The above approach of using rational approximations shows how to reduce the approximation of $\exp(-A)v$ to solving a small number of linear systems involving the matrix $I + A/d$, or equivalently, approximating $(I + A/d)^{-1} u$ for a given vector u. For an important special class of matrices, we can exploit the fact that there exist algorithms that are much faster than Conjugate Gradi-

[1] To see this, consider a matrix A with $\lambda_1(A) \gg d$, and $\lambda_n(A) = 1$. It is known that $|\beta_i| \leq d$; see [68]. Hence, for such a matrix A, the condition number of $(A^2 - (\beta_i + \bar{\beta}_i)A + |\beta_i|^2 I)$ is $\Omega(\lambda_1^2(A)/d^2)$, which is approximately the square of the condition number of A for small d.

ent and allow us to approximate $(I + A/d)^{-1}u$, for a given u. In particular, for an SDD matrix A, there are powerful near-linear-time SDD system solvers whose guarantees are given by the following theorem (see [62, 34, 32]).

Theorem 11.2. Given an $n \times n$ SDD matrix B with m non-zero entries, a vector v, and $\delta_1 > 0$, there is an algorithm that, in $\widetilde{O}(m \log 1/\delta_1)$ time, computes a vector u such that $\|u - B^{-1}v\|_B \leq \delta_1 \|B^{-1}v\|_B$. Moreover, $u = Zv$ where Z depends on B and δ_1, and is such that $(1-\delta_1)B^{-1} \preceq Z \preceq (1+\delta_1)B^{-1}$.

At the end of Section 7, we proved that the coefficients of p_d from Theorem 7.1 and are bounded by $d^{O(d)}$, and can be computed up to an accuracy of $2^{-\text{poly}(d)}$ in $\text{poly}(d)$ time. We now show that, assuming that we know the coefficients of p_d exactly, we can compute $p_d((I + A/d)^{-1})v$ as an approximation to $\exp(-A)v$ in near-linear time using Theorem 11.2. The error in the approximation due of the error in the coefficients of p_d can be handled in a term-by-term manner, and is ignored in this monograph.

Note that if A is SDD, so is $(I + A/d)$. We repeatedly use the SDD solver of Theorem 11.2 to approximate $(I + A/d)^{-i}v$, for all $i = 1, \ldots, d$, and let Z denote the linear operator such that the SDD solver returns the vector $Z^i u$ as the approximation. Let $B \stackrel{\text{def}}{=} (I + A/d)^{-1}$. From the guarantee on the SDD solver from the theorem above, we know that $-\delta_1 B \preceq Z - B \preceq \delta_1 B$. Applying the triangle inequality to the identity

$$Z^i - B^i = \sum_{j=0}^{i-1} Z^{i-1-j}(Z - B)B^j,$$

and using $\|B\| \leq 1$, we obtain, $\left\|Z^i - B^i\right\| \leq \delta_1 \cdot i(1+\delta_1)^i$. Thus,

$$\left\|p_d(Z) - p_d(B)\right\| \leq d^{O(d)} \cdot \delta_1 (1+\delta_1)^d.$$

Hence, we can choose $\delta_1 = \delta \cdot d^{-\Theta(d)}$ for the SDD solver in order for the final approximation to have error at most δ. Since $d = \Theta(\log 1/\delta)$ suffices, this results in an overall running time of $\widetilde{O}(m \, \text{polylog} 1/\delta)$, completing the proof of Theorem 11.1.

Coming back to graphs, an important corollary of this theorem is that $\exp(-s\mathcal{L})v$, the distribution after an s-length continuous time random walk on a regular graph with normalized Laplacian $\mathcal{L}$ starting with distribution v, can be approximately computed in $\widetilde{O}(m \log s)$ time.

Corollary 11.3. Let $\widetilde{W}$ be the random walk matrix for a graph G with n vertices and m edges . Then, for any positive integer s, and a unit vector v, and $\delta \in (0, 1/2]$, there is an algorithm that computes a vector w such that $\left\| \exp(-s(I-\widetilde{W}))v - w \right\| \leq \delta$ in $O\left((m+n)\log(2+s) \cdot \text{polylog} 1/\delta\right)$ arithmetic operations.

One should contrast this corollary with an analogous result for simple random walks from Section 8.1. In the latter case, we do not know how to improve upon the $\sqrt{s}$ term in the running time.

Notes

For irregular graphs, the walk matrix $\widetilde{W}$ is not symmetric and, hence, the normalized Laplacian $\mathcal{L}$ is not SDD. We can extend the results of this section by working with the symmetric matrix $I - D^{-1/2}\widetilde{W}D^{1/2}$ and observing that $\exp(-s(I-\widetilde{W})) = D^{1/2}\exp(-s(I-D^{-1/2}\widetilde{W}D^{1/2}))D^{-1/2}$. Thus, to compute $\exp(-s(I-\widetilde{W}))v$, it suffices to be able to compute $\exp(-s(I-D^{-1/2}\widetilde{W}D^{1/2}))u$, which can be approximated using the results of this section.

Theorem 11.1 was first proved by Orecchia, Sachdeva, and Vishnoi in [49]. However, instead of computing the coefficients of the polynomial p_d explicitly, the authors in [49] appealed to the Lanczos method which allows one to achieve the error guarantee of the approximating polynomial *without* explicit knowledge of the polynomial (briefly described in the notes at the end of Section 10).

We can now also reconsider the problem of approximating $\widetilde{W}^s v$ considered in Section 8. One approach to an algorithm for this problem, of improving the dependence of the running time on s beyond $\sqrt{s}$, would be to use the rational approximation for x^s given by Newman [45] (see notes at the end of Section 6), implying that we can approximate $\widetilde{W}^s v$ as $\left(\sum_{i=0}^{d} \binom{s+i-1}{i}(I-\widetilde{W})^i\right)^{-1} v$, for any random walk matrix $\widetilde{W}$, and vector v, where $d = O(\log 1/\delta)$ suffices for δ-approximation. However, it is not clear if the required matrix-inverse-vector-product can be computed or approximated quickly.

12

Matrix Inversion via Exponentiation

In this section we illustrate the power of approximation by *transcendental* functions. We show that $1/x$ can be well approximated by a *sparse* sum of the form $\sum_i w_i e^{-t_i x}$. As an immediate application we obtain that the problems of approximating the matrix exponential and the matrix inverse are essentially equivalent.

We present a rather short and surprising result which reduces a computation of the form $A^{-1}v$ for a PSD A, to the computation of a small number of terms of the form $\exp(-sA)v$. One way to interpret this result is that the linear system solvers deployed in the previous section are *necessary*. The other way is to see this as a new approach to speed up computations beyond the Conjugate Gradient method to compute $A^{-1}v$ for PSD matrices, a major open problem in numerical linear algebra with implications far beyond. The following theorem summarizes the main result of this section.

Theorem 12.1. Given $\varepsilon, \delta \in (0,1]$, there exist $\mathrm{poly}(\log 1/\varepsilon\delta)$ numbers $0 < w_j, t_j = O(\mathrm{poly}(1/\varepsilon\delta))$, such that for all symmetric matrices A satisfying

$\varepsilon I \preceq A \preceq I$, we have

$$(1-\delta)A^{-1} \preceq \sum_j w_j e^{-t_j A} \preceq (1+\delta)A^{-1}.$$

Moreover, the coefficients w_j, t_j in the above theorem can be computed efficiently. Thus, the above theorem implies a reduction from approximate matrix inversion (equivalent to approximately solving a linear system) to approximating the matrix exponential. Combined with the reduction from the last section, this theorem proves that the two problems are essentially equivalent.

Since the above reduction only requires that the matrix A be positive-definite, it immediately suggests an approach to approximating $A^{-1}v$: Approximate $e^{-t_j A}v$ for each j and return the vector $\sum_j w_j e^{-t_j A}v$ as an approximation for $A^{-1}v$. Since the weights w_j are $O(\mathrm{poly}(1/\delta\varepsilon))$, we lose only a polynomial factor in the approximation error.

Finally, we emphasize that the approximation for A^{-1} by exponentials given in the above theorem has $\mathrm{poly}(\log(1/\varepsilon\delta))$ terms as opposed to $\mathrm{poly}(1/\varepsilon\delta)$ terms. This distinction is crucial: for applications it is important to be able to compute a δ-approximation to $A^{-1}v$ with polylogarithmic time dependence on both $1/\delta$ and the condition number of A ($1/\varepsilon$ in this case); see Theorem 11.2.

12.1 Approximating x^{-1} Using Exponentials

Theorem 12.1 is an immediate corollary of the following theorem which shows than we can approximate x^{-1} with a sum of a small number of exponentials, where the approximation is valid for all $x \in [\varepsilon, 1]$.

Theorem 12.2. Given $\varepsilon, \delta \in (0,1]$, there exist $\mathrm{poly}(\log 1/\varepsilon\delta)$ numbers $0 < w_j, t_j = O(\mathrm{poly}(1/\varepsilon\delta))$, such that for all $x \in [\varepsilon, 1]$, we have

$$(1-\delta)x^{-1} \leq \sum_j w_j e^{-t_j x} \leq (1+\delta)x^{-1}.$$

Similar results have appeared in the works of Beylkin and Monzón [9, 10]. We present a proof that is shorter and simpler. The starting point of the proof of Theorem 12.2 is the identity $x^{-1} = \int_0^\infty e^{-xt}\,\mathrm{d}t$. The crux of the proof is to discretize this integral to a *sparse* sum of exponentials. One approach is to

discretize an integral to a sum is via the *trapezoidal rule*; *i.e.*, by approximating the area under the integral using *trapezoids* of small width, say h:

$$\int_a^b g(t)\,\mathrm{d}t \;\approx\; T_g^{[a,b],h} \stackrel{\text{def}}{=} \frac{h}{2}\cdot\sum_{j=0}^{K-1}\big(g(a+jh)+g(a+(j+1)h)\big)\,,$$

where $K \stackrel{\text{def}}{=} {}^{(b-a)}\!/_{h}$ is an integer. Applying this rule to the above integral after truncating it to a large enough interval $[0,b]$, we obtain the approximation

$$x^{-1} \approx \frac{h}{2}\sum_{j=0}^{{}^{b}\!/_{h}-1}\left(e^{-xjh}+e^{-x(j+1)h}\right).$$

The error in truncating the integral is $\int_b^\infty e^{-xt}\,\mathrm{d}t = x^{-1}e^{-bx}$. For the truncation error to be at most ${}^{\delta}\!/_{x}$ for all $x\in[\varepsilon,1]$, we need b to be at least ${}^{1}\!/_{\varepsilon}\cdot\log{}^{1}\!/_{\delta}$. The discretization of the interval and, hence, the sparsity of the approximating sum K, are determined by the choice of h. The error due to discretization is given by

$$\left|x^{-1}-\frac{h}{2}\sum_j\left(e^{-xjh}+e^{-x(j+1)h}\right)\right|.^{1}$$

We would like this discretization error to be at most ${}^{\delta}\!/_{x}$ for all $x\in[\varepsilon,1]$. In particular, for $x=1$, this gives us the following constraint, which only involves h and δ:

$$\left|1-\frac{h}{2}\cdot\frac{1+e^{-h}}{1-e^{-h}}\right|\le\delta.$$

This implies an upper bound of $O(\sqrt{\delta})$ on h, for small enough δ. Together, these constraints imply that the sparsity $K={}^{b}\!/_{h}$ must be $\Omega({}^{1}\!/_{(\varepsilon\sqrt{\delta})})$. Thus, controlling the truncation error and the discretization error under uniform discretization can only lead to a sum which uses $\mathrm{poly}({}^{1}\!/_{(\varepsilon\delta)})$ exponential terms which, as mentioned earlier, is insufficient for applications. We now show that under a clever (non-uniform) discretization of the above integral, we are able to achieve the desired error using only $\mathrm{poly}(\log({}^{1}\!/_{\varepsilon\delta}))$ exponentials.

The above argument suggests that we should select a discretization where t increases much more rapidly with h, *e.g.*, exponentially instead of linearly.

[1] To be precise, the error is given by $\left|\int_0^b e^{-xt}\,\mathrm{d}t-\frac{h}{2}\sum_{j=0}^{{}^{b}\!/_{h}-1}\left(e^{-xjh}+e^{-x(j+1)h}\right)\right|$ and, with a little more care, the argument above carries through.

This can be achieved by substituting $t = e^y$ in the above integral to obtain the identity

$$x^{-1} = \int_{-\infty}^{\infty} e^{-xe^y+y}\,\mathrm{d}y.$$

Let $f_x(y) \stackrel{\text{def}}{=} e^{-xe^y+y}$. Observe that $f_x(y) = x^{-1} \cdot f_1(y+\ln x)$. Since we allow the error to scale with x^{-1} as x varies over $[\varepsilon, 1]$, y needs to change only by an additive $\log 1/\varepsilon$ to compensate for x. This suggests that only roughly $1/h \cdot \log 1/\varepsilon$ additional terms are needed above those required for $x = 1$ in order for the approximation to hold for all $x \in [\varepsilon, 1]$, giving a logarithmic dependence on $1/\varepsilon$. We show that discretizing this integral using the trapezoidal rule, and bounding the error using the Euler-Maclaurin formula, does indeed give us the desired result.

12.2 Bernoulli Numbers and the Euler-Maclaurin Formula

The Bernoulli numbers, denoted by b_i for any integer $i \geq 0$, are a sequence of rational numbers which, while discovered in an attempt to compute sums of the form $\sum_{i\geq 0}^{k} i^j$, have deep connections to several areas of mathematics.[2] They can be defined recursively: $b_0 = 1$, and for all $k \geq 1$,

$$\sum_{j=0}^{k-1} \binom{k}{j} b_j = 0.$$

Given the Bernoulli numbers, the Bernoulli polynomials are defined to be

$$B_k(y) \stackrel{\text{def}}{=} \sum_{j=0}^{k} \binom{k}{j} b_j y^{k-j}.$$

Using properties of the Bernoulli polynomials, and a well-known connection to the Riemann zeta function, we obtain the following bounds (see [21]).

Lemma 12.3. For any non-negative integer k, and for all $y \in [0,1]$,

$$\frac{1}{(2k)!}|B_{2k}(y)| \leq \frac{1}{(2k)!}\,|b_{2k}| \leq \frac{4}{(2\pi)^{2k}}.$$

[2]The story goes that when Charles Babbage designed the Analytical Engine in the 19th century, one of the most important tasks he hoped the Engine would perform was the calculation of Bernoulli numbers.

One of the most significant connections in analysis involving the Bernoulli numbers is the *Euler-Maclaurin formula* which *exactly* describes the error in approximating an integral by the trapezoidal rule. For a function $g(y)$, let its k^{th} derivative be denoted by $g^{(k)}(y) \stackrel{\text{def}}{=} \frac{\mathrm{d}^k}{\mathrm{d}y^k} g(y)$.

Lemma 12.4. Given a function $g : \mathbb{R} \to \mathbb{R}$, for any $a < b$, any $h > 0$, and $N \in \mathbb{N}$, we have,

$$\int_a^b g(y)\,\mathrm{d}y - T_g^{[a,b],h} = h^{2N+1} \int_0^K \frac{1}{(2N)!} B_{2N}(y - [y]) g^{(2N)}(a + yh)\,\mathrm{d}y - \sum_{j=1}^{N} \frac{1}{(2j)!} b_{2j} h^{2j} \left(g^{(2j-1)}(b) - g^{(2j-1)}(a) \right), \tag{12.1}$$

where $K \stackrel{\text{def}}{=} \frac{b-a}{h}$ is an integer, and $[\cdot]$ denotes the integral part.

Note that it is really a family of formulae, one for each choice of N, called the *order* of the formula. The choice of N is influenced by how well behaved the higher order derivatives of the function are. For example, if $g(y)$ is a polynomial, when $2N > \text{degree}(g)$, we obtain an exact expression for $\int_a^b g(y)\,\mathrm{d}y$ in terms of the values of the derivatives of g at a and b. Since the sparsity of the approximation is $\Omega(1/h)$, for the sparsity to depend logarithmically on the error parameter δ, we need to choose N to be roughly $\Omega(\log 1/\delta)$ so that the first error term in (12.1) is comparable to δ.

The Proof

Proof. (of Theorem 12.2) We fix the step size h, approximate the integral $\int_{-bh}^{bh} f_x(y)\,\mathrm{d}y$ using the trapezoidal rule (b is a positive integer), and bound the approximation error using the Euler-Maclaurin formula. We let b go to ∞, which allows us to approximate the integral over $[-\infty, \infty]$ by an infinite sum of exponentials. Finally, we truncate this sum to obtain our approximation. Applying the order N Euler-Maclaurin formula to the integral $\int_{-bh}^{bh} f_x(y)\,\mathrm{d}y$,

and using Lemma 12.3, we obtain,

$$\left|\int_{-bh}^{bh} f_x(y)\,\mathrm{d}y - T_{f_x}^{[-bh,bh],h}\right| \le 4\left(\frac{h}{2\pi}\right)^{2N}\int_{-bh}^{bh}\left|f_x^{(2N)}(y)\right|\mathrm{d}y + \sum_{j=1}^{N} 4\left(\frac{h}{2\pi}\right)^{2j}\left(\left|f_x^{(2j-1)}(-bh)\right| + \left|f_x^{(2j-1)}(bh)\right|\right). \quad (12.2)$$

Now, the derivatives of the function $f_x(y)$ are well-behaved and easy to compute. By direct computation, for any k, its k^{th} derivative $f_x^{(k)}(y)$ is of the form $f_x(y)p_k(-xe^y)$, where p_k is a degree-k polynomial. Since the exponential function grows faster than any polynomial, this implies that for any fixed k and x, $f_x^{(k)}(y)$ vanishes as y goes to $\pm\infty$. We let b go to ∞ and observe that the discretized sum converges to $h\sum_{j\in\mathbb{Z}} f_x(jh)$, hence, (12.2) implies that

$$\left|\int_{-\infty}^{\infty} f_x(y)\,\mathrm{d}y - h\sum_{j\in\mathbb{Z}} f_x(jh)\right| \le 4\left(\frac{h}{2\pi}\right)^{2N}\int_{-\infty}^{\infty}\left|f_x^{(2N)}(y)\right|\mathrm{d}y. \quad (12.3)$$

Thus, all we need to show is that the function f_x is *smooth enough*. There is an easy recurrence between the coefficients of p_k for various k, and it allows us to crudely bound the sum of their absolute values by $(k+1)^{k+1}$. This, in turn, implies the bound $\int_{-\infty}^{\infty}\left|f_x^{(2N)}(y)\right|\mathrm{d}y \le x^{-1}\cdot\Theta(N)^{4N}$ (Both these bounds have been proved in the lemmas at the end of this section). Thus, we can choose $h=\Theta(1/N^2)$ and $N=\Theta(\log 1/\delta)$ to obtain the following approximation for all $x>0$:

$$\left|x^{-1} - h\sum_{j\in\mathbb{Z}} e^{jh}\cdot e^{-xe^{jh}}\right| = \left|\int_{-\infty}^{\infty} f_x(y)\,\mathrm{d}y - h\sum_{j\in\mathbb{Z}} f_x(jh)\right| = O\left(\delta\cdot x^{-1}\right). \quad (12.4)$$

The final step is to truncate the above discretization. Since the function $f_x(y)$ is non-decreasing for $y<\log 1/x$, we can majorize the lower tail with an integral to obtain

$$h\sum_{j<A} f_x(jh) \le \int_{-\infty}^{Ah} f_x(t)\,\mathrm{d}t = x^{-1}\left(1-e^{-xe^{Ah}}\right).$$

Thus, for $A=\left\lfloor -1/h\cdot\log 1/\delta\right\rfloor$, we obtain that the lower tail is $O(\delta\cdot x^{-1})$. Similarly, for the upper tail, using the fact that $f_x(y)$ is non-increasing

for $y \geq \log \frac{1}{x}$, for $B = \left\lceil {}^{1}/_{h} \cdot \log\left({}^{1}/_{\varepsilon} \log {}^{1}/_{\delta}\right) \right\rceil$, we obtain that the upper tail $h\sum_{j>B} f_x(jh)$ is $O(\delta \cdot x^{-1})$. Combining these tail bounds with (12.4), we obtain

$$\left| x^{-1} - h \sum_{j \geq A}^{B} e^{jh} \cdot e^{-xe^{jh}} \right| = O\left(\delta \cdot x^{-1}\right),$$

which completes the proof. Moreover, the coefficients are easily seen to be efficiently computable. □

For completeness we include the proof of the following two lemmas about the derivates of f_x used in the proof above.

Lemma 12.5. For any non-negative integer k, $f_x^{(k)}(s) = f_x(s)\sum_{j=0}^{k} c_{k,j}(-xe^s)^j$, where $c_{k,j}$ are some non-negative integers satisfying $\sum_{j=0}^{k} c_{k,j} \leq (k+1)^{k+1}$.

Proof. We prove this lemma by induction on k. For $k = 0$, we have $f_x^{(0)}(s) = f_x(s)$. Hence, $f_x^{(0)}$ is of the required form, with $c_{0,0} = 1$, and $\sum_{j=0}^{0} c_{0,j} = 1$. Hence, the claim holds for $k = 0$. Suppose the claim holds for k. Hence, $f_x^{(k)}(s) = f_x(s)\sum_{j=0}^{k} c_{k,j}(-xe^s)^j$, where $c_{k,j}$ are non-negative integers satisfying $\sum_{j=0}^{k} c_{k,j} \leq (k+1)^{k+1}$. We can compute $f_x^{(k+1)}(s)$ as follows,

$$\begin{aligned} f_x^{(k+1)}(s) &= \frac{\mathrm{d}}{\mathrm{d}s}\left(\sum_{j=0}^{k} c_{k,j}(-xe^s)^j f_x(s)\right) \\ &= \sum_{j=0}^{k} c_{k,j}(j - xe^s + 1)(-xe^s)^j f_x(s) \\ &= f_x(s)\sum_{j=0}^{k+1}((j+1)c_{k,j} + c_{k,j-1})(-xe^s)^j, \end{aligned}$$

where we define $c_{k,k+1} \stackrel{\text{def}}{=} 0$, and $c_{k,-1} \stackrel{\text{def}}{=} 0$. Thus, if we define

$$c_{k+1,j} \stackrel{\text{def}}{=} (j+1)c_{k,j} + c_{k,j-1},$$

we get that $c_{k+1,j} \geq 0$, and that $f_x^{(k+1)}$ is of the required form. Moreover, we

get,

$$\sum_{j=0}^{k+1} c_{k+1,j} \leq (k+2)(k+1)^{k+1} + (k+1)^{k+1} = (k+3)(k+1)(k+1)^k$$
$$\leq (k+2)^2(k+1)^k \leq (k+2)^{k+2}.$$

This proves the claim for $k+1$ and, hence, the fact follows by induction. □

The next lemma uses the fact above to bound the L_1 norm of $f_x^{(k)}$.

Lemma 12.6. For every non-negative integer k,

$$\int_{-\infty}^{\infty} \left| f_x^{(k)}(s) \right| \mathrm{d}s \leq 2x^{-1} \cdot e^k (k+1)^{2k}.$$

Proof. By Fact 12.5,

$$\begin{aligned}
\int_{-\infty}^{\infty} \left| f_x^{(k)}(s) \right| \mathrm{d}s &\leq \int_{-\infty}^{\infty} \left| \left(\sum_{j=0}^{k} c_{k,j} (-xe^s)^j \right) \right| e^{-xe^s+s} \, \mathrm{d}s \\
&\overset{t=xe^s}{=} x^{-1} \cdot \int_0^{\infty} \left| \left(\sum_{j=0}^{k} c_{k,j} (-t)^j \right) \right| e^{-t} \, \mathrm{d}t \\
&\overset{\text{Fact 12.5}}{\leq} x^{-1} \cdot (k+1)^{k+1} \left(\int_0^1 e^{-t} \, \mathrm{d}t + \int_1^{\infty} t^k e^{-t} \, \mathrm{d}t \right) \\
&\leq \quad x^{-1} \cdot (k+1)^{k+1} \cdot (1+k!) \leq 2x^{-1} \cdot e^k (k+1)^{2k},
\end{aligned}$$

where the last inequality uses $k+1 \leq e^k$ and $1+k! \leq 2(k+1)^k$. □

Notes

The result in this section implies that for any matrix A and a vector v, if we have a fast algorithm for approximating $e^{-tA}v$, for any $t > 0$, then we can obtain a fast algorithm to approximate $A^{-1}v$, or equivalently, to approximately solve the linear system $Ax = v$. We believe this is a promising approach towards constructing fast linear system solvers. In a recent paper, Chung and

Graham [18] give a linear system solver based on a similar approach. However, their notion of a δ-approximate solution is significantly weaker (additive error in each coordinate), and their solver has a $\text{poly}(1/\delta)$ dependence on the error parameter δ, making it unsuitable for several applications.

References

[1] S. Aaronson. The polynomial method in quantum and classical computing. In *Foundations of Computer Science, 2008. FOCS '08. IEEE 49th Annual IEEE Symposium on*, pages 3–3, 2008.

[2] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions*. Dover, New York, fifth edition, 1964.

[3] N. Alon and V. D. Milman. λ_1, isoperimetric inequalities for graphs, and superconcentrators. *J. Comb. Theory, Ser. B*, 38(1):73–88, 1985.

[4] J.-E. Andersson. Approximation of e^{-x} by rational functions with concentrated negative poles. *Journal of Approximation Theory*, 32(2):85 – 95, 1981.

[5] S. Arora, E. Hazan, and S. Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(6):121–164, 2012.

[6] S. Arora and S. Kale. A combinatorial, primal-dual approach to semidefinite programs. In *STOC*, pages 227–236, 2007.

[7] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials. *J. ACM*, 48(4):778–797, July 2001.

[8] R. Beigel, N. Reingold, and D. Spielman. PP is closed under intersection. *Journal of Computer and System Sciences*, 50(2):191 – 202, 1995.

[9] G. Beylkin and L. Monzón. On approximation of functions by exponential sums. *Applied and Computational Harmonic Analysis*, 19(1):17 – 48, 2005.

[10] G. Beylkin and L. Monzón. Approximation by exponential sums revisited. *Applied and Computational Harmonic Analysis*, 28(2):131 – 149, 2010.

[11] É. Borel. *Lecons sur les Fonctions de Variables Réelles et les Développements en Séries de Polynomes*. Gauthier-Villars, Paris (2nd edition, 1928), 1905.

[12] M. Bun and J. Thaler. Dual lower bounds for approximate degree and Markov-Bernstein inequalities. In *Automata, Languages, and Programming*, volume 7965 of *Lecture Notes in Computer Science*, pages 303–314. Springer Berlin Heidelberg, 2013.

[13] P. L. Chebyshev. Théorie des mécanismes connus sous le nom de parallélogrammes. *Mém. Acad. Sci. Pétersb.*, 7:539–568, 1854.

[14] P. L. Chebyshev. Sur les questions de minima qui se rattachent à la représentation approximative des fonctions. *Mém. Acad. Sci. Pétersb.*, 7:199–291, 1859.

[15] J. Cheeger. A lower bound for the smallest eigenvalue of the Laplacian. *Problems Anal.*, pages 195–199, 1970.

[16] E. W. Cheney. *Introduction to approximation theory*. McGraw-Hill, New York, 1966.

[17] F. Chung. *Spectral Graph Theory (CBMS Regional Conference Series in Mathematics, No. 92)*. American Mathematical Society, 1997.

[18] F. Chung and O. Simpson. Solving linear systems with boundary conditions using heat kernel pagerank. In *Algorithms and Models for the Web Graph*, volume 8305 of *Lecture Notes in Computer Science*, pages 203–219. Springer International Publishing, 2013.

[19] W. J. Cody, G. Meinardus, and R. S. Varga. Chebyshev rational approximations to e^{-x} in $[0,\infty)$ and applications to heat-conduction problems. *Journal of Approximation Theory*, 2(1):50 – 65, 1969.

[20] A. A. Gonchar and E. A. Rakhmanov. On convergence of simultaneous Padé approximants for systems of functions of Markov type. *Proc. Steklov Inst. Math.*, 157:31–50, 1983.

[21] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 1994.

[22] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, STOC '96, pages 212–219, New York, NY, USA, 1996. ACM.

[23] N. J. A. Harvey, J. Nelson, and K. Onak. Sketching and streaming entropy via approximation theory. In *Foundations of Computer Science, 2008. FOCS '08. IEEE 49th Annual IEEE Symposium on*, pages 489–498, Oct 2008.

[24] M. R. Hestenes and E. Stiefel. Methods of Conjugate Gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49:409–436, December 1952.

[25] M. Hochbruck and C. Lubich. On Krylov subspace approximations to the matrix exponential operator. *SIAM J. Numer. Anal.*, 34(5):1911–1925, October 1997.

[26] R. Jain, Z. Ji, S. Upadhyay, and J. Watrous. QIP=PSPACE. *J. ACM*, 58(6):30:1–30:27, December 2011.

[27] R. Jain, S. Upadhyay, and J. Watrous. Two-message quantum interactive proofs are in PSPACE. In *Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '09, pages 534–543, 2009.

[28] R. Jain and J. Watrous. Parallel approximation of non-interactive zero-sum quantum games. *2012 IEEE 27th Conference on Computational Complexity*, 0:243–253, 2009.

[29] S. Kale. Efficient algorithms using the multiplicative weights update method. Technical report, Princeton University, Department of Computer Science, 2007.

[30] N. N. Kalitkin and I. A. Panin. On the computation of the exponential integral. *Mathematical Models and Computer Simulations*, 1(1):88–90, 2009.

[31] S. Kaniel. Estimates for some computational techniques in linear algebra. *Math. Comp.*, 20:369–378, 1966.

[32] J. A. Kelner, L. Orecchia, A. Sidford, and Z. A. Zhu. A simple, combinatorial algorithm for solving SDD systems in nearly-linear time. In *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing*, STOC '13, pages 911–920. ACM, 2013.

[33] A. R. Klivans and R. Servedio. Learning DNF in time $2^{\widetilde{O}(n^{1/3})}$. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 258–265. ACM, 2001.

[34] I. Koutis, G. L. Miller, and R. Peng. A nearly- $m \log n$ time solver for SDD linear systems. In *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, FOCS '11, pages 590–598. IEEE Computer Society, 2011.

[35] J. Kuczyński and H. Woźniakowski. Estimating the largest eigenvalues by the power and Lanczos algorithms with a random start. *SIAM J. Matrix Anal. Appl.*, 13(4):1094–1122, October 1992.

[36] C. Lanczos. Solution of systems of linear equations by minimized iterations. *J. Res. Natl. Bur. Stand*, 49:33–53, 1952.

[37] D. A. Levin, Y. Peres, and E. L. Wilmer. *Markov chains and mixing times*. American Mathematical Society, 2006.

[38] A. A. Markov. On a question by D.I. mendeleev. *Zapiski Imperatorskoi Akademii Nauk*, 62:1–24, 1890.

[39] V. A. Markov. On functions deviating least from zero in a given interval. *Izdat. Imp. Akad. Nauk, St. Petersburg*, pages 218–258, 1892.

[40] M. Mitzenmacher and E. Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press, New York, NY, USA, 2005.

[41] Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. In *Soviet Mathematics Doklady*, volume 27, pages 372–376, 1983.

[42] Y. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer, 2004.

[43] D. J. Newman. Rational approximation to $|x|$. *Michigan Math. J.*, 11:11–14, 1964.

[44] D. J. Newman. Rational approximation to e^{-x}. *Journal of Approximation Theory*, 10(4):301 – 303, 1974.

[45] D. J. Newman. Approximation to x^n by lower degree rational functions. *Journal of Approximation Theory*, 27(3):236 – 238, 1979.

[46] N. Nisan and M. Szegedy. On the degree of Boolean functions as real polynomials. *Computational Complexity*, 4(4):301–313, 1994.

[47] L. Orecchia. *Fast Approximation Algorithms for Graph Partitioning using Spectral and Semidefinite-Programming Techniques*. PhD thesis, EECS Department, University of California, Berkeley, May 2011.

[48] L. Orecchia, S. Sachdeva, and N. K. Vishnoi. Approximating the exponential, the Lanczos method and an $\widetilde{O}(\mathrm{m})$-time spectral algorithm for Balanced Separator. *CoRR*, abs/1111.1491, 2011.

[49] L. Orecchia, S. Sachdeva, and N. K. Vishnoi. Approximating the exponential, the Lanczos method and an $\widetilde{O}(\mathrm{m})$-time spectral algorithm for Balanced Separator. STOC '12, pages 1141–1160, 2012.

[50] L. Orecchia, L. J. Schulman, U. V. Vazirani, and N. K. Vishnoi. On partitioning graphs via single commodity flows. In *STOC '08: Proc. 40th Ann. ACM Symp. Theory of Computing*, pages 461–470, 2008.

[51] L. Orecchia and N. K. Vishnoi. Towards an SDP-based approach to spectral methods: A nearly-linear-time algorithm for graph partitioning and decomposition. In *SODA'11: Proc. 22nd Ann. ACM-SIAM Symp. Discrete Algorithms*, pages 532–545, 2011.

[52] V. Y. Pan and Z. Q. Chen. The complexity of the matrix eigenproblem. In *STOC'99*, pages 507–516, 1999.

[53] B. N. Parlett. *The symmetric eigenvalue problem*, volume 7. SIAM, 1980.

[54] T. J. Rivlin. *An introduction to the approximation of functions*. Blaisdell book in numerical analysis and computer science. Blaisdell Pub. Co., 1969.

[55] W. Rudin. *Principles of mathematical analysis*. McGraw-Hill Book Co., New York, third edition, 1976. International Series in Pure and Applied Mathematics.

[56] Y. Saad. On the rates of convergence of the Lanczos and the block-Lanczos methods. *SIAM Journal on Numerical Analysis*, 17(5):pp. 687–706, 1980.

[57] Y. Saad. *Numerical methods for large eigenvalue problems*. Society for Industrial and Applied Mathematics, 2011.

[58] Y. Saad and H. A. van der Vorst. Iterative solution of linear systems in the 20th century. *Journal of Computational and Applied Mathematics*, 123(1-2):1–33, 2000. Numerical Analysis 2000. Vol. III: Linear Algebra.

[59] E. B. Saff, A. Schönhage, and R. S. Varga. Geometric convergence to e^{-z} by rational functions with real poles. *Numerische Mathematik*, 25:307–322, 1975.

[60] A. Schönhage. Zur rationalen Approximierbarkeit von e^{-x} über $[0,\infty)$. *Journal of Approximation Theory*, 7(4):395 – 398, 1973.

[61] A. Sherstov. Lower bounds in communication complexity and learning theory via analytic methods. Technical report, University of Texas at Austin, 2009.

[62] D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *STOC*, pages 81–90, New York, NY, USA, 2004. ACM.

[63] G. Szegö. Über eine Eigenschaft der Exponentialreihe. *Sitzungsber. Berl. Math. Ges.*, 23:50–64, 1924.

[64] G. Szegö. *Orthogonal polynomials*. American Mathematical Society Providence, 4th ed. edition, 1939.

[65] G. Valiant. Finding correlations in subquadratic time, with applications to learning parities and juntas. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 11–20. IEEE, 2012.

[66] N. K. Vishnoi. *$Lx = b$. Foundations and Trends in Theoretical Computer Science*, 8(1-2):1–141, 2012.

[67] K. Weierstrass. Über die analytische Darstellbarkeit sogenannter willkürlicher Functionen einer reellen veränderlichen. *Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften zu Berlin*, 2:633–639, 1885.

[68] S. M. Zemyan. On the zeroes of the Nth partial sum of the exponential series. *The American Mathematical Monthly*, 112(10):pp. 891–909, 2005.