Approximation Theory and the Design of Fast Algorithms

Sushant Sachdeva∗ Nisheeth K. Vishnoi†

Abstract

We survey key techniques and results from approximation theory in the context of uniform approximations to real functions such as $e^{-x}$, $1/x$, and $x^k$. We then present a selection of results demonstrating how such approximations can be used to speed up primitives crucial for the design of fast algorithms for problems such as simulating random walks, graph partitioning, solving linear system of equations, computing eigenvalues and combinatorial approaches to solve semi-definite programs.

Contents

1 Introduction

2 Basics
  2.1 Uniform Approximations ....................................................... 2
  2.2 Chebyshev Polynomials ....................................................... 7

3 Polynomial Approximations
  3.1 Approximating $x^s$ on $[-1,1]$ .................................................. 8
  3.2 Approximating $e^{-x}$ on $[0,b]$ ............................................. 9
  3.3 Markov’s Theorem and Lower Bounds for Polynomial Approximations ............................................. 11

4 Rational Approximations
  4.1 Approximating $e^{-x}$ on $[0,\infty)$ ........................................... 13
  4.2 Approximating $e^{-x}$ on $[0,\infty)$ with Negative Poles ................. 14

5 Approximating $x^{-1}$ Using Exponentials ...................................... 17
  5.1 Bernoulli Numbers and the Euler-Maclaurin Formula ........................... 17

6 Applications
  6.1 Matrices and Graphs .............................................................. 19
  6.2 Simulating Random Walks and Finding Sparse Cuts .......................... 21
    6.2.1 Quadratically Faster Random Walks .................................. 20
    6.2.2 Finding Sparse Cuts ...................................................... 21
  6.3 Solving Linear Equations ...................................................... 22
    6.3.1 A Gradient Descent Based Linear Solver .............................. 23
    6.3.2 The Conjugate Gradient Method ...................................... 23
  6.4 Computing Eigenvalues via the Lanczos Method ............................. 25
  6.5 Computing the Matrix Exponential .......................................... 27
  6.6 Matrix Inversion via Exponentiation ....................................... 29

∗Research Fellow, Simons Institute for the Theory of Computing. UC Berkeley, USA. Part of this work was done when this author was a graduate student at the Department of Computer Science, Princeton University. Email: sachdeva@eecs.berkeley.edu
†Microsoft Research, Bangalore, India. Email: nisheeth.vishnoi@gmail.com
1 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 \( f : \mathbb{R} \rightarrow \mathbb{R} \), and an interval \( I \), what is the closest a degree \( d \) polynomial can remain to \( f(x) \) in the entire interval? Formally, if \( \Sigma_d \) is the class of all univariate real polynomials of degree at most \( d \), the goal is to understand

\[
\varepsilon_{f,I}(d) \equiv \inf_{p \in \Sigma_d} \sup_{x \in I} |f(x) - p(x)|.
\]

This notion of approximation, sometimes called Chebyshev approximation, is attributed to Pafnuty Chebyshev, who essentially started 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 computed the best degree \( d - 1 \) polynomial to approximate the monomial \( x^d \), see [14]. The result equivalently showed that any degree \( d \) polynomial with leading coefficient 1 cannot come more than \( 1/2^d \) close to 0 everywhere in the interval \([-1, 1]\). Moreover, he showed that the degree \( d \) polynomial that arises when one writes \( \cos(d \theta) \) as a polynomial in \( \cos \theta \) achieves this bound. These polynomials are called Chebyshev polynomials which find use in several different areas of science and mathematics and, indeed, repeatedly make an appearance in this survey because of their extremal properties.

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 \( \varepsilon_{d}(f, I) \) does not increase as we increase \( d \), it was Weierstrass [64] who later established that, for any continuous function \( f \) and a bounded interval \( I \), \( \varepsilon_{f,I}(d) \to 0 \) as \( d \to \infty \). 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 [35], motivated by a question in chemistry due to Mendeleev, proved that the absolute value of the derivative of a degree \( d \) polynomial which is bounded by 1 in the interval \([-1, 1]\) cannot exceed \( d^2 \). These, and several 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 come up with 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 the best polynomial (or rational) approximation for it 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.

Among all this activity, an important development happened in the 1960s when Donald Newman [39] 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 only achieves an error of \( \Theta(1/d) \). Though rational functions had also been considered by Chebyshev, it was Newman’s result that revived the area of uniform approximation with rational functions and led to several 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 [18]), was to understand the best rational approximation to \( e^{-x} \) over the interval \([0, \infty)\). Note that \( e^{-x} \) goes to 0 as \( x \) goes to infinity, while any polynomial must necessarily go to infinity. Rational functions of degree \( d \) were shown to approximate \( e^{-x} \) on \([0, \infty)\) up to an error of \( c^d \) for \( c < 1 \). This line of research culminated in a landmark result by Gonchar and Rakhmanov [19] who determined the optimal \( c \). Despite remarkable progress in the theory of approximation by rational functions, unfortunately,
there seems to be no clear understanding of why rational approximations are often significantly better than polynomials of the same degree, and this area seems to be flooded with many surprising results often proven using clever tricks. Perhaps, this is what makes the area of rational approximations promising and worth understanding; it seems capable of magic.

**Approximation theory in algorithms and complexity.** Two of the first applications of approximation theory in algorithms were the Conjugate Gradient method \[22\] and the Lanczos method \[33\], which are used to solve linear systems of 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 (e.g., singular value/singular vector) calculations. These methods are iterative and reduce such computations to a small number of calculations 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 \[53\] for a detailed discussion.

Until recently, the main applications of approximation theory in theoretical computer science have been in complexity theory: one of the first being a result of Beigel et al. \[8\] who used Newman’s result on rational approximations to show that the complexity class PP is closed under intersections and unions.\(^2\) Another important result where approximation theory, in particular Chebyshev polynomials, played a role is the quadratic speed-up for quantum search algorithms, initiated with a work by Grover \[21\]. The fact that one cannot speed up beyond Grover’s result was showed 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 \[41\]. For more on applications of approximation theory to complexity theory, communication complexity and computational learning theory, we refer the reader to the bibliography by Aaronson \[1\] and the thesis by Sherstov \[59\].

In this survey, we present applications of approximation theory to the design of fast algorithms. We show how to compute good approximations to matrix-vector products such as \(A^iv\), \(A^{-1}v\) and \(\exp(-A)v\) for any matrix \(A\) and a vector \(v\). Such primitives are useful for performing several fundamental computations quickly, such as random walk simulation, graph partitioning, solving linear system of equations, and combinatorial approaches to solve semi-definite programs. The algorithms for computing these primitives end up performing 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.

As a simple but important application, we show how to speed up the computation of \(A^iv\) 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 \(A^iv\) 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 there is a polynomial \(p\) of degree \(d \approx \sqrt{s}\log 1/\delta\) that \(\delta\)-approximates \(x^s\) over \([-1,1]\). A straightforward diagonalization argument then implies that \(\|A^iv - \sum_{i=0}^d a_ia^iv\|_2 \leq \delta\), where \(p(x) = \sum_{i=0}^d a_ix^i\). More importantly, the time it takes to

\(^1\)More precisely, in the area of numerical linear algebra, since algorithms was not yet established as a field.

\(^2\)PP is the complexity class that contains sets which are accepted by a polynomial-time bounded probabilistic Turing machine that accepts with probability strictly more than 1/2.

\(^3\)Recall that the matrix exponential is defined to be \(\exp(A) \overset{\text{def}}{=} \sum_{k \geq 0} A^k/k!\).
compute $\sum_{i=0}^{d} a_i A^i v$ is $O(md) = O(m\sqrt{s\log 1/\delta})$, 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 $v$ 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]. At the heart of all algorithms based on the matrix multiplicative weights update method is a procedure to quickly compute $e^{-A}v$ 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 the interval $[0, |A|]$. A simple approach is to truncate the Taylor series expansion of $e^{-x}$. It is easy to show that using roughly $|A| + \log 1/\delta$ terms in the expansion suffices to obtain a $\delta$ approximation. However, we can use a polynomial approximation result for $e^{-x}$ over the interval $[0, |A|]$ to produce an algorithm that runs in time roughly $O(m\sqrt{|A|})$. In fact, when $A$ has more structure, we can go beyond the square-root barrier.

For fast graph algorithms, often the quantity of interest is $e^{-L}v$, where $L$ is the combinatorial Laplacian of a graph, and $v$ is a vector. The vector $e^{-L}v$ can also be interpreted as the resulting distribution of a certain continuous-time random walk on the graph with starting distribution $v$. Appealing to a rational approximation to $e^{-x}$ with negative poles, the computation of $e^{-L}v$ can be reduced to a small number of computations of the form $L^{-1}u$. Thus, using the near-linear-time Laplacian solver due to Spielman and Teng [60], this gives an $O(m)$-time algorithm for approximating $e^{-L}v$ 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 that arises from our last application is whether the Spielman-Teng result is necessary in order to compute $e^{-L}v$ in near-linear time. In our final application of approximation theory, we answer this question in the affirmative by presenting a reduction in the other direction; we prove that the inverse of a positive-definite matrix can be approximated by a weighted-sum of a small number of matrix exponentials.

**Organization.** The goal of this survey is to bring out how classical and modern results from approximation theory play a crucial role in obtaining results which are relevant today to the emerging theory of fast algorithms. The approach we have taken is to first present the ideas and results from approximation theory that we think are central, elegant, and have wider applicability in TCS. For the sake of clarity, we have sometimes sacrificed tedious details. This means that we admittedly either do not present complete proofs or theorems with optimal parameters for a few important results. The survey is organized as follows.

In Section[2], we present some essential notations and results from approximation theory, and introduce Chebyshev polynomials. We prove certain extremal properties of these polynomials which are used in this survey. In Section[3] we construct polynomial approximations to the monomial $x^d$ over the interval $[-1,1]$ and $e^{-x}$ over the interval $[0,b]$. Both results are based on Chebyshev polynomials. In the same Section we prove a special case of Markov’s theorem which is then used to show that these polynomial approximations are asymptotically optimal.

---

4See also [24] [25] [26] [27] [45] [46] [63] [5] [57] [58].

5A Laplacian solver is an algorithm that (approximately) solves a given system of linear equations $Lx = b$, where $L$ is a graph Laplacian and $b \in \text{Im}(L)$, i.e., it (approximately) computes $L^{-1}b$, see [63].
In Section 4 we consider rational approximations for the function $e^{-x}$ over the interval $[0, \infty)$. 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 4.2 we present a proof of the remarkable 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.

In Section 5 we show how $x^{-1}$ can be approximated by a sparse sum of the form $\sum_i w_i e^{-b_i x}$ over the interval $[0, 1]$. The proof relies on the Euler-Maclaurin formula and certain bounds derived from the Riemann zeta function.

Section 6 contains the presentation of applications of the approximation theory results. In Section 6.2 we show how the results of Section 3 imply that we can quadratically speed up random walks in graphs, and find sparse cuts faster. Here, we discuss the important issue of computing the coefficients of the polynomials in Section 5. In Section 6.3 we present the famous Conjugate gradient method for solving symmetric PSD systems of equations $Ax = v$ iteratively where the number of iterations depend on the square-root of the condition number of $A$. The square-root saving is shown to be exactly because of the scalar approximation result for $x^t$ from Section 3. In Section 6.4 we present the Lanczos method and show how it can be used to compute the largest eigenvalue of a symmetric matrix. We show how the existence of a good approximation for $x^t$, yet again, allows a quadratic speedup over the power method.

In Section 6.5 we show how the polynomial and rational approximations to $e^{-x}$ developed in Section 4 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 $\exp(-A)v$. We also show how to bound and compute the coefficients involved in the rational approximation result in Section 4.2 this is crucial for the application.

Finally, in 6.6 using the result from Section 5 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 $\exp(-A)v$. 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 $A$.

2 Basics

2.1 Uniform Approximations

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 $\mathcal{I}$. Of particular interest is the following notion of approximation.

Definition 2.1. A function $g$ is called a $\delta$-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. Since then, a central topic of study in approximation theory has been to understand how well a function $f$ can be approximated using polynomials. More precisely, 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 basic questions are 1) does $\lim_{d \to \infty} \varepsilon_{f, \mathcal{I}}(d) = 0$? and 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. Later, Weierstrass (see [49]) 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 $\delta$-approximation to $f$ on $[a, b]$. 

5
The existence and uniqueness of a degree-$d$ polynomial that achieves the best approximation $\epsilon_{f,\mathcal{I}}(d)$ was proved by Borel.

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 survey.

In an attempt to get a handle on best approximations, Chebyshev showed that a degree-$d$ polynomial $p$ is a best 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 $f(x_i) - p(x_i) = (-1)^i \epsilon$ where $\epsilon = \sup_{x \in [-1,1]} |f(x) - p(x)|$. We prove the following theorem, attributed to de La Vallee-Poussin, which implies the sufficient side of Chebyshev’s alternation theorem and is often good enough.

**Theorem 2.2.** Suppose $f$ is a function over $[-1,1]$, $p$ is a degree-$d$ polynomial, and $\delta > 0$ is such that the error function $\epsilon \overset{\text{def}}{=} f - p$ assumes alternately positive and negative signs at $d + 2$ increasing points $-1 \leq x_0 < x_1 \cdots < x_{d+1} \leq 1$, and satisfies $|\epsilon(x_i)| \geq \delta$ 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, to 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 $\epsilon(x_i) - \delta < q(x_i) - p(x_i) < \epsilon(x_i) + \delta$. Since $|\epsilon(x_i)| \geq \delta$, the polynomial $q - p$ is non-zero at each of the $x_i$s, and must have the same sign as $\epsilon$. Thus, $q - p$ assumes alternating signs at the $x_i$s, and hence must have a zero between each pair of successive $x_i$s. 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 $2^{-d+1}$, see Theorem 2.6. However, finding the best degree-$d$ polynomial for other functions is usually intractable, and will not be the focus in the survey. Rather, we either find a $\delta$-approximation for a suitably small $\delta$ or prove that there are none.

Often, an effective way to study $\delta$-approximations is to consider relaxations of the problem of finding the best uniform approximation. A natural relaxation to consider is to find the degree-$d$ polynomial $p$ that minimizes the $\ell_2$ error $\int_{-1}^{1} (f(x) - p(x))^2 \, dx$. Algorithmically, we know how to solve this 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 $\int_{-1}^{1} p_i(x) p_j(x) \, dx = 0$ if $i \neq j$ and 1 otherwise. 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 $[-1, 1]$.

Given such an orthonormal basis, the best approximation is given by $p(x) = \sum_i \hat{f}_i p_i(x)$, where $\hat{f}_i = \int_{-1}^{1} f(x) p_i(x) \, dx$.

Given a relaxation, we must consider how good that relaxation is, i.e., if $p(x)$ is the best $\ell_2$-approximation to the function $f(x)$, how does it compare to the best uniform approximation to $f(x)$? For the straightforward relaxation above, the approximation turns out to not be meaningful. However, if we modify the relaxation to minimize the $\ell_2$ error with respect to the weight function $w(x) \overset{\text{def}}{=} 1/\sqrt{1-x^2}$, i.e., minimize $\int_{-1}^{1} (f(x) - p(x))^2 \frac{dx}{\sqrt{1-x^2}}$, then when $f$ is continuous the best degree-$d$ $\ell_2$-approximation with respect to $w$ turns out to be an $O(\log d)$ approximation for the best uniform approximation (see [49], Section 2.4) for a proof). Formally, if we let $p$ be the degree-$d$ polynomial that minimizes $\ell_2$-approximation with respect to $w$, and let $p^*$ be the degree-$d$ polynomial that minimizes $\ell_2$-approximation with respect to $w$, then we have $\|w f - \tilde{p}\|_2 \leq C \epsilon$, where $\tilde{p}$ is the best degree-$d$ polynomial with respect to $w$. 

---

6These orthogonal polynomials are given explicitly by $\left\{ \sqrt{(2d+1)/2} L_d(x) \right\}$, where $L_d(x)$ denotes the degree-$d$ Legendre polynomials. See [61].
The magnitude at exactly degree of alternating signs. The result now follows from Theorem 2.2.

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

### 2.2 Chebyshev Polynomials

There are several ways to define Chebyshev polynomials. For a non-negative integer \( d \), if \( T_d(x) \) denotes the Chebyshev polynomial of degree \( d \), then it can be defined recursively as follows: \( T_0(x) \overset{\text{def}}{=} 1, T_1(x) \overset{\text{def}}{=} x \), and for \( d \geq 2 \),

\[
T_d(x) \overset{\text{def}}{=} 2xT_{d-1}(x) - T_{d-2}(x). \tag{2}
\]

For convenience, we extend the definition of Chebyshev polynomials to negative integers by defining \( T_d(x) \overset{\text{def}}{=} T_{|d|}(x) \) for \( d < 0 \). It is easy to verify that with this definition, the recurrence given by (2) is satisfied for all integers \( d \). Rearranging (2), we obtain the following:

**Proposition 2.3.** 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 2.4.** 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 \( T_0(\theta) = \cos(0) = 1 \) and \( T_1(\theta) = \cos(\theta) = x \). Additionally, \( \cos(d\theta) = 2 \cdot \cos \theta \cdot \cos((d - 1)\theta) - \cos((d - 2)\theta) \) and, hence, the recursive definition for Chebyshev polynomials applies. This proposition also immediately implies that over the interval \([-1, 1]\), the value of any Chebyshev polynomial is bounded by 1 in magnitude.

**Proposition 2.5.** For any integer \( d \), and \( x \in [-1, 1] \), we have \( |T_d(x)| \leq 1 \).

In fact, Proposition 2.4 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. This is ideally suited for an application of Theorem 2.2 and we can now prove Chebyshev’s result mentioned in the previous section.

**Theorem 2.6.** 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., \( \inf_{p_{d-1} \in P_{d-1}} \sup_{x \in [-1, 1]} |x^d - p_{d-1}(x)| = 2^{-d+1} \).

**Proof.** Observe that the leading coefficient of \( T_d(x) \) is \( 2^d-1 \) and, hence, \( 2^{-d+1}T_d(x) - x^d \) is a polynomial of degree \((d - 1)\). The error this polynomial achieves in approximating \( x^d \) on \([-1, 1]\) is \( 2^{-d+1}T_d(x) \), 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 2.2.

The fact that \( T_d(x) \) takes alternating \( \pm 1 \) values \( d + 1 \) times in \([-1, 1]\), leads to another property of the Chebyshev polynomials:
Proposition 2.7. For any degree-d polynomial \( p(x) \) such that \(|p(x)| \leq 1 \) for all \( x \in [-1, 1] \), for any \( y \) such that \(|y| > 1\), we have \(|p(y)| \leq |T_d(y)|\).

Proof. For sake of contradiction, let \( y \) be such that \(|p(y)| > |T_d(y)|\) and let \( q(x) \defeq \frac{T_d(x)}{p(x)} \cdot p(x)\). Hence, \(|q(x)| < |p(x)| \leq 1\) for all \( x \in [-1, 1]\), and \(q(y) = T_d(y)\). Thus, strictly between any two consecutive points where \(T_d(x)\) alternates between +1 and −1, there must be a point \(x_i\) at which \(T_d(x_i) = q(x_i)\) since \(|q(x)| < 1\) in \([-1, 1]\). 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. \(\square\)

This proposition is used to prove a lower bound for rational approximations to \(e^{-x}\) in Section 4.1. In order to do so, we need to upper bound their growth. 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 2.8. For any integer \(d\), and \(x\), we have
\[
T_d(x) = \frac{1}{2} \left( x + \sqrt{x^2 - 1} \right)^d + \frac{1}{2} \left( x - \sqrt{x^2 - 1} \right)^d.
\]

3 Polynomial Approximations

In this section, we use Chebyshev polynomials and their properties to construct polynomial approximations to some fundamental functions such as the monomial \(x^d\), and the exponential function \(e^{-x}\). We also introduce the famous Markov’s theorem, and prove a special case, which is then used to prove lower bounds on the degree of best polynomial approximations.

3.1 Approximating \(x^d\) on \([-1, 1]\)

Recall from Proposition 2.3 that for any \(d\), we can write \(xT_d(x) = \frac{1}{2} \cdot (T_{d-1}(x) + T_{d+1}(x))\). 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 \(x^d\) 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 \(D_s \defeq \sum_{i=1}^s Y_i\) where \(D_0 \defeq 0\).

Lemma 3.1. For any integer \(s \geq 0\), we have \(\mathbb{E}_{Y_1, \ldots, Y_s}[T_{D_s}(x)] = x^s\).

Proof. We proceed by induction. For \(s = 0\), \(D_s = 0\) and, hence, \(\mathbb{E}[T_{D_s}(x)] = T_0(x) = 1 = x^0\). Moreover, for any \(s \geq 0\),
\[
x^{s+1} \overset{\text{Induction}}{=} \mathbb{E}_{Y_1, \ldots, Y_s} \left[ x \cdot T_{D_s}(x) \right] = \mathbb{E}_{Y_1, \ldots, Y_s} \left[ x \cdot \mathbb{E}_{Y_1} \left[ T_{D_s}(x) \right] - \mathbb{E}_{Y_1, \ldots, Y_s} \left[ T_{D_s}(x) \right] \right] \overset{\text{Prop. 2.3}}{=} \mathbb{E}_{Y_1, \ldots, Y_s, Y_{s+1}} \left[ \frac{T_{D_{s+1}}(x) + T_{D_{s-1}}(x)}{2} \right] = \mathbb{E}_{Y_1, \ldots, Y_s, Y_{s+1}} \left[ T_{D_{s+1}}(x) \right].
\]
Lemma 3.1 allows us to obtain polynomials that approximate \( x^d \), but have degree close to \( \sqrt{s} \). The main observation is that the probability that \( |D_s| \geq d \) is small. In particular, using Chernoff bounds, the probability that \( |D_s| > \sqrt{2s \log \frac{2}{\delta}} \) is at most \( \delta \). Moreover, since \( |T_{D_s}(x)| \leq 1 \) for all \( x \in [-1, 1] \), we can ignore all terms with degree greater than \( d \) without incurring an error greater than \( \delta \). We now prove this formally.

Let \( 1_{|D_s| \leq d} \) denote the indicator variable for the event that \( |D_s| \leq d \). Our polynomial of degree \( d \) approximating \( x^d \) is obtained by truncating the above expansion to degree \( d \), i.e.,

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

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

**Theorem 3.2.** For any positive integers \( s, d \), the degree-\( d \) polynomial \( p_{s,d} \) satisfies

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

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

**Proof.** Using Chernoff bounds (see [38, Chapter 4]), we know that

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

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

\[
\sup_{x \in [-1, 1]} |p_{s,d}(x) - x^d| \overset{\text{Lem. 3.1}}{=} \sup_{x \in [-1, 1]} \left| \mathbb{E}_{Y_1, \ldots, Y_s} \left[ T_{D_s}(x) \cdot 1_{|D_s| > d} \right] \right|
\]

\[
\leq \sup_{x \in [-1, 1]} \mathbb{E}_{Y_1, \ldots, Y_s} \left[ |T_{D_s}(x)| \cdot 1_{|D_s| > d} \right]
\]

\[
\leq \mathbb{E}_{Y_1, \ldots, Y_s} \left[ 1_{|D_s| > d} \cdot \sup_{x \in [-1, 1]} |T_{D_s}(x)| \right] \overset{\text{Prop. 2.5}}{=} \mathbb{E}_{Y_1, \ldots, Y_s} \left[ 1_{|D_s| > d} \right] \leq 2e^{-d^2/2s},
\]

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

Over the next several sections, we explore several interesting consequences of this seemingly simple approximation. In Section 3.2 we use this approximation to give improved polynomial approximations to the exponential function. In Sections 6.3 and 6.4 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 3.3.

### 3.2 Approximating \( e^{-x} \) on \([0, b]\)

In this section we consider the problem of approximating \( e^{-x} \) over the interval \([0, \infty)\). The first problem one faces when one looks for polynomial approximations for \( e^{-x} \) over \([0, \infty)\) is that none exist. The reason is that a polynomial goes to infinity with \( x \), while \( e^{-x} \) goes to 0. However, if one restricts to approximating \( e^{-x} \) over an interval \([0, b]\), then a simple approach 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 \( \delta \) approximation. We show that the approximation for \( x^d \) we developed in the previous section allows us to obtain a quadratic improvement over this simple approximation.
**Theorem 3.3.** For every $0 < b$, 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 $O\left(\sqrt{\max\{b, \log 1/\delta\} \cdot \log 1/\delta}\right)$.

After a scaling and translation, it suffices to approximate the function $e^{-\lambda x - \lambda}$ over the interval $[-1, 1]$, where $\lambda = b/2$. As mentioned before, if we truncate its Taylor expansion, we obtain $\sum_{i=0}^{\infty} e^{-\lambda} \frac{(-\lambda)^i}{i!} x^i$ as a candidate approximating polynomial. Our 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 the previous section. Formally,

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

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

**Lemma 3.4.** For every $\lambda > 0$ and $\delta \in (0, 1/2]$, we can choose $t = O(\max\{\lambda, \log 1/\delta\})$, and $d = O\left(\sqrt{\log 1/\delta}\right)$ such that the polynomial $q_{\lambda, i, d}$ defined above, $\delta$-approximates the function $e^{-\lambda - \lambda x}$ over the interval $[-1, 1]$, i.e.,

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

**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.

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

$$\leq \sum_{i=0}^{t} e^{-\lambda} \frac{\lambda^i}{i!} \sup_{x \in [-1, 1]} |x^i - p_{i, d}(x)| + e^{-\lambda} \sum_{i=t+1}^{\infty} \frac{\lambda^i}{i!}.$$

From Theorem 3.2, we know that $p_{i, d}$ is a good approximation to $x^i$, and we can use it to bound the first error term.

$$\sum_{i=0}^{t} e^{-\lambda} \frac{\lambda^i}{i!} \sup_{x \in [-1, 1]} |x^i - p_{i, d}(x)| \leq \sum_{i=0}^{t} e^{-\lambda} \frac{\lambda^i}{i!} \cdot 2e^{-d^2/2}$$

$$\leq 2e^{-d^2/2} \cdot \sum_{i=0}^{\infty} e^{-\lambda} \frac{\lambda^i}{i!} = 2e^{-d^2/2}.$$

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

$$e^{-\lambda} \sum_{i=t+1}^{\infty} \frac{\lambda^i}{i!} \leq e^{-\lambda} \sum_{i=t+1}^{\infty} \left(\frac{\lambda e^2}{i}\right)^i \leq e^{-\lambda} \sum_{i=t+1}^{\infty} e^{-i} \leq e^{-\lambda - t}.$$

Thus, if we let $t = \left[\max\{\lambda e^2, \log 2/\delta\}\right]$ and $d = \left[\sqrt{2\log 4/\delta}\right]$, combining the above and using $\lambda > 0$, we obtain $\sup_{x \in [-1, 1]} |e^{-\lambda x} - q_{\lambda, i, d}(x)| \leq 2e^{-d^2/2} + e^{-\lambda - t} \leq \delta/2 + \delta/2 \leq \delta$. $\square$

Now, we can complete the proof of Theorem 3.3.
Proof of Theorem 3.3 Let \( \lambda \overset{\text{def}}{=} b/2 \), and let \( t \) and \( d \) be given by Lemma 3.4 for the given value of \( \delta \). Define \( r_{b,\delta} \overset{\text{def}}{=} q_{\lambda,t,d}(1/\lambda \cdot (x-b)) \), where \( q_{\lambda,t,d} \) is the polynomial given by Lemma 3.4. Then,

\[
\sup_{x \in [0,b]} |e^{-x} - r_{b,\delta}(x)| = \sup_{x \in [0,b]} |e^{-x} - q_{\lambda,t,d}(1/\lambda \cdot (x-b))| = \sup_{z \in [-1,1]} |e^{-\lambda z - \lambda} - q_{\lambda,t,d}(z)| \leq \delta,
\]

where the last inequality follows from the guarantee of Lemma 3.4. The degree of \( r_{b,\delta}(x) \) is the same as that of \( q_{\lambda,t,d}(x) \), i.e., \( d = O\left(\sqrt{\max\{b, \log 1/\delta\} \cdot \log 1/\delta}\right) \).

\( \square \)

Theorem 3.3 is implicit in the work by Hochbruck and Lubich [23]. A weaker version of this theorem has also been proved in [44].

### 3.3 Markov’s Theorem and Lower Bounds for Polynomial Approximations

In this section, we prove that the bounds in the previous section are essentially optimal. Specifically, we show that the degrees of the approximating polynomials require \( \sqrt{s} \) dependence for approximating \( x^t \) on \([-1,1]\), and a \( \sqrt{b} \) dependence for approximating \( e^{-x} \) on \([0,b] \). Such lower bounds often use the following well known Markov’s theorem from approximation theory.

**Theorem 3.5** (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, for all \( x \in [-1,1] \), the derivative of \( p \), \( p^{(1)} \) satisfies \( |p^{(1)}(x)| \leq d^2 \).

In fact, the above theorem is another example of an extremal property of the Chebyshev polynomials since they can be seen to be a tight example for this theorem. The above theorem also generalizes to higher derivatives, where it implies that for any such \( p \), we have \( |p^{(k)}(x)| \leq \sup_{y \in [-1,1]} |T_d^{(k)}(y)| \), for any \( k \) and \( x \in [-1,1] \) [49, Section 1.2]. This was proved by V. A. Markov [16].

We sketch a proof of the following special case of Markov’s theorem, based on the work by Bun and Thaler [12], that suffices for proving our lower bounds.

**Lemma 3.6.** For any degree-\( d \) polynomial \( q \) such that \( |q(x)| \leq 1 \) for all \( x \in [-1,1] \), \( |q^{(1)}(1)| \leq d^2 \).

We now present the main idea behind both of the lower bound proofs. Say \( p(x) \) is the approximating polynomial. We start by using the bound on the approximation error to bound the range of values taken by the polynomial in the interval. 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 \( p(t) \) to \( q(1) \). In order to show the existence of a point with a large derivative, we use the Mean Value theorem and the fact that our polynomial is a good approximation to the function of interest. 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 3.7.** 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 \( \delta \), 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 \( \delta \). 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 \( \inf_{x \in [0,b]} |p(x)| \geq e^{-\delta} \).

Assume that \( \delta \leq 1/8 \), and \( b \geq 5 > 3 \log_e 4 \). Applying the Mean Value theorem (see [50, Chapter 5]) on the interval \([0,\log_e 4]\), we know that there exists \( t \in [0,\log_e 4] \), such that

\[
|p^{(1)}(t)| = \left| \frac{p(\log_e 4) - p(0)}{\log_e 4} \right| \geq \frac{(1 - \delta) - (e^{-\log_e 4} + \delta)}{\log_e 4} \geq \frac{1}{2\log_e 4}
\]
Consider the polynomial \( q(x) = \frac{1}{1+2\delta} \left( 2p \left( \frac{(1+1-x)+b(1-x)}{2} \right) - 1 \right) \). Since \( p([0,b]) \subseteq [-\delta, 1+\delta] \), we obtain \(|q(x)| \leq 1\) for all \( x \in [-1,1] \). Thus, using Lemma 3.6 we obtain \(|q^{(1)}(1)| \leq d^2\). This implies that \(|q^{(1)}(1)| = (b-t) |p^{(1)}(t)| \leq d^2(1+2\delta)\). Plugging in the lower bound on \(|p^{(1)}(t)|\) proved above and rearranging, we obtain \(d \geq \sqrt{\frac{b-t}{2\delta / \log 4}} \geq \frac{1}{3} \sqrt{b} \), 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 \( t \in [1-1/\delta, 1] \) such that \(|p^{(1)}(t)| \geq \Omega(s)\) (assuming \( \delta \) small enough). The lower bound now follows immediately by applying Lemma 3.6 to the polynomial \( (1+\delta \cdot p(tx)) \). Now, we give a proof of the special case of Markov’s theorem given by Lemma 3.6.

**Proof of Lemma 3.6** 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 \( c_j \) are variables and there are an infinite number of constraints:

\[
\max c_1 \quad \text{s.t.} \quad \sum_{i=0}^{d} c_j(x-1)^i \leq 1 \quad \forall x \in [-1,1].
\]

Since \((-c_j)\) is a feasible solution whenever \((c_j)\) is, 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 \( k \) between 0 and \( d \): i.e., \( \max c_1 \) subject to \( \sum_{i=0}^{d} c_i(x-1)^i \leq 1 \) for \( x = \cos(k\pi/d) \) with even \( k \) and \( \sum_{i=0}^{d} c_i(x-1)^i \geq -1 \) for \( x = \cos(k\pi/d) \) 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. We can write the dual to the restricted linear program as follows:

\[
\min \sum_{i=0}^{d} y_i \quad \text{s.t.} \quad Ay = e_1 \quad \text{and} \quad y_j \geq 0 \quad \forall j.
\]

Here \( e_1 \in \mathbb{R}^{d+1} \) is the vector \((0,1,0,\ldots,0)\)\(^\top\), and \( A \) is the matrix defined by \( A_{ij} = \frac{(-1)^j (\cos(j\pi/d) - 1)^i}{2^i} \), where \( i = 0, \ldots, d \), and \( j = 0, \ldots, d \). Using elementary trigonometric identities (see [12]), one can show that

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

is, in fact, the unique solution to \( Ay = e_1 \), and satisfies \( \sum y_i = d^2 \). It trivially satisfies the positivity constraints and, hence, by weak duality implies an upper bound of \( d^2 \) on the optimum value of primal linear program.

### 4 Rational Approximations

In this section we introduce approximations to functions by rational functions such as \( p(x) / q(x) \) where \( p, q \) are polynomials. The error in approximation is again measured as the worst error in the interval of interest and we would be interested in trade-off between the error and the maximum of the degrees of \( p, q \). The surprising power of rational approximations was first demonstrated by Newman [39] who showed that rational approximations can be much more powerful than polynomial approximations. He proved that the best degree-\( d \) rational approximation to the function \(|x|\) on \([-1,1]\) achieves an approximation error of \( e^{-\Theta(\sqrt{d})} \). Contrast
this with the fact that the best degree-$d$ polynomial approximation to $|x|$ on $[-1, 1]$ only achieves an error of $\Theta(1/d)$.

Unlike the lower bound results proved in the previous section, we show that rational functions can provide approximations to $e^{-x}$ that hold for all $x \geq 0$, and achieve an approximation error that is exponentially small in their degree. We also show how to construct such rational approximations which in addition have additionally small in degree. We also show how to construct such rational approximations which in addition have negative poles. Such rational functions are extremely useful in applications, see Section 6.5.

4.1 Approximating $e^{-x}$ on $[0, \infty)$

In this section we show that, somewhat surprisingly, there exist simple rational functions of the form $\frac{1}{p(x)}$, 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 $\frac{1}{p(x)}$ can do much better.

Upper Bound

In the last section, we showed that the partial sums of the Taylor series expansion of $e^{-x}$ requires 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 $e^x = \frac{1}{e^{-x}}$ to degree $d$, and take its reciprocal, we can approximate $e^{-x}$ on $[0, \infty)$ up to $2^{-\Omega(d)}$ error. We let $S_d(x) \equiv \sum_{k=0}^{d} \frac{x^k}{k!}$.

**Theorem 4.1.** 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 $O(\log \frac{1}{\delta})$ that is a $\delta$-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, $\frac{1}{S_d(x)} - e^{-x} \geq 0$. We divide $[0, \infty)$ into three intervals: $[0, d+1/3)$, $[d+1/3, 2(d+1)/3)$, and $[2(d+1)/3, \infty)$, 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 $d! \leq \left(\frac{d+1}{3}\right)^d$, we obtain

$$\forall x \in \left[0, \frac{d+1}{3}\right) \quad \left| \frac{1}{S_d(x)} - e^{-x} \right| \leq \frac{1}{S_d(x)} \leq \frac{d!}{x^d} \leq \left(\frac{d+1}{2x}\right)^d \leq \left(\frac{3}{4}\right)^d = 2^{-\Omega(d)},$$

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

$$\left| \frac{1}{S_d(x)} - e^{-x} \right| = \frac{e^{-x}}{S_d(x)} \left( \frac{x^{d+1}}{(d+1)!} + \frac{x^{d+2}}{(d+2)!} + \ldots \right) \leq \frac{e^{-x}}{S_d(x)} \cdot \frac{x^{d+1}}{(d+1)!} \left( 1 + \frac{x}{d+1} + \frac{x^2}{(d+1)^2} + \ldots \right) \leq \frac{3 e^{-x}}{S_d(x)} \cdot \frac{x^{d+1}}{(d+1)!} \quad (4)$$

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 $(4)$ to obtain

$$\forall x \in \left[ \frac{d+1}{3}, \frac{2(d+1)}{3} \right) \quad \left| \frac{1}{S_d(x)} - e^{-x} \right| \leq 3e^{-\frac{d+1}{3}} \cdot \frac{x}{d+1} \leq 2e^{-d/3} = 2^{-\Omega(d)}.$$
Finally, if $x < \frac{d+1}{e}$, we use that $S_d(x)$ is an exponentially good approximation of $e^x$ in this range. Using $(d+1)! \geq \left(\frac{d+1}{e}\right)^{d+1}$ and $S_d(x) \geq 1$ in (4) to obtain

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

\[\square\]

A more careful argument shows that, in fact, $\frac{1}{S_d(x)}$ approximates $e^{-x}$ up to an error of $2^{-d}$ (Lemma 1 in [18]).

**Lower Bound**

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

**Theorem 4.2.** For every degree-$d$ polynomial $p_d(x)$ with $d$ large enough, $\sup_{x \in [0,\infty)} \left|e^{-x} - \frac{1}{p_d(x)}\right| \geq 50$.

**Proof.** Assume for sake of contradiction that for some large enough $d$ there exists a degree-$d$ polynomial $p_d(x)$ such that $\frac{1}{p_d(x)}$ approximates $e^{-x}$ up to an error of $50^{-d}$ on $[0,\infty)$. Thus, for all $x \in [0,d]$, we have $\frac{1}{p_d(x)} \geq e^{-d} - 50^{-d} \geq 1/2 \cdot e^{-d}$, i.e., $|p_d(x)| \leq 2e^d$. Hence, the degree-$d$ polynomial $1/2 \cdot e^{-d} \cdot p_d(d/2 + d/2 \cdot y)$ is bounded by 1 in absolute value over the interval $[-1,1]$. Using Proposition 2.7, which implies that the Chebyshev polynomials have the fastest growth amongst such polynomials, we obtain $1/2 \cdot e^{-d} \cdot p_d(d/2 + d/2 \cdot y) \leq T_d(y)$. Using the closed-form expression for Chebyshev polynomials given in Proposition 2.8, we have $T_d(y) = \frac{1}{2} \left(\frac{y + \sqrt{y^2 - 1}}{d}\right)^d + \left(\frac{y - \sqrt{y^2 - 1}}{d}\right)^d$. 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 $|e^{-x} - \frac{1}{p_d(x)}| \geq \frac{1}{p_d(x)} - e^{-x} \geq \frac{1}{2} (14e)^{-d} - e^{-4d}$, 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 $50^{-d}$. \[\square\]

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 [13] 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 [56] proved that the best approximation of the form $1/p_d(x)$ achieves an approximation error of $3^{-d+o(d)}$. Newman [40] 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 [19] who finally proved that the smallest approximation error achieved by arbitrary degree-$d$ rational functions is $c^{-d+o(1)}$, where $c$ is the solution to the equation involving elliptic integrals.

**4.2 Approximating $e^{-x}$ on $[0,\infty)$ with Negative Poles**

In this section we study the question of rational approximations to $e^{-x}$ with geometric convergence and with negative zeros. Such rational approximations have been used, in combination with the powerful Laplacian solvers [60, 31, 30] to design near-linear time algorithms to compute approximations of $\exp(-L)v$ when $L$ is a graph Laplacian; see Section 6.5.

Unfortunately, the rational approximation $1/S_d(x)$ that we studied in the last section does not satisfy this requirement of having all negative poles. The zeros of $S_d(x)$ have been well studied (see [65] 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 [61] that if all the (complex) zeros of $S_d$ are scaled by $d$, as $d$ goes to infinity they converge to a point on the curve $|ze^{1/z}| = 1$ on the complex plane.

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 $d$ and it is easy to see that the error in the approximation at $x = 1$ is $\Theta(1/d)$. Saff, Schönhage, and Varga [55] showed that for every rational function of the form $p_d(x)$, 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 in [55] showed that if we instead consider rational functions of the form $p_d(x)(1 + y/d)^{-d}$, then we can approximate $e^{-x}$ up to $O(d^{-2})$ for some degree-$d$ polynomial $p_d(x)$, see also [4]. Formally, [55] proved the following.

**Theorem 4.3.** For every $d$, there exists a degree-$d$ polynomial $p_d$ such that,

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

Since we seek an approximation over an infinite interval, we first apply a variable transformation to convert the interval into a finite one. Towards this, we can write $p_d(x)(1 + y/d)^{-d}$ as a degree-$d$ polynomial in $(1 + y/d)^{-1}$. Hence, in order to make a transformation so that the new variable varies over the interval $[-1, 1]$, we can attempt the transformation $y = 1 - 2(1 + y/d)^{-1}$. Thus, we are looking for a polynomial approximation $q_d(y)$ to the function $e^{-y + \exp(-d \cdot (1+y)/(1-y))}$. Observe that $y$ now varies over the interval $[-1, 1]$, and the approximation error has remained unchanged.

Let $f_d(y) \overset{\text{def}}{=} \exp(-d \cdot (1+y)/(1-y))$, with $f_d(1) = 0$. We could attempt to use the Taylor series approximations in order to approximate this function. One strategy, that can be shown to work, is to consider the polynomial $r_d(y)$ obtained by truncating, up to degree $d$, the Taylor series expansion of the function $f_d \overset{\text{def}}{=} \exp(-1+y)/(1-y)$ around $y = -1$, and to consider the degree-$d^2$ polynomial $r_d^2(y)$ as the approximating polynomial. We do not pursue this approach here. Instead, we now present a simplification of the proof from [55].

**Proof of Theorem 4.3** We start by relaxing the question of a uniform approximation of $f_d$, to an $\ell_2$-approximation problem. However, the relaxation is through an intermediate $\ell_1$ problem. Let $f_d(t)$ denote the $k$th derivative of $f_d$, i.e., $f_d^{(k)}(t) \overset{\text{def}}{=} \frac{d^k}{dt^k} f_d(t)$. Then, the following is a simple sequence of equalities and inequalities which rely on Cauchy-Schwartz.

$$\inf_{q_d} \sup_{y \in [-1, 1]} |f_d(y) - q_d(y)| = \inf_{r_{d-1}} \sup_{y \in [-1, 1]} \left| \int_y^1 (f_d^{(1)}(t) - r_{d-1}(t)) \, dt \right|$$

$$\leq \inf_{r_{d-1}} \int_{-1}^1 \left| f_d^{(1)}(t) - r_{d-1}(t) \right| \, dt$$

$$\leq \sqrt{2} \inf_{r_{d-1}} \sqrt{\int_{-1}^1 \left( f_d^{(1)}(t) - r_{d-1}(t) \right)^2 \, dt}. \quad (5)$$

The first equality holds if we take the infimum over 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 constant weight function, which are given by Legendre polynomials.
\[ L_k(t) \overset{\text{def}}{=} \frac{1}{2^k \cdot k!} [(t^2 - 1)^k], \] and satisfy \( \int_{-1}^{1} L_i(t) L_j(t) \, dt = \frac{2}{2^i+1} \) if and only if \( i = j \) and 0 otherwise (see [62]).

Hence, we can write the last expression explicitly to obtain
\[
\inf_{q_d} \sup_{y \in (-1,1]} |f_d(y) - q_d(y)| \leq \sqrt{\sum_{k \geq d} (2k+1) \gamma_k^2},
\]

where \( \gamma_k \) denotes the inner product with the \( k \)th Legendre polynomial \( \gamma_k = \int_{-1}^{1} f_d^{(1)}(t)L_k(t) \, dt \). Plugging in the definition of Legendre polynomials, and using the integration by parts successively, we obtain
\[
\int_{-1}^{1} f_d^{(1)}(t) \frac{d^k}{dt^k} [(t^2 - 1)^k] \, dt = \frac{(-1)^k}{2^k \cdot k!} \int_{-1}^{1} (t^2 - 1)^k f_d^{(k+1)}(t) \, dt.
\]

If we let \( v \overset{\text{def}}{=} \frac{2d}{1-t} \), we obtain, \( f_d(t) = e^{d-v} \) and \( f_d^{(1)}(t) = \frac{1-v}{(1-t)} e^{d-v} \). A simple induction argument generalizes this to give
\[
(1-t)^k \frac{d^{k+1}}{dt^{k+1}} f_d(t) = (1-t)^{k+1} f_d^{(k+1)}(t) = -e^{d} \frac{d^k}{dv} \left[ v^{k+1} e^{-v} \right].
\]

We now invoke the generalized Laguerre polynomials of degree \( k \) orthogonal with respect to the weight function \( ve^{-v} \), denoted by \( G_k(v) \) and defined to be \( \frac{1}{k!} \frac{d^k}{dv} [v^{k+1} e^{-v}] \) (see [62]). Hence, simplifying (7), we obtain
\[
\gamma_k = -e^{d} \frac{1}{2^k} \int_{-1}^{1} (t+1)^k \frac{ve^{-v}}{(1-t)} G_k(v) \, dt = -e^{d} \int_{d}^{\infty} \left( 1 - \frac{d}{v} \right)^k e^{-v} G_k(v) \, dv.
\]

Squaring the above equality, and applying Cauchy-Schwartz, we obtain
\[
\gamma_k^2 \leq e^{2d} \int_{d}^{\infty} ve^{-v} (G_k(v))^2 \, dv \cdot \int_{d}^{\infty} \frac{1}{v} \left( 1 - \frac{d}{v} \right)^{2k} e^{-v} \, dv.
\]

Now, we use \( \int_{0}^{\infty} ve^{-v}(G_k(v))^2 \, dv = k + 1 \) (see [62]), and substitute \( v = d(1 + z) \) to obtain
\[
\gamma_k^2 \leq e^{d} (k + 1) \int_{0}^{\infty} \frac{z^{2k}}{(z+1)^{2k+1}} e^{-dz} \, dz.
\]

Plugging this back in (6), we obtain
\[
\left( \inf_{q_d} \sup_{y \in (-1,1]} |f_d(y) - q_d(y)| \right)^2 \leq e^{d} \int_{0}^{\infty} \sum_{k \geq d} (k + 1)(2k+1) \frac{z^{2k}}{(z+1)^{2k+1}} e^{-dz} \, dz.
\]

We can sum up the series in the above equation for any \( z \geq 0 \) to obtain \( \sum_{k \geq d} (k + 1)(2k+1) \frac{z^{2k}}{(z+1)^{2k+1}} \lesssim \left( \frac{z}{z+1} \right)^{2d-2} (d^2 + dz + z^2) \). Here \( \lesssim \) means that the inequality holds up to an absolute constant. This implies that
\[
\left( \inf_{q_d} \sup_{y \in (-1,1]} |f_d(y) - q_d(y)| \right)^2 \lesssim \int_{0}^{\infty} \left( \frac{z}{z+1} \right)^{2d-2} (d^2 + dz + z^2) e^{-dz} \, dz.
\]

It is a simple exercise to show that for all \( z \geq 0 \), the expression \( e^{d-\frac{dz+z^2}{z+1}} \left( \frac{z}{z+1} \right)^{2d-2} \) is maximized for \( z = 1 \) and, hence, this expression is bounded by \( 4e \cdot 4^{-d} \). Thus,
\[
\left( \inf_{q_d} \sup_{y \in (-1,1]} |f_d(y) - q_d(y)| \right)^2 \lesssim 4^{-d} \int_{0}^{\infty} (d^2 + dz + z^2) e^{-z} \, dz \lesssim d^2 \cdot 4^{-d},
\]

which concludes the proof. \( \square \)

In Section 6.5, we also bound the magnitudes of the coefficients in this polynomial and show how to compute arbitrarily good approximations to them efficiently.

16
5 Approximating $x^{-1}$ Using Exponentials

In this section we give an approximation to $x^{-1}$ using a small number of exponentials. As we see in Section 6.3, this immediately implies a reduction from approximate matrix inversion (equivalent to approximately solving a linear system) to approximating the matrix exponential, thus proving that the problems are essentially equivalent.

**Theorem 5.1.** Given $\varepsilon, \delta \in (0, 1]$, there exist $\text{poly}(\log 1/\varepsilon\delta)$ numbers $0 < w_j$ and $t_j = O(\text{poly}(1/\varepsilon\delta))$, 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 literature [9, 10]. The proof we present is from [54]. The starting point of the proof is to discretize this integral to a sparse sum of exponentials. One approach to discretize an integral to a sum is via the trapezoidal rule - by approximating the area under the integral using trapezoids of small width, say $h$,

$$
\int_a^b g(t) \, dt \approx T^h_{b-a} \defeq \frac{h}{2} \sum_{j=0}^{K-1} \left( g(a + jh) + g(a + (j + 1)h) \right),
$$

where $K \defeq (b-a)/h$ is an integer. Applying this rule to the above integral after truncating it to a large enough interval $[0, b]$, we obtain the approximation $x^{-1} \approx \frac{1}{2} \sum_{j=0}^{b/h-1} \left( e^{-x(j+1)h} + e^{-xj}h \right)$. The choice of $h$ determines the discretization of the integral and, hence, the sparsity of the approximating sum $K$. Recall that the error must be of the form

$$
\forall x \in [\varepsilon, 1], \quad \left| x^{-1} - \frac{1}{2} \sum_{j=0}^{K} \left( e^{-xj} + e^{-x}e^{j}h \right) \right| \leq \delta x^{-1}.
$$

For $x = 1$, we obtain $h \leq O_\delta(1)$ and, hence, $K \geq \Omega_\delta(b)$. Moreover, the error in truncating the integral is $\int_b^\infty e^{-ax} \, dx = x^{-1}e^{-bx}$, forcing $b \geq 1/\varepsilon \cdot \log 1/\delta$ to be at most $\delta/\varepsilon$ for all $x \in [\varepsilon, 1]$. Thus, this approach to discretization can only give us a sum which uses $\text{poly}(1/\varepsilon)$ exponentials.

This suggests that we should select a discretization where $t$ increases much more rapidly with $h$, e.g., exponentially instead of linearly. This can be achieved by substituting $t = e^y$ in the above integral to obtain the identity $x^{-1} = \int_{-\infty}^\infty e^{-x e^y} \, dy$. Let $f_i(y) \defeq e^{-x e^y}$. Observe that $f_i(y) = x^{-1} \cdot f_{i+1}(y + \ln x)$. 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 $1/\varepsilon \cdot \log 1/\varepsilon$ 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 above result.

5.1 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 $\sum_{i=0}^{k} i^j$, have deep connections to several areas of mathematics\footnote{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.}. They can be defined recursively: $b_0 = 1$, and for all $k \geq 1$, $\sum_{j=0}^{k} \binom{k}{j} b_j = 0$. Given the Bernoulli numbers, the Bernoulli polynomials are defined to be $B_k(y) \defeq \sum_{j=0}^{k} \binom{k}{j} b_j y^{k-j}$. Using properties of the Bernoulli polynomials, and a well-known connection to the Riemann zeta function, we obtain the following bounds (see [20]).

**Lemma 5.2.** For any non-negative integer $k$, and for all $y \in [0, 1]$, $\frac{|B_{2k}(y)|}{(2k)!} \leq \frac{|B_{2k}|}{(2k)!} \leq \frac{4}{(2k)^{2k}}$. 

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 $g^{(k)}(y) \overset{\text{def}}{=} \frac{d^k}{dy^k} g(y)$.

**Lemma 5.3.** Given a function $g : \mathbb{R} \to \mathbb{R}$, for any $a < b$, any $h > 0$, and $N \in \mathbb{N}$, we have,

$$
\int_a^b g(y) \, dy - T_a^b g = h^{2N+1} \sum_{j=0}^N \frac{B_{2j}(y)}{(2j)!} g^{(2j)}(a + yh) \, dy - \sum_{j=1}^N \frac{b_{2j}}{(2j)!} h^{2j} \left( g^{(2j-1)}(b) - g^{(2j-1)}(a) \right),
$$

where $K \overset{\text{def}}{=} \frac{b-a}{h}$ is an integer, and $\lfloor \cdot \rfloor$ denotes the integer 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 $\int_a^b g(y) \, dy$ 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 $\delta$, we need to choose $N$ to be roughly $\Theta(\log 1/\delta)$ so that the first error term in (8) is comparable to $\delta$.

**The Proof**

**Proof of Theorem 5.7.** We fix the step size $h$, approximate the integral $\int_{-bh}^{bh} f_x(y) \, dy$ using the trapezoidal rule ($b$ is a positive integer), and bound the approximation error using the Euler-Maclaurin formula. We let $b$ go to $\infty$, 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 $\int_{-bh}^{bh} f_x(y) \, dy$, and using Lemma 5.2, we obtain,

$$
\left| \int_{-bh}^{bh} f_x(y) \, dy - T_{-bh}^{bh} f_x \right| \leq 4 \left( \frac{h}{2\pi} \right)^{2N} \int_{-bh}^{bh} |f_x^{(2N)}(y)| \, dy + \sum_{j=1}^N 4 \left( \frac{h}{2\pi} \right)^{2j} \left( \left| f_x^{(2j-1)}(-bh) \right| + \left| f_x^{(2j-1)}(bh) \right| \right).
$$

(9)

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 $s$ goes to $\pm \infty$. We let $b$ go to $\infty$ and observe that the discretized sum converges to $h \sum_{j \in \mathbb{Z}} f_x(jh)$, hence, (9) implies that

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

(10)

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 $(k+1)^{k+1}$ (Fact 1.3 in [54]). This, in turn, implies the bound $\int_{-\infty}^{\infty} |f_x^{(2N)}(y)| \, dy \leq x^{-1} \cdot \Theta(N)^{4N}$ (Lemma 1.4 in [54]). Thus, we can choose $h = \Theta(1/\gamma^2)$ and $N = \Theta(\log 1/\delta)$ to obtain the following approximation for all $x > 0$:

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

(11)
The final step is to truncate the above discretization. Since the function $f_h(y)$ is non-decreasing for $y < \log 1/x$, we can majorize the lower tail with an integral to obtain $h \sum_{j \leq A} f_h(j) \leq \int_{-\infty}^{h} f_h(t) \, dt = x^{-1} \left(1 - e^{-x e^h}\right)$. Thus, for $A = \left[ -1/h \cdot \log 1/\delta \right]$, we obtain that the lower tail is $O(\delta \cdot d^{-1})$. Similarly, for the upper tail, using that $f_h(y)$ is non-increasing for $y \geq \log 1/x$, for $B = \left[ 1/h \cdot \log (1/\delta) \right]$, we obtain that the upper tail $h \sum_{j \geq B} f_h(j) \leq O(\delta \cdot d^{-1})$. Combining these tail bounds with (11), we obtain

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

which completes the proof.

\[\square\]

6 Applications

In this section, we present several algorithmic applications of the approximation theory results obtained in the previous sections. All these results are obtained by lifting the approximation results for scalar functions such as $x^e$, $e^{-x}$ or $x^{-1}$ to the matrix world. Since matrices capture graphs, we often obtain fast algorithms for important graph problems. We start with some basics on matrices and graphs.

6.1 Matrices and Graphs

We are primarily concerned with $n \times n$ symmetric matrices over the reals. A fundamental theorem in linear algebra (see [63, 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 $\Lambda$ is the diagonal matrix corresponding to its eigenvectors. $A$ 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. The spectral norm of a matrix $A$ is its $2 \rightarrow 2$ norm, which is defined to be $\sup_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2}$. Thus, all eigenvalues of $A$ are bounded in absolute value by the spectral norm of $A$. For a PSD matrix, its norm is its largest eigenvalue. Henceforth, $\|\cdot\|$ is used to denote the $\ell_2$ norm for vectors and the spectral norm for matrices.

For a function $f : \mathbb{R} \mapsto \mathbb{R}$ and a real symmetric matrix $A$, one can define $f(A)$ as follows. First, for a diagonal matrix $\Lambda$ let $f(\Lambda)$ denote the diagonal matrix where the $(i,i)$th entry is $f(\Lambda_{ii})$. Then, $f(A)$ is defined to be $U f(\Lambda) U^\top$ where $U \Lambda U^\top$ is the spectral decomposition of $A$. Thus, a polynomial $p(x) = \sum_{i=0}^d c_i x^i$, when applied to $A$, is a matrix $p(A)$ which can be seen to be $\sum_{i=0}^d c_i A^i$ since $U^{-1} U = I$. Moreover, $\exp(A)$ or $e^A$ is $\sum_{k=0}^\infty \frac{A^k}{k!}$.

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 $x = A^{-1} v$.

Finally, we are in interested in undirected graphs $G = (V, E)$ with $n \overset{\text{def}}{=} |V|$ vertices and $m \overset{\text{def}}{=} |E|$ 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 $m$ non-zero entries. Let $e_i$ denote the vector with 1 in the $i$th coordinate and 0 elsewhere. The matrix $L \overset{\text{def}}{=} \sum_{i \neq j} A_{i,j} (e_i - e_j)(e_i - e_j)^\top$ is called the combinatorial Laplacian of $G$. If $D$ is the diagonal matrix with $D_{i,i} \overset{\text{def}}{=} \sum_{j \neq i} A_{i,j}$, then $L = D - A$. The Laplacian of a graph $L$ is always PSD; $L \succeq 0$. 

19
6.2 Simulating Random Walks and Finding Sparse Cuts

Consider a graph $G = (V, E)$ with $|V| = n, |E| = m$, and let $A$ denote its adjacency matrix. 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 $A_{i,j}$, 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 $\tilde{W}^s v$ where $\tilde{W} \overset{\text{def}}{=} D^{-1/2}A D^{1/2}$. Computing such distributions, sometimes starting from arbitrary real vectors rather than probability vectors, is a fundamental problem that finds many applications, for instance in finding sparse cuts in graphs as explained below; often, a good enough approximation to such a distribution suffices.

6.2.1 Quadratically Faster Random Walks

A simple way to compute $\tilde{W}^s v$ is to multiply the matrix $\tilde{W}$ 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 $\tilde{W}^s v$ for all distributions, sometimes starting from arbitrary real vectors rather than probability vectors, is a fundamental problem that finds many applications, for instance in finding sparse cuts in graphs as explained below; often, a good enough approximation to such a distribution suffices.

Theorem 6.1 (Corollary to Theorem 3.2). For a symmetric $M$ with $\|M\| \leq 1$, a positive integer $s$ and any $\delta > 0$, define $d \overset{\text{def}}{=} \lceil \sqrt{2}\log \frac{2}{\delta} \rceil$. Then, the degree-$d$ polynomial $p_{s,d}(M)$, defined by (3) satisfies $\|M^s - p_{s,d}(M)\| \leq \delta$.

Proof. Let $\{\lambda_i\}_i$ be the eigenvalues of $M$ with $\{u_i\}_i$ as the 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 3.2 implies that for all $i$, $|\lambda_i^s - p_{s,d}(\lambda_i)| \leq \delta$. Note that if $\lambda_i$ is an eigenvalue of $M$, then $\lambda_i^s$ is the corresponding eigenvalue of $M^s$ and $p_{s,d}(\lambda_i)$ is that of $p_{s,d}(M)$ with the same eigenvector. Hence, we have $\|M^s - p_{s,d}(M)\| = \left\| \sum_i (\lambda_i^s - p_{s,d}(\lambda_i)) u_i u_i^\top \right\| = \max_i |\lambda_i^s - p_{s,d}(\lambda_i)| \leq \delta$. \hfill \Box

When we try to apply this theorem to $\tilde{W}$ we face the obvious problem that $\tilde{W}$ is not necessarily symmetric. This can be handled by considering the matrix $W \overset{\text{def}}{=} D^{-1/2}A D^{1/2}$, which is symmetric. Thus, $\tilde{W}^s v = D^{1/2}W^s D^{-1/2} v$. For now, we consider the case that $G$ is $d$-regular, i.e., $D = d \cdot I$ for some $d$. In this case $\tilde{W} = W$. Further, it can be seen 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 $d = \lceil \sqrt{2}\log \frac{2}{\delta} \rceil$. Thus, appealing to the theorem above, we obtain an efficient $\delta$ approximation to $W^s v$, i.e., $\|W^s v - p_{s,d}(W)v\| \leq \delta \|v\| \leq \delta$. 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) and the recursive definition of Chebyshev polynomials from (2) 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) implies that the non-zero coefficients are binomial coefficients up to a scaling factor. For instance, assuming that $s$ is even, the coefficient of $T_i(\cdot)$ is $2^{s-i} \binom{s/2}{i/2}$. \textit{Prima facie}, computing 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, 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

---

9The convention in the Markov Chains literature is to express the probability distribution as a row vector $v^\top$ instead, giving the probability after $s$ steps as $v^\top W^s$. We will use the column vector convention. The only resulting change is that the walk matrix is replaced by its transpose everywhere.
Theorem 6.2. Let $W$ be the random walk matrix for a regular graph $G$ with $n$ vertices and $m$ edges. Then, for any positive integer $s$, starting distribution $v$, and $\delta \in (0, 1/2]$, there is an algorithm that computes a vector $w$ such that $\|W^sv-w\| \leq \delta$ in $O \left( (m+n) \sqrt{s \log 1/\delta} \right)$ arithmetic operations.

Theorem 6.2 can be easily generalized to a reversible irreducible Markov chain with transition matrix $P$ and stationary distribution $\pi$. Let $\Pi$ be the diagonal matrix defined by $\Pi(i,i) = \pi(i)$, the matrix $\Pi^{1/2}P\Pi^{-1/2}$ is symmetric and has norm at most 1 and, hence, we can apply the above algorithm with $W = \Pi^{1/2}P\Pi^{-1/2}$ and the vector $\Pi^{1/2}v$, and obtain a vector $u$ with the approximation guarantee $\|(P^T)^sv-u\| \leq \delta \sqrt{\max_{i,j} \pi(i) \min_{i,j} \pi(j)} \|v\|$ in $O \left( (tp+n) \sqrt{s \log 1/\delta} \right)$ arithmetic operations, where $tp$ is the cost of multiplying the matrix $P^T$ with a given vector.

6.2.2 Finding Sparse Cuts

We now outline how we can use the algorithm in the proof of Theorem 6.2 to speed up an algorithm to find sparse cuts in a graph. For a graph $G = (V,E)$ with adjacency matrix $A$, a cut $S \subseteq V$ is said to have sparsity or conductance

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

The conductance of a graph, $\phi \overset{\text{def}}{=} \min_{S \subseteq V} \phi(S)$, gives a measure of how interconnected a graph and is an important problem theory and practice, see [63] Chapter 5] for a detailed discussion on this problem. It is also NP-hard to find the cut of least conductance and, hence, one has to be satisfied with algorithms that compute cuts whose sparsity is close to that of the sparsest cut. A celebrated result of Cheeger [15] and Alon and Milman [3] relates the second smallest eigenvalue of the Laplacian $L$ 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 $\phi \leq O \left( \sqrt{\lambda_2/d} \right)$. Let $\lambda \overset{\text{def}}{=} \lambda_2/d$ be the normalized spectral gap and $\ell \overset{\text{def}}{=} 1/\ell$ be the normalized Laplacian. In fact, a cut of conductance $O \left( \sqrt{\lambda} \right)$ can be recovered from the second eigenvector of $L$ and, thus, algorithmically, it is sufficient to compute the second eigenvector of $L$. Mihail [37] proved a stronger version of this theorem which showed how to produce a cut of sparsity at most $O(\sqrt{\lambda})$ from any vector $v$ (orthogonal to the all ones vector) such that $v^T \nabla v = \lambda'$. Note that for $d$-regular graphs, the all ones vector is an eigenvector of $L$ with eigenvalue $0$. Hence, the second eigenvector is orthogonal to this vector.

We show how, as a direct consequence to Theorem 6.2, we can produce a vector $u$ such that $u^T \nabla u / u^T u \leq O(\lambda)$ giving us an algorithm to find a cut of sparsity at most $O(\sqrt{\lambda})$ in time roughly $O(m/\sqrt{\lambda})$. This gives a

---

10 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 few bits to store each of these numbers are sufficient. 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 $O(d^2 2^{-b})$. Hence, the absolute error in $\sigma$ 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.
quadatically better dependence in \( \lambda \) than the standard algorithm using the Power method. Formally, we prove the following theorem.

\textbf{Theorem 6.3.} Given an undirected regular graph \( G \) with normalized spectral gap \( \lambda \), we can find a cut of conductance \( O(\sqrt{\lambda}) \) with probability at least \( 1/3 \) using \( O(m/\sqrt{\lambda} \cdot \log n/\lambda) \) operations.

\textit{Proof.} We use the algorithm in the proof of Theorem \( 6.2 \) to approximate \( W^t v \) where \( W = \frac{1}{n} A \) is the random walk matrix, \( v \) a random unit vector orthogonal to the all ones vector, with parameters \( s \) and \( \delta \) (the required \( s \) and \( \delta \) is determined later). Note that \( \lambda \) is the second smallest eigenvalue of \( I - W \) and, hence, \( 1 - \lambda \) is the second largest eigenvalue of \( W \). Let \( v_2 \) be the corresponding unit eigenvector.

Let \( u \) be the approximating vector as obtained from Theorem \( 6.2 \). Let \( u^* \defeq W^t v \) and let \( \Delta \defeq u - u^* \). Thus, we know that \( \| \Delta \| \leq \delta \). Since \( v \) was chosen as a random unit vector orthogonal to the uniform distribution, with probability at least \( 2/3 \) we have, \( v_2^t v \geq \frac{1}{3\sqrt{n}} \). This implies, \[
    u^* u^* \geq \frac{1}{2} u^* L u^* \geq \lambda (1 - \lambda) \frac{1}{18n}. \]

We choose \( \delta \defeq \sqrt{(1 - \lambda)^2 n/2} \), implying \( u^* L u^* \geq \frac{1}{4} \delta^2 \). Thus, we have, \[
    \frac{u^* L u}{u^* u} = \frac{(u^* + \Delta)^T L (u^* + \Delta)}{(u^* + \Delta)^T (u^* + \Delta)} \leq \frac{2(u^* L u^* + \Delta^T L \Delta)}{(||u^*|| - ||\Delta||)^2} \leq \frac{2(u^* L u^* + \Delta^T L \Delta)}{(||u^*|| - \delta)^2} \leq \frac{12u^* L u^*}{u^* u}, \]

where the first inequality uses \( (u^* + \Delta)^T L (u^* + \Delta) \leq 2(u^* L u^* + \Delta^T L \Delta) \), which is the same as \( 0 \leq (u^* - \Delta)^T L (u^* - \Delta) \) after rearranging. As dictated by the Power method to approximate the spectral gap, we choose \( s = \left\lceil \frac{\log(29/\delta)}{2 \log(1/\sqrt{1 - \lambda})} \right\rceil \) to ensure \( \frac{u^* L u}{u^* u} \geq 1 - O(\lambda) \) (see [63, Chapter 8] for a proof). This implies, \[
    \frac{u^* L u^*}{u^* u^*} = 1 - \frac{u^* A u^*}{u^* u^*} = O(\lambda). \]

Thus, \( \frac{u^* L u}{u^* u} \leq O(\sqrt{\lambda}) \). We note that since \( v \) is orthogonal to the all ones vector, the vector \( u \) returned is also orthogonal to the all ones vector. Hence, by Mihail’s theorem, we can round \( u \) to find a cut of conductance \( O(\sqrt{\lambda}) \).

The running time for this procedure is dominated by the time required to compute \( u \), which requires \( O\left( (t_A + n) \sqrt{s \log 1/\delta} \right) \) operations. We note that \( \delta = \Omega(\sqrt{\lambda/n}) \), which implies that the total number of operations required is \( O\left( m/\sqrt{\lambda} \cdot \log n/\lambda \right) \).

\[\square\]

6.3 \textbf{Solving Linear Equations}

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 \( x^\perp \defeq A^{-1} v \) can be computed by Gaussian elimination, but the fastest known implementation requires \( O(n^{2.373}) \) time. 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 \( \tilde{O}(m) \)[11], 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 positive definite (PD). The solutions these methods produce are, in

---

[11] The \( \tilde{O} \) notation hides polynomial factors in \( \log n \).
general, approximate which 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 of the $\sqrt{s}$ degree polynomials approximating $x'$.

### 6.3.1 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: Let the squared $A$-norm of the error $x - x^*$ be $f_A(x) \overset{\text{def}}{=} \|x - x^*\|_A^2 = \langle x - x^* \rangle^T A \langle x - x^* \rangle = x^T A x - 2x^T v + x^T A x^*$, and 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 $x = x^*$.

When minimizing $f_A$, each iteration of the gradient descent method is as follows: Start from the current estimate of $x^*$, 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. Thus, $x_{t+1} = x_t - \alpha_t \nabla f_A(x_t) = x_t - \alpha_t (A x_t - v)$. If we define the residual $r_t = v - A x_t$, we can easily compute the $\alpha_t$ that minimizes $f_A$ as $\frac{r_t^T r_t}{r_t^T A r_t}$. Substituting this value of $\alpha_t$, and using $x^* - x_t = A^{-1} r_t$, we obtain

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

For any $z$, we have $z^T A z \leq \lambda_1 z^T z$ and $z^T A^{-1} z \leq \lambda_n^{-1} z^T z$, where $\lambda_1$ and $\lambda_n$ are the smallest and the largest eigenvalues of $A$ respectively. Thus, $\|x_{t+1} - x^*\|_A^2 \leq (1 - \kappa^{-1}) \|x_t - x^*\|_A^2$, where $\kappa \overset{\text{def}}{=} \lambda_1/\lambda_n$ is the condition number of $A$. Hence, assuming we start with $x_0 = 0$, we can find an $x_t$ such that $\|x_t - x^*\|_A \leq \delta \|x^*\|_A$ 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 $\delta$ approximate solution to $x^*$ in time $O((t_A + n) \kappa \log 1/\delta)$.

### 6.3.2 The Conjugate Gradient Method

Suppose we run the gradient descent-based method described in the previous section for $k$ iterations. Observe that at any step $t$, we have $x_{t+1} \in \text{Span}\{x_t, Ax_t, v\}$. Hence, for $x_0 = 0$, it follows by induction that $x_k \in \text{Span}\{v, Av, \ldots, A^k v\}$. 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 3.1, 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 $A$.

Let us consider the first question. Let $\{v_0, \ldots, v_k\}$ be a basis for $K = \text{Span}\{v, Av, \ldots, A^k v\}$ (called the Krylov subspace of order $k$). Hence, any vector in the subspace can be written as $\sum_{i=0}^k \alpha_i v_i$. Our objective then becomes $\|x^* - \sum \alpha_i v_i\|_A^2 = \langle \sum \alpha_i v_i \rangle^T A \langle \sum \alpha_i v_i \rangle - 2 \langle \sum \alpha_i v_i \rangle^T v + \|x^*\|_A^2$. Solving this optimization problem for $\alpha_i$ requires matrix inversion, the very problem we set out to mitigate. The crucial observation is that if the $v_i$s are $A$-orthogonal, i.e., $v_i^T A v_j = 0$ for $i \neq j$, then all the cross-terms disappear.
Then, \[\|x^* - \sum_i \alpha_i v_i\|_A^2 = \sum_i (\alpha_i^2 v_i^\top A v_i - 2\alpha_i v_i^\top v) + \|x^*\|_A^2,\]
and we can explicitly obtain the best solution since \[\alpha_i = \frac{v_i^\top v}{v_i^\top A v_i}\] as in the gradient descent-based method.

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 \(v_0 \equiv v\). At the \(i\)th iteration, we compute \(A v_{i-1}\) 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 \(\text{Span}\{v_0, \ldots, v_i\} = \text{Span}\{v, A v, \ldots, A^i v\}\). Now, since \(A v_j \in \text{Span}\{v_0, \ldots, v_{j+1}\}\) for every \(j\), and \(A\) is symmetric, \(A\)-orthonormality of the vectors implies \(v_i^\top A (A v_j) = v_j^\top A (A v_i) = 0\) for all \(j\) such that \(j + 1 < i\). This implies that we need to \(A\)-orthogonalize \(A v_i\) 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. We now analyze the approximation guarantee achieved by this vector. Note that the Krylov subspace \(\mathcal{K} = \text{Span}\{v, A v, \ldots, A^k v\}\) 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 \(\beta_i\). Let \(\Sigma_k\) 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

\[
\|x_k - x^*\|_A^2 = \inf_{x \in \mathcal{K}} \|x - x^*\|_A^2 = \inf_{p \in \Sigma_k} \|x^* - p(A)v\|_A^2 = \|x^*\|_A^2 \inf_{p \in \Sigma_k} \|I - p(A)\|_A^2.
\]

Observe that the last expression can be written as \(\|x^*\|_A^2 \inf_{q \in \Sigma_{k+1}, q(0) = 1} \|q(A)\|_A^2\), 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 \(\|q(A)\|_A^2 = \max_{\lambda \in \sigma(A)} |q(\lambda)|^2 \leq \sup_{\lambda \in [\lambda_n, \lambda_1]} |q(\lambda)|^2\), 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 \(\|x_k - x^*\|_A \leq \delta\) \(\|x^*\|_A\) is achieved after \(k\) rounds, it suffices to show that there exists a polynomial of degree \(k + 1\) that takes value 0 at 1, and whose magnitude is less than \(\delta\) on the interval \([\lambda_n, \lambda_1]\).

As a first attempt, we consider the degree-\(s\) polynomial \(q_0(x) \equiv (1 - 2x/(\lambda_1 + \lambda_n))^s\). The maximum value attained by \(q_0\) over the interval \([\lambda_n, \lambda_1]\) is \((s-1)/(x+1))^s\). Hence, \(d_0 \equiv s \log 1/\delta\) suffices for this value to be less than \(\delta\). Or equivalently, approximately \(s \log 1/\delta\) rounds suffice for error guarantee \(\|x - x^*\|_A \leq \delta\) \(\|x^*\|_A\), 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 5.1. Let \(\varepsilon \equiv 1 - 2x/(\lambda_1 + \lambda_n)\). Hence, \(q_0(x) = z^s\). As \(x\) ranges over \([0, \lambda_n + \lambda_1]\), the variable \(z\) varies over \([-1, 1]\). Theorem 3.2 implies that for \(d \equiv \sqrt{2d_0 \log 2/\delta}\), the polynomial \(p_{d_0, d}(z)\) approximates the polynomial \(z^d_0\) up to an error of \(\delta\) over \([-1, 1]\). Hence, the polynomial \(q_1(x) \equiv p_{d_0, d}(z)\) approximates \(q_0(x)\) up to \(\delta\) 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 \(2\delta\) on the interval \([\lambda_n, \lambda_1]\), and at least \(1 - \delta\) at 0. Thus, the polynomial \(d_1 / q_1(0)\) is a polynomial of degree \(d = O(\sqrt[3]{\kappa} \log 1/\delta)\) that takes value 1 at 0, and at most \(2\delta / (1 - \delta) = O(\delta)\) on the interval \([\lambda_n, \lambda_1]\). Or equivalently, \(O(\sqrt[3]{\kappa} \log 1/\delta)\) rounds suffice for an error guarantee \(\|x - x^*\|_A \leq O(\delta)\) \(\|x^*\|_A\), which gives a quadratic improvement over the guarantee provided by the gradient descent-based method.

We summarize the guarantees of the Conjugate Gradient method in the following theorem:

**Theorem 6.4.** Given an \(n \times n\) symmetric matrix \(A > 0\), and a vector \(v \in \mathbb{R}^n\), the Conjugate Gradient method can find a vector \(x\) such that \(\|x - A^{-1} b\|_A \leq \tilde{O}(\delta)\|A^{-1} b\|_A\) in time \(O((t_A + n) \cdot \sqrt[3]{\kappa(A)} \log 1/\delta)\), where \(t_A\) is the time required to multiply \(A\) with a given vector; and \(\kappa(A)\) is the condition number of \(A\).
We note that this proof of the guarantee of the Conjugate Gradient method is different from the traditional proof, which directly proves that the polynomial $T_d(1 - 2/(\lambda_i + \lambda_j))$ for $d = O(\sqrt{\log 1/\delta})$ is such that it takes value 0 at 1, and is smaller than $\delta$ in magnitude on the interval $[\lambda_n, \lambda_1]$ (see, e.g., [63]).

### 6.4 Computing Eigenvalues via the Lanczos Method

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 [48] for an extensive discussion. In this section, we present the Lanczos method for approximating the largest eigenvalue of a symmetric matrix and show how existence of good polynomial approximations to $A$ implies fast algorithms for computing good approximations to $f(A)v$ quickly. For simplicity, in this section we assume that the matrix is PSD.

We start by recalling the variational characterization of eigenvalues: The largest eigenvalue of $A$ is equal to the maximum value of the Rayleigh quotient $\frac{w^\top Aw}{w^\top w}$ over all non-zero vectors $w$. The power method (see [63], Chapter 8) tells us that for a unit vector $v$ picked uniformly at random, with constant probability, the vector $Av$ achieves a Rayleigh quotient of at least $(1 - s)\lambda_1$ for $s$ roughly $1/\delta$, where $\lambda_1$ is the largest eigenvalue of $A$. The Lanczos method essentially finds the vector in the Krylov subspace $K \define \{v, Av, \ldots, A^k v\}$ that maximizes the Rayleigh quotient. We prove below, again using the polynomial approximations to $x^i$ from Section 6.4 that in order to find a vector with Rayleigh quotient at least $(1 - s)\lambda_1$, it suffices to choose $k$ to be approximately $1/\sqrt{\delta}$. Such a result was proven in [32]. We present a simpler proof here with a slightly worse bound.

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. Pick $v$ to be a unit vector chosen uniformly at random. Assume that $v$ can be expressed in the eigenbasis for $A$; i.e., $v = \sum_{i=0}^n \alpha_i u_i$. Let $\{v_0, \ldots, v_k\}$ be any orthonormal basis for the Krylov subspace $K$.

Let $T$ 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 on to $K$. Let $T \define V^\top AV$. The $(k+1) \times (k+1)$ matrix $T$ denotes the operator $A$ restricted to $K$, expressed in the basis $\{v_i\}_{i=0}^k$. Now, since $v, Av \in K$, we have $Av = (VV^\top)A(VV^\top)v = VTV^\top v$. Iterating this argument, we obtain that for all $i \leq k$, we have $A^i v = VTV^\top v$ and, hence, by linearity, $p(A)v = Vp(T)V^\top v$ for any $p \in \Sigma_k$. Also, note that for every $w \in K$, we have $w = VV^\top w$, and hence,

$$w^\top Aw = (w^\top V V^\top) A (V V^\top w) = w^\top V (V^\top AV) V^\top w = (w^\top V) T (V^\top w).$$

In words, the above equality says that for any vector $w \in K$, 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 $T$.

The Lanczos method computes the largest eigenvalue of $T$, $\lambda_1(T)$ and outputs it as an approximation to $\lambda_1(A)$. By the variational characterization of the largest eigenvalues, it follows that $\lambda_1(T) \leq \lambda_1(A)$. We have

$$\lambda_1(T) = \max_{w \in \mathbb{R}^{k+1}} \frac{w^\top Tw}{w^\top w} = \max_{z \in K} \frac{z^\top TV^\top z}{z^\top z} = \max_{z \in K} \frac{z^\top Az}{z^\top z} = \max_{p \in \Sigma_k} \frac{v^\top p(A)p(A)^2v}{v^\top (A^2)^2v} = \max_{p \in \Sigma_k} \sum_{i} \lambda_i p(\lambda_i)^2 \alpha_i^2.$$\[\text{12}\]

Later on we show how to construct such a basis quickly, similar to the case of the Conjugate Gradient method.
where the second equality holds since $\mathcal{K}$ is a $k + 1$ dimensional subspace with the columns of $V$ as an orthonormal basis, and the fourth equality holds because every $z \in \mathcal{K}$ can be expressed as $p(A)V$ for some $p \in \Sigma_k$.

Since $v$ is picked uniformly at random, with probability at least $1/2$, we have $\alpha_i^2 \geq 1/4n$. Thus, assuming that $\alpha_i^2 \geq 1/4n$, for any $p \in \Sigma_k$, we can bound the relative error:

$$\frac{\lambda_1(A) - \lambda_1(T)}{\lambda_1(A)} \leq \frac{\sum_{i=0}^{n}(1 - \lambda_i/\lambda_1)p(\lambda_1)\alpha_i^2}{\sum_{i=0}^{n}p(\lambda_i)\alpha_i^2} \leq \delta + \frac{\sum_{i<(1-\delta)\lambda_1}p(\lambda_i)\alpha_i^2}{\sum_{i=0}^{n}p(\lambda_i)\alpha_i^2} \leq \delta + \frac{4n}{\lambda_1} \sup_{\lambda \in [0,(1-\delta)\lambda_1]} p(\lambda)\delta,$$

where the second inequality follows by splitting the sum in the numerator depending on whether $\lambda \geq (1 - \delta)\lambda_1$, or otherwise.

Observe that if we pick the polynomial $p(\lambda) = (\lambda/\lambda_1)^s$ for $s \overset{\text{def}}{=} \lceil 1/2 \delta \cdot \log 4n/\delta \rceil$ in the above bounds, the relative error is bounded by $O(\delta)$. 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 $x^d$ from Section 3.1 to show that the Lanczos method can do better. We use the polynomial $p(\lambda) = p_{s,d}(\lambda/\lambda_1)$ for $s = \lceil 1/2 \delta \cdot \log 4n/\delta \rceil$ as above, and $d = \lceil \sqrt{2s \cdot \log 2n/\delta} \rceil$. In this case, we know that for all $\lambda$ such that $|\lambda| \leq \lambda_1$, we have $|p(\lambda) - (\lambda/\lambda_1)^s| \leq \delta/n$. Hence, $p(\lambda_1) \geq 1 - \delta/n$, and

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

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

It remains to analyze the time taken by this algorithm to compute $\lambda_1(T)$. Let $t_A$ denote the number of operations required to compute $Av$, given a vector $v$. 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 $Av_1$, orthogonalize it with respect to $v_1, \ldots, v_0$, and scale it to norm 1 in order to obtain $v_{i+1}$. As in the Conjugate Gradient method, we have $Av_j \in \text{Span}\{v_0, \ldots, v_{j+1}\}$ for all $j < k$ and, hence, using the symmetry of $A$, we obtain, $v_j^T(Av_i) = v_i^T(Av_j) = 0$ 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; here, we ensure the basis vectors are orthonormal, instead of $A$-orthogonal as in the case of Conjugate Gradient.) 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 [47]). Thus, we have the following theorem:

**Theorem 6.5.** 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 over the choice of random $v$. 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 eigenvector $w$ of $T$ which achieves $\lambda_1(T)$ can be used to give a candidate for the the largest eigenvector of $A$, i.e., the vector $Vw$.

**Beyond the largest eigenvalue.** 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
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 polynomial.

### 6.5 Computing the Matrix Exponential

In this section 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 \( \exp(-s(I-W)) = e^{-s} \sum_{k \geq 0} \frac{s^k}{k!} W^k \) where \( W \) is the random walk matrix associated to a graph \( G = (V,E) \) defined in Section 6.2.2. In terms of the normalized Laplacian \( \mathcal{L} = I - W \), this is the same as \( \exp(-s\mathcal{L}) \). 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, 34]. 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 \( \exp(-s\mathcal{L}) = e^{-s} \sum_{k \geq 0} \frac{s^k}{k!} \). 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 such as the balanced version of the Sparsest Cut problem introduced in Section 6.2.2. More generally, fast computation of \( \exp(-A)v \) plays a crucial role, via the Matrix Multiplicative Weights Update method, in obtaining fast combinatorial algorithms to solve semi-definite programs, see [6, 42, 5].

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 3.2. Indeed, Theorem 3.3 can be used to compute a \( \delta \) approximation to \( \exp(-A)v \) in time \( O \left( (t_A + n) \sqrt{\|A\|} \log^2 \delta \right) \); similar to Theorem 6.2. However, Theorem 3.7 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 4? Indeed, we can use the rational approximation from Theorem 4.1 to obtain \( \| \exp(-A) - (S_d(A))^{-1} \| \leq 2^{-\Omega(d)} \), where \( (S_d(A))^{-1} \) is the approximation to \( \exp(-A)v \). For most applications an error of \( \delta = 1/poly(n) \) suffices, so it is sufficient to choose \( d = O(\log n) \). How do we compute \( (S_d(A))^{-1} v \)? 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 is suffices to speed a computation of this form. The first problem is that $\beta_i$s could be complex, as is indeed the case for the polynomial $S_d$ as discussed in Section 4. However, since $S_d$ has real coefficients, its complex roots appear as conjugates. Hence, we can combine the factors corresponding to the 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 $\sqrt{|A|}$. To see this, observe that $|\beta_i| \leq d$ (see [65]), and consider a matrix $A$ with $\lambda_1(A) \gg d$, and $\lambda_n(A) = 1$. For such a matrix, the condition number of $(A^2 - (\beta_i + \bar{\beta}_i)A + |\beta_i|^2 I)$ is $\Omega(\lambda^3(A)/d^2)$, which is approximately the square of the condition number of $A$ for small $d$.

Similarly, the rational approximations to $e^{-x}$ in Section 4.2 suggest the vector $p_d((I + A/\delta)^{-1})v$ as an approximation to $\exp(-A)v$, where $p_d$ is the polynomial given by Theorem 4.3. Once again, for any PSD matrix $A$, though the matrix $(I + A/\delta)$ is PSD, the condition number of $(I + A/\delta)$ 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, $O\left((t(A) + n)\sqrt{|A|} \log \frac{1}{\delta} \right)$ is the best result known for computing the matrix exponential-vector product for a general PSD matrix $A$, see [44].

The above approach of using rational approximations shows how to reduce the computation of $\exp(-A)v$ to a small number of linear systems involving the matrix $A$. For an important special class of matrices, we can exploit the fact that there exist algorithms that are much faster than Conjugate Gradient and allow us to approximate $(I + A/\delta)^{-1}u$, for a given $u$. In particular, for a symmetric and diagonally dominant (SDD) matrix $A$, there are powerful near-linear-time SDD system solvers [60, 31, 30] whose guarantees are given in the following theorem.

**Theorem 6.7.** Given an $n \times n$ SDD matrix $A$ with $m$ non-zero entries, a vector $v$, and $\delta_1 > 0$, there is an algorithm, in $\tilde{O}(m \log \frac{1}{\delta_1})$ time, computes a vector $u$ such that $\|u - A^{-1}v\|_A \leq \delta_1 \|A^{-1}v\|_A$. Moreover, $u = Zv$ where $Z$ depends on $A$ and $\delta_1$, and is such that $(1 - \delta_1)A^{-1} \preceq Z \preceq (1 + \delta_1)A^{-1}$.

At the end of this section, we show how to compute the coefficients of $p_d$ from Theorem 4.3 efficiently, and show that each coefficient is bounded by $d^{O(d)}$. Assuming this we show that we can compute $p_d((I + A/\delta)^{-1})v$ as an approximation to $\exp(-A)v$ in near-linear time using Theorem 6.7. Note that if $A$ is SDD, so is $(I + A/\delta)$. We repeatedly use the SDD solver of Theorem 6.7 to approximate $(I + A/\delta)^{-1}v$, for all $i = 1, \ldots, d$, and let $Z$ denote the linear operator such that the SDD solver returns the vector $Z'u$ as the approximation.

Let $B \defeq (I + A/\delta)^{-1}$. 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 $Z' - B' = \sum_{j=0}^{d-1} Z^{i-1-j} (Z - B) B^j$, and using $\|B\| \leq 1$, we obtain, $\|Z' - B'\| \leq \delta_1 \cdot i (1 + \delta)^i$. Thus, $\|p_d(Z) - p_d(B)\| \leq d^{O(d)} \cdot \delta_1 (1 + \delta)^d$. Hence, we can choose $\delta_1 = \delta \cdot d^{-O(\delta)}$ for the SDD solver in order for the final approximation to have error at most $\delta$. Since $d = \Theta(\log \frac{1}{\delta})$ suffices, this results in an overall running time of $\tilde{O}(m)$. We summarize the result in the following theorem.

**Theorem 6.8.** There is an algorithm that, given an SDD matrix $A$ with $m$ non-zero entries, a vector $v$, and $\delta \in (0, 1]$, computes a vector $u$ such that $\|\exp(-A)v - u\| \leq \delta \|v\|$ in time $\tilde{O}((m + n) \log(2 + \|A\|) \cdot \text{polylog } \frac{1}{\delta})$.

The above theorem was first proved in [44]. However, instead of computing the coefficients of the polynomial $p_d$ explicitly, the authors in [44] appealed to the Lanczos method from numerical linear algebra that

---

13 A matrix $A$ is said to be Symmetric and Diagonally Dominant (SDD) if it is symmetric, and for all $i, A_{ii} \geq \sum_{j \neq i} |A_{ij}|$. 

allows them to achieve the error guarantee of the approximating polynomial without explicit knowledge of the polynomial.

Coming back to graphs, an important corollary of this theorem is that \( \exp(-sL) \nu \), the distribution after an \( s \)-length continuous time random walk on the graph with normalized Laplacian \( L \) starting with a distribution \( \nu \), can be approximately computed in \( \tilde{O}(m \log s) \) time. Recall that for simple random walks, from Section \ref{sec:graphs} we do not know how to do better than \( O(m \sqrt{s}) \) time.

**Computing the coefficients of \( p_d \).** We now address the issue of explicitly computing the coefficients of \( p_d \). It suffices to compute them to a precision of \( 2^{-\poly(d)} \) and we present the salient steps. Recall that in the proof of Theorem \ref{thm:approximation}, the polynomial \( r_{d-1}(t) \) that minimizes \( \int_1^\infty \left( \frac{t^d}{t} - r_{d-1}(t) \right)^2 \) (see Equation \ref{eq:rd}) is given by \( r_{d-1}(t) = \sum_{k=0}^{d-1} \sqrt{2k+1} \cdot \gamma_k \cdot L_k(t) \). The Legendre polynomials can be written as \( L_k(t) = 2^{-k} \sum_{i=0}^{[k/2]} k^{2i} \cdot \left( \frac{k}{i} \right) \binom{2k-2i}{k} \), see \cite{[2]} Chapter 22. Thus, assuming we know \( \{\gamma_k\}_{k=0}^{d-1} \), we can compute the coefficients of \( r_{d-1} \) in \( \poly(d) \) operations, and the sizes of the coefficients of \( r_{d-1} \) can only be \( 2^{O(d)} \) larger. Since \( q_d(y) = \int_y^1 r_{d-1}(t) \) \( dt \), 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 \( p_d(x) \equiv q_d(1-2x) \). Hence, given the coefficients of \( q_d \), those of \( p_d \) can be calculated in \( \poly(d) \) 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 \ref{eq:gamma}, we have \( \gamma_k = -d \int_0^\infty \left( \frac{z}{z+1} \right)^k e^{-dz} G_k(d(1+z)) \) \( dz \). The Laguerre polynomials (of order \( 1 \)) \( G_k \) are explicitly given as \( G_k(t) = \sum_{i=0}^k (-1)^i \binom{k}{i} t^i \) \( i \leq k \) \( \) \cite{[2]} Chapter 22. After using this expansion for \( G_k \), it suffices to compute the integrals \( \int_0^\infty \frac{z^i}{(z+1)^j} e^{-dz} \) \( dz \) for \( 0 \leq i \leq j \leq d \). If we know the values of these integrals, we can compute \( \gamma_k \) in \( \poly(d) \) operations, though the coefficients may now increase by a factor of \( d^{O(d)} \). For any \( 0 \leq j \leq i \leq d \), substituting \( w = z + 1 \), we obtain \( \int_0^\infty \frac{z^i}{(z+1)^j} e^{-dz} \) \( dz = e^{-d} \int_1^\infty \frac{(w-1)^j}{w^i} e^{-dw} \) \( dw \). Since we can expand \( (w-1)^j \) using the Binomial theorem, it suffices to compute integrals of the form \( \int_1^\infty w^{-j} e^{-dw} \) \( dw \) for \( d \leq j \leq d \), where again we lost at most \( 2^d \) in the magnitude of the numbers. For \( j \leq 0 \), this is a simple integration. For \( j \geq 1 \), the integral can be expressed using the Exponential Integral \cite{[23]}. Hence, it has the following rapidly convergent power series for \( d > 1 \), which can be used both to compute \( E_j(d) \)'s and bound them easily:

\[
E_j(d) = \int_1^\infty w^{-j} e^{-dw} \, dw = \frac{e^{-d}}{d} \sum_{k=0}^\infty \frac{(-1)^k(j+k-1)!}{(j-1)! d^k},
\]

see \cite{[23]}. Combining everything, the coefficients of \( p_d \) can be approximated up to \( d^{-\Theta(d)} \) error in time \( \poly(d) \) using \( \poly(d) \) sized registers.

### 6.6 Matrix Inversion via Exponentiation

Our final application of approximation theory is a rather 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 \( \exp(-s A) v \). 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.

This result is an immediate corollary of Theorem \ref{thm:inverse} proved in Section \ref{sec:polynomial} which shows that we can approximate \( x^{-1} \) with a sum of a small number of exponentials, where the approximation is valid for all \( x \in [\delta, 1] \).
Theorem 6.9 (Corollary to Theorem 5.1, \[54\]). Given \( \varepsilon, \delta \in (0, 1] \), there exist \( \text{poly}(\log 1/\varepsilon \delta) \) numbers \( 0 < w_j, t_j = O(\text{poly}(1/\varepsilon \delta)) \), such that for all symmetric matrices \( A \) satisfying \( \varepsilon I \preceq A \preceq I \), we have \( (1 - \delta)A^{-1} \preceq \sum_j w_j e^{-t_j A} \preceq (1 + \delta)A^{-1} \).

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 \( O(\text{poly}(1/\varepsilon \delta)) \), we lose only a polynomial factor in the approximation error.

Acknowledgments. We would like to thank Elisa Celis and Oded Regev for useful discussions. We would also like to thank László Babai for helpful comments.

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] Noga Alon and V. D. Milman. \( \lambda_1 \), isoperimetric inequalities for graphs, and superconcentrators. J. Comb. Theory, Ser. B, 38(1):73–88, 1985.

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

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

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

[7] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald 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] Gregory Beylkin and Lucas Monzón. On approximation of functions by exponential sums. Applied and Computational Harmonic Analysis, 19(1):17 – 48, 2005.

[10] Gregory Beylkin and Lucas Monzón. Approximation by exponential sums revisited. Applied and Computational Harmonic Analysis, 28(2):131 – 149, 2010. Special Issue on Continuous Wavelet Transform in Memory of Jean Morlet, Part I.

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

[12] Mark Bun and Justin 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] Fan R.K. Chung. *Spectral Graph Theory (CBMS Regional Conference Series in Mathematics, No. 92)*. American Mathematical Society, 1997.

[18] 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.

[19] 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.

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

[21] Lov 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.

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

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

[24] Rahul Jain, Zhengfeng Ji, Sarvagya Upadhyay, and John Watrous. QIP=PSPACE. *J. ACM*, 58(6):30:1–30:27, December 2011.

[25] Rahul Jain, Sarvagya Upadhyay, and John 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, Washington, DC, USA, 2009. IEEE Computer Society.

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

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

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

[29] Shmuel Kaniel. Estimates for some computational techniques in linear algebra. *Math. Comp.*, 20:369–378, 1966.
[30] Jonathan A. Kelner, Lorenzo Orecchia, Aaron Sidford, and Zeyuan Allen 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, New York, NY, USA, 2013. ACM.

[31] Ioannis Koutis, Gary L. Miller, and Richard 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, Washington, DC, USA, 2011. IEEE Computer Society.

[32] 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.

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

[34] David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. Markov chains and mixing times. American Mathematical Society, 2006.

[35] Andrei Andreyevich Markov. Ob odnom voproc D.I. Mendeleeva. Zapiski Imperatorskoi Akademii Nauk SPb, 62:1–24, 1890.

[36] V. A. Markov. O funktsiyakh, naimeneye uklonyayushchikhsya ot nulya vdannom promezhutke. 1892.

[37] Milena Mihail. Conductance and convergence of Markov chains-A combinatorial treatment of expanders. In FOCS, pages 526–531, 1989.

[38] Michael Mitzenmacher and Eli Upfal. Probability and computing: Randomized algorithms and probabilistic analysis. Cambridge University Press, New York, NY, USA, 2005.

[39] Donald J. Newman. Rational approximation to \( |x| \). Michigan Math. J., 11:11–14, 1964.

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

[41] Noam Nisan and Mario Szegedy. On the degree of Boolean functions as real polynomials. computational complexity, 4(4):301–313, 1994.

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

[43] Lorenzo Orecchia, Sushant Sachdeva, and Nisheeth K. Vishnoi. Approximating the exponential, the Lanczos method and an \( \tilde{O}(m) \)-time spectral algorithm for Balanced Separator. CoRR, abs/1111.1491, 2011.

[44] Lorenzo Orecchia, Sushant Sachdeva, and Nisheeth K. Vishnoi. Approximating the exponential, the Lanczos method and an \( O(m) \)-time spectral algorithm for Balanced Separator. STOC ’12, pages 1141–1160, 2012.

[45] Lorenzo Orecchia, Leonard J. Schulman, Umesh V. Vazirani, and Nisheeth K. Vishnoi. On partitioning graphs via single commodity flows. In STOC ’08: Proc. 40th Ann. ACM Symp. Theory of Computing, pages 461–470, 2008.
[46] Lorenzo Orecchia and Nisheeth 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.

[47] Victor Y. Pan and Zhao Q. Chen. The complexity of the matrix eigenproblem. In *STOC’99*, pages 507–516, 1999.

[48] Beresford N Parlett. *The symmetric eigenvalue problem*, volume 7. SIAM, 1980.

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

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

[51] 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.

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

[53] Yousef Saad and Henk A. van der Vorst. Iterative solution of linear systems in the 20th century. *Journal of Computational and Applied Mathematics*, 123(12):1 – 33, 2000. Numerical Analysis 2000. Vol. III: Linear Algebra.

[54] S. Sachdeva and N K. Vishnoi. Matrix inversion is as easy as exponentiation. *ArXiv e-prints*, May 2013.

[55] 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.

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

[57] Jonah Sherman. Breaking the multicommodity flow barrier for $O(\sqrt{\log n})$-approximations to Sparsest Cut. In *FOCS’09: Proc. 50th Ann. IEEE Symp. Foundations of Computer Science*, 2009.

[58] Jonah Sherman. Nearly maximum flows in nearly linear time. In *FOCS’13: Proc. 54th Ann. IEEE Symp. Foundations of Computer Science*, 2013.

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

[60] Daniel A. Spielman and Shang-Hua 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.

[61] G. Szegö. Über eine Eigenschaft der Exponentialreihe. *Sitzungsber. Berl. Math. Ges.*, 23:50–64, 1924. cited By (since 1996)37.

[62] Gabor Szegő. *Orthogonal polynomials*. American Mathematical Society Providence, 4th ed. edition, 1939.

[63] Nisheeth K Vishnoi. $Lx = b$. *Foundations and Trends in Theoretical Computer Science*, 8(1-2):1–141, 2012.
[64] Karl Weierstrass. Über die analytische Darstellbarkeit sogenannter willkürlicher Functionen einer reellen veränderlichen. \textit{Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften zu Berlin}, 2:633–639, 1885.

[65] Stephen M. Zemyan. On the zeroes of the Nth partial sum of the exponential series. \textit{The American Mathematical Monthly}, 112(10):pp. 891–909, 2005.