arXiv:1708.07788v1 [cs.DS] 25 Aug 2017

Stability of the Lanczos Method for Matrix Function Approximation

Cameron Musco
MIT
cnmusco@mit.edu

Christopher Musco
MIT
cpmusco@mit.edu

Aaron Sidford
Stanford
sidford@stanford.edu

August 28, 2017

Abstract

Theoretically elegant and ubiquitous in practice, the Lanczos method can approximate $f(\mathbf{A})\mathbf{x}$ for any symmetric matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, vector $\mathbf{x} \in \mathbb{R}^n$, and function f. In *exact arithmetic*, the method's error after k iterations is bounded by the error of the best degree-k polynomial uniformly approximating the scalar function $f(x)$ on the range $[\lambda_{\min}(\mathbf{A}), \lambda_{\max}(\mathbf{A})]$. However, despite decades of work, it has been unclear if this powerful guarantee holds in finite precision.

We resolve this problem, proving that when $\max_{x\in[\lambda_{\min},\lambda_{\max}]} |f(x)| \leq C$, Lanczos essentially matches the exact arithmetic guarantee if computations use roughly $\log(nC\|\mathbf{A}\|)$ bits of precision. Our proof extends work of Druskin and Knizhnerman [DK91], leveraging the stability of the classic Chebyshev recurrence to bound the stability of any polynomial approximating $f(x)$.

We also study the special case of $f(\mathbf{A}) = \mathbf{A}^{-1}$ for positive definite $\mathbf{A}$, where stronger guarantees hold for Lanczos. In exact arithmetic the algorithm performs as well as the best polynomial approximating $1/x$ *at each of* $\mathbf{A}$*'s eigenvalues*, rather than on the full range $[\lambda_{\min}(\mathbf{A}), \lambda_{\max}(\mathbf{A})]$. In seminal work, Greenbaum gives a natural approach to extending this bound to finite precision: she proves that finite precision Lanczos and the related conjugate gradient method match any polynomial approximating $1/x$ *in a tiny range around each eigenvalue* [Gre89].

For $\mathbf{A}^{-1}$, Greenbaum's bound appears stronger than our result. However, we exhibit matrices with condition number κ where exact arithmetic Lanczos converges in polylog(κ) iterations, but Greenbaum's bound predicts at best $\Omega(\kappa^{1/5})$ iterations in finite precision. It thus cannot offer more than a polynomial improvement over the $O(\kappa^{1/2})$ bound achievable via our result for general $f(\mathbf{A})$. Our analysis bounds the power of *stable approximating polynomials* and raises the question of if they fully characterize the behavior of finite precision Lanczos in solving linear systems. If they do, convergence in less than poly(κ) iterations cannot be expected, even for matrices with clustered, skewed, or otherwise favorable eigenvalue distributions.

1 Introduction

The Lanczos method for iteratively tridiagonalizing a Hermitian matrix is one of the most important algorithms in numerical computation. Introduced for computing eigenvectors and eigenvalues [Lan50], it remains the standard algorithm for doing so over half a century later [Saa11]. It also underlies state-of-the-art iterative solvers for linear systems [HS52, Saa03].

More generally, the Lanczos method can be used to iteratively approximate any function of a matrix's eigenvalues. Specifically, given $f : \mathbb{R} \mapsto \mathbb{R}$, symmetric $\mathbf{A} \in \mathbb{R}^{n\times n}$ with eigendecomposition $\mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$, and vector $\mathbf{x} \in \mathbb{R}^n$, it approximates $f(\mathbf{A})\mathbf{x}$, where:

$$f(\mathbf{A}) \stackrel{\text{def}}{=} \mathbf{V}f(\mathbf{\Lambda})\mathbf{V}^T.$$

$f(\mathbf{\Lambda})$ is the result of applying f to each diagonal entry of $\mathbf{\Lambda}$, i.e., to the eigenvalues of $\mathbf{A}$. In the special case of linear systems, $f(x) = 1/x$ and $f(\mathbf{A}) = \mathbf{A}^{-1}$. Other important matrix functions include the matrix log, the matrix exponential, the matrix sign function, and the matrix square root [Hig08]. These functions are broadly applicable in scientific computing, and are increasingly used in theoretical computer science [AK07, OSV12, SV14] and machine learning [HMS15, FMMS16, US16, AZL17, TPGV16]. In theses areas, there is interest in obtaining worst-case, end-to-end runtime bounds for approximating $f(\mathbf{A})\mathbf{x}$ up to a given precision.

The main idea behind the Lanczos method is to iteratively compute an orthonormal basis $\mathbf{Q}$ for the rank-k *Krylov subspace* $\mathcal{K}_k = [\mathbf{x}, \mathbf{A}\mathbf{x}, \mathbf{A}^2\mathbf{x}, \ldots, \mathbf{A}^{k-1}\mathbf{x}]$. The method then approximates $f(\mathbf{A})\mathbf{x}$ with a vector in $\mathcal{K}_k$ – i.e. with $p(\mathbf{A})\mathbf{x}$ for some polynomial p with degree $< k$.

Specifically, along with $\mathbf{Q}$, the algorithm computes $\mathbf{T} = \mathbf{Q}^T\mathbf{A}\mathbf{Q}$ and approximates $f(\mathbf{A})\mathbf{x}$ with $\mathbf{y} = \|\mathbf{x}\| \cdot \mathbf{Q}f(\mathbf{T})\mathbf{e}_1$.[1] Importantly, $\mathbf{y}$ can be computed efficiently: iteratively constructing $\mathbf{Q}$ and $\mathbf{T}$ requires just $k-1$ matrix-vector multiplications with $\mathbf{A}$. Furthermore, due to a special iterative construction, $\mathbf{T}$ is tridiagonal. It is thus possible to accurately compute its eigendecomposition, and hence apply arbitrary functions $f(\mathbf{T})$, including $\mathbf{T}^{-1}$, in $\tilde{O}(k^2)$ time.

Note that $\mathbf{y} \in \mathcal{K}_k$ and so can be written as $p(\mathbf{A})\mathbf{x}$ for some polynomial p. While this is not necessarily the polynomial minimizing $\|p(\mathbf{A})\mathbf{x} - f(\mathbf{A})\mathbf{x}\|$, $\mathbf{y}$ still satisfies:

$$\|f(\mathbf{A})\mathbf{x} - \mathbf{y}\| \leq 2\|\mathbf{x}\| \cdot \min_{\substack{\text{polynomial } p\\ \text{with degree } < k}} \left(\max_{x\in[\lambda_{\min}(\mathbf{A}),\lambda_{\max}(\mathbf{A})]} |f(x) - p(x)| \right). \tag{1}$$

where $\lambda_{\max}(\mathbf{A})$ and $\lambda_{\min}(\mathbf{A})$ are the largest and smallest eigenvalues of $\mathbf{A}$ respectively. That is, up to a factor of 2, the error of Lanczos in approximating $f(\mathbf{A})\mathbf{x}$ is bounded by the uniform error of the best polynomial approximation to f with degree $< k$. Thus, to bound the performance of Lanczos after k iterations, it suffices to prove the existence of any degree-k polynomial approximating the scalar function f, even if the explicit polynomial is not known.

2 Our contributions

Unfortunately, as has been understood since its introduction, the performance of the Lanczos algorithm in exact arithmetic does not predict its behavior when implemented in finite precision.

[1] Here $\mathbf{e}_1$ is the first standard basis vector. There are a number of variations on the Lanczos method, especially for the case of solving linear systems, however we consider just this simple, general version.

Specifically, it is well known that the basis $\mathbf{Q}$ loses orthogonality. This leads to slower convergence when computing eigenvectors and values, and a wide range of reorthogonalization techniques have been developed to remedy the issue (see e.g. [PS79, Sim84] or [Par98, MS06] for surveys).

However, in the case of matrix function approximation, these remedies appear unnecessary. Vanilla Lanczos continues to perform well in practice, despite loss of orthogonality. In fact, it even converges when $\mathbf{Q}$ has numerical rank $\ll k$ and thus cannot span $\mathcal{K}_k$. Understanding when and why the Lanczos algorithm runs efficiently in the face of numerical breakdown has been the subject of intensive research for decades – we refer the reader to [MS06] for a survey. Nevertheless, despite experimental and theoretical evidence, no iteration bounds comparable to the exact arithmetic guarantees were known for general matrix function approximation in finite precision.

2.1 General function approximation in finite precision

Our main positive result closes this gap for general functions by showing that a bound nearly matching (1) holds even when Lanczos is implemented in finite precision. In Section 6 we show:

Theorem 1 (Function Approximation via Lanczos in Finite Arithmetic)**.** *Given real symmetric* $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{x} \in \mathbb{R}^n$, $\eta \leq \|\mathbf{A}\|$, $\epsilon \leq 1$, *and any function* f *with* $|f(x)| < C$ *for* $x \in [\lambda_{\min}(\mathbf{A}) - \eta, \lambda_{\max}(\mathbf{A}) + \eta]$, *let* $B = \log\left(\frac{nk\|\mathbf{A}\|}{\epsilon\eta}\right)$. *The Lanczos algorithm run on a floating point computer with* $\Omega(B)$ *bits of precision for* k *iterations returns* $\mathbf{y}$ *satisfying:*

$$\|f(\mathbf{A})\mathbf{x} - \mathbf{y}\| \leq (7k \cdot \delta_k + \epsilon C)\|\mathbf{x}\| \tag{2}$$

where

$$\delta_k \stackrel{\text{def}}{=} \min_{\substack{\text{polynomial } p \\ \text{with degree } < k}} \left(\max_{x\in[\lambda_{\min}(\mathbf{A})-\eta, \lambda_{\max}(\mathbf{A})+\eta]} |p(x) - f(x)| \right).$$

If basic arithmetic operations on floating point numbers with $\Omega(B)$ *bits of precision have runtime cost* $O(1)$, *the algorithm's runtime is* $O(\mathrm{mv}(\mathbf{A})k + k^2B + kB^2)$, *where* $\mathrm{mv}(\mathbf{A})$ *is the time required to multiply the matrix* $\mathbf{A}$ *with a vector.*

The bound of (2) matches (1) up to an $O(k)$ factor along with a small ϵC additive error term, which decreases exponentially in the bits of precision available. For typical functions, the degree of the best uniform approximating polynomial depends logarithmically on the desired accuracy. So the $O(k)$ factor equates to just a logarithmic increase in the degree of the approximating polynomial, and hence the number of iterations required for a given accuracy. The theorem requires a uniform approximation bound on the slightly extended range $[\lambda_{\min}(\mathbf{A}) - \eta, \lambda_{\max}(\mathbf{A}) + \eta]$, however in typical cases this has essentially no effect on the bounds obtainable.

In Section 8 we give several example applications of Theorem 1 that illustrate these principles. We show how to stably approximate the matrix sign function, the matrix exponential, and the top singular value of a matrix. Our runtimes all either improve upon or match state-of-the-art runtimes, while holding rigorously under finite precision computation. They demonstrate the broad usefulness of the Lanczos method and our approximation guarantees for matrix functions.

Techniques and comparison to prior work

We begin with the groundbreaking work of Paige [Pai71, Pai76, Pai80], which gives a number of results on the behavior of the Lanczos tridiagonalization process in finite arithmetic. Using Paige's bounds, we demonstrate that if $f(x)$ is a degree $< k$ Chebyshev polynomial of the first kind, Lanczos can apply it very accurately. This proof, which is the technical core of our error bound, leverages the well-understood stability of the recursive formula for computing Chebyshev polynomials [Cle55], *even though this formula is not explicitly used when applying Chebyshev polynomials via Lanczos.*

To extend this result to general functions, we first show that Lanczos will effectively apply the 'low degree polynomial part' of $f(\mathbf{A})$, incurring error depending on the residual δ_k (see Lemma 11). So we just need to show that this polynomial component can be applied stably. To do so, we appeal to our proof for the special case of Chebyshev polynomials via the following argument, which appears formally in the proof of Lemma 9: If $|f(x)| \leq C$ on $[\lambda_{\min}(\mathbf{A}), \lambda_{\max}(\mathbf{A})]$, then the optimal degree k polynomial approximating $f(x)$ on this range is bounded by $2C$ in absolute value since it must have uniform error $< C$, the error given by setting $p(x) = 0$. Since its magnitude is bounded, this polynomial has coefficients bounded by $O(C)$ when written in the Chebyshev basis. Accordingly, by linearity, Lanczos only incurs error $O(C)$ times greater than what is obtained when applying Chebyshev polynomials. This yields the additive error bound ϵC in Theorem 1, proving that, *for any bounded function, Lanczos can apply the optimal approximating polynomial accurately.*

Ultimately, our proof can be seen as a more careful application of the techniques of Druskin and Knizhnerman [DK91, DK95]. They also use the stability of Chebyshev polynomials to understand stability for more general functions, but give an error bound which depends on a coarse upper bound for δ_k. Additionally, their work ignores stability issues that can arise when computing the final output $\mathbf{y} = \|\mathbf{x}\| \cdot \mathbf{Q} f(\mathbf{T}) \mathbf{e}_1$. We provide a complete analysis by showing that $\mathbf{y}$ can be computed stably whenever $f(x)$ is well approximated by a low degree polynomial, and hence give the first end-to-end runtime bound for Lanczos in finite arithmetic.

Our work is also similar to that of Orecchia, Sachdeva, and Vishnoi, who give accuracy bounds for a slower variant of Lanczos with re-orthogonalization that requires $\sim O(\mathrm{mv}(\mathbf{A})k + k^3)$ time, in contrast to the $\sim O(\mathrm{mv}(\mathbf{A})k + k^2)$ time required for our Theorem 1 [OSV12]. Furthermore, their results require a bound on the coefficients of the polynomial $p(x)$. Many optimal approximating polynomials, like the Chebyshev polynomials, have coefficients which are exponential in their degree. Accordingly, [OSV12] requires that the number of bits used to match such polynomials with Lanczos grows polynomially (rather than logarithmically) with the approximating degree. In fact, as shown in [FMMS16], any degree k polynomial with coefficients bounded by C can be well approximated by a polynomial with degree $O(\sqrt{k \log(kC)})$. So [OSV12] only gives good bounds for polynomials that are inherently suboptimal. Additionally, like Druskin and Knizhnerman, [OSV12] only addresses roundoff errors that arise during matrix vector multiplication with $\mathbf{A}$, assuming stability for other components of their algorithm.

2.2 Linear systems in finite precision

Theorem 1 shows that for general functions, the Lanczos method performs nearly as accurately in finite precision as in exact arithmetic: after k iterations, it still nearly matches the accuracy of the best degree $< k$ uniform polynomial approximation to $f(x)$ over $\mathbf{A}$'s eigenvalue range.

However, in the important special case of solving positive definite linear systems, i.e., when $\mathbf{A}$ has all positive eigenvalues and $f(\mathbf{A}) = \mathbf{A}^{-1}$, it is well known that (1) can be strengthened in exact arithmetic. Lanczos performs as well as the best polynomial approximating $f(x) = 1/x$ *at each of* $\mathbf{A}$*'s eigenvalues* rather than over the full range $[\lambda_{\min}(\mathbf{A}), \lambda_{\max}(\mathbf{A})]$. Specifically,[2]

$$\|\mathbf{A}^{-1}\mathbf{x} - \mathbf{y}_k\| \leq \sqrt{\kappa(\mathbf{A})} \cdot \|\mathbf{x}\| \cdot \min_{\substack{\text{polynomial } p\\ \text{with degree } < k}} \max_{x \in \{\lambda_1(\mathbf{A}), \lambda_2(\mathbf{A}), \ldots, \lambda_n(\mathbf{A})\}} |p(x) - 1/x| \tag{3}$$

where $\kappa(\mathbf{A}) = \|\mathbf{A}\|\|\mathbf{A}^{-1}\|$ is $\mathbf{A}$'s condition number. (3) is proven in Appendix B. It can be much stronger than (1), and correspondingly Theorem 1. Specifically, the best bound obtainable from (1) is that after $\tilde{O}(\sqrt{\kappa(\mathbf{A})})$ iterations, $\mathbf{y} \approx \|\mathbf{A}^{-1}\mathbf{x}\|$. In contrast, (3) shows that even when $\kappa(\mathbf{A})$ is very large, n iterations are enough to compute $\mathbf{A}^{-1}\mathbf{x}$ exactly: $p(x)$ can be set to the polynomial which exactly interpolates $1/x$ at each of $\mathbf{A}$'s eigenvalues. (3) also gives improved bounds for matrices with clustered, skewed, or otherwise favorable eigenvalue distributions [AL86, DH07]. For example, assuming exact arithmetic, it can be used to analyze preconditioners for graph Laplacians, which induce heavily skewed eigenvalue distributions [SW09, DPSX17]. It can also be applied to algorithms for solving asymmetric Laplacian systems corresponding to *directed graphs* [CKP+16].

Understanding whether (3) carries over to finite precision is an important open question, which has actually received more attention than the general matrix function problem. In seminal work, Greenbaum [Gre89] gives a natural finite precision extension of (3): performance can be bounded by the error in approximating $1/x$ in a *tiny range around each eigenvalue*. Here "tiny" means essentially on the order of machine precision – the approximation need only be over ranges of width η as long as the bits of precision used is $\gtrsim \log(1/\eta)$.

Greenbaum's bound applies to the conjugate gradient (CG) method, a somewhat optimized way of applying Lanczos to linear systems. A precise version of Theorem 3 in [Gre89] can be summarized as follows (see Appendix B for a detailed discussion):

Theorem 2 (Conjugate Gradient in Finite Arithmetic [Gre89])**.** *Given positive definite* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *and* $\mathbf{x} \in \mathbb{R}^n$*, after* k *iterations, the conjugate gradient algorithm run on a computer with* $\Omega\left(\log \frac{nk\|\mathbf{A}\|}{\min(\eta, \lambda_{\min}(\mathbf{A}))}\right)$ *bits of precision returns* $\mathbf{y}$ *satisfying:*

$$\|\mathbf{A}^{-1}\mathbf{x} - \mathbf{y}\| \leq 2\kappa(\mathbf{A}) \cdot \bar{\delta}_k \|\mathbf{x}\|$$

where

$$\bar{\delta}_k \stackrel{\text{def}}{=} \min_{\substack{\text{polynomial } p\\ \text{with degree } < k}} \left(\max_{x \in \bigcup_{i=1}^n [\lambda_i(\mathbf{A}) - \eta, \lambda_i(\mathbf{A}) + \eta]} |p(x) - 1/x| \right).$$

The CG algorithm run for k *iterations requires* $O(\mathrm{mv}(\mathbf{A})k + nk)$ *time, where* $\mathrm{mv}(\mathbf{A})$ *is the time required to multiply* $\mathbf{A}$ *by a vector.*

Theorem 2 does not apply to general matrix functions but, at least for the special case of $f(\mathbf{A}) = \mathbf{A}^{-1}$, it is stronger than our Theorem 1. It is natural to ask by how much.

Lower bound

Surprisingly, we show that Greenbaum's bound is much weaker than the exact arithmetic guarantee (3), and in fact is not significantly more powerful than Theorem 1. Specifically, in Section 7 we

[2] Note that slightly stronger bounds where p depends on $\mathbf{x}$ are available. We work with (3) for simplicity since it only depends on $\mathbf{A}$'s eigenvalues.

prove that for any κ and interval width η, there is a natural class of matrices with condition number κ and just $O(\log\kappa\cdot\log 1/\eta)$ eigenvalues for which any 'stable approximating polynomial' of the form required by Theorem 2 achieving $\bar{\delta}_k \leq 1/6$ must have degree $\Omega(\kappa^c)$ for a fixed constant $c \geq 1/5$.

Theorem 3 (Stable Approximating Polynomial Lower Bound)**.** *There exists a fixed constant* $1/5 \leq c \leq 1/2$ *such that for any* $\kappa \geq 2$, $0 < \eta \leq \frac{1}{20\kappa^2}$, *and* $n \geq \lfloor \log_2 \kappa \rfloor \cdot \lceil \ln 1/\eta \rceil$, *there is a positive definite* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *with condition number* $\leq \kappa$, *such that for any* $k < \lfloor \kappa^c/377 \rfloor$*:*

$$\bar{\delta}_k \stackrel{\text{def}}{=} \min_{\substack{\text{polynomial } p\\ \text{with degree } < k}} \left(\max_{x\in\bigcup_{i=1}^n [\lambda_i(\mathbf{A})-\eta,\lambda_i(\mathbf{A})+\eta]} |p(x) - 1/x| \right) \geq 1/6.$$

Theorem 3 immediately gives a strong lower bound against Greenbaum's result, even if we only require constant factor error. Setting $\log(1/\eta) = n/\log(\kappa)$ we have:

Corollary 4. *There exists a fixed* $c \geq 1/5$ *such that for any* $\kappa \geq 2$, *there is a positive definite* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *with condition number* $\leq \kappa$ *such that Theorem 2 predicts that CG must run for* $\Omega(\kappa^c)$ *iterations to guarantee* $\|\mathbf{A}^{-1}\mathbf{x} - \mathbf{y}\| \leq \frac{\kappa\cdot\|\mathbf{x}\|}{3}$ *if* $o(n/\log\kappa)$ *bits of precision are used.*

As a consequence, if we set $\kappa = n^d$ for arbitrarily large constant d, Theorem 2 only guarantees a $\Omega(n^{cd})$ iteration bound, even when the precision used is *nearly exponential* in n. Since $O(\kappa^{1/2}) = O(n^{d/2})$ is already achievable via Theorem 1 with $O(\log n)$ bits of precision, Greenbaum's bound is not a significant improvement, except in very high precision regimes. While our constant c is $< 1/2$, we believe the proof can be tightened to show that $\sim \kappa^{1/2}$ degree is necessary.

Corollary 4 can also be interpreted as showing the existence of matrices with $O(\log^2\kappa)$ eigenvalues for which Theorem 2 requires $\Omega(\kappa^c)$ iterations for convergence if $O(\log\kappa)$ bits of precision are used. This is nearly exponentially worse than the exact arithmetic case, where (3) gives convergence to perfect accuracy in $O(\log^2\kappa)$ iterations.

Theorem 3 seems damning for establishing iteration bounds on the Lanczos and CG methods in finite precision that go significantly beyond uniform approximation of $1/x$. Informally, all known bounds improving on $O(\sqrt{\kappa})$ iterations, including those for clustered or skewed eigenvalue distributions, require a polynomial that stably approximates $1/x$ on some small subset of poorly conditioned eigenvalues. We rule out the existence of such polynomials.

However, Theorem 3 is not a general lower bound on the performance of finite precision Lanczos methods for solving linear systems. It is possible that these methods do something "smarter" than applying a fixed stable polynomial. Thus, we see our result as pointing to two possibilities:

> **Optimistic:** Bounds comparable (3) can be proven for finite precision Lanczos or conjugate gradient, but are out of the reach of current techniques. Proving such bounds may require looking beyond a "polynomial" view of these methods.
>
> **Pessimistic:** For finite precision Lanczos methods to converge in k iterations, there must essentially exist a stable degree k polynomial approximating $1/x$ in small ranges around $\mathbf{A}$'s eigenvalues. If this is the case, our lower bound could be extended to an unconditional lower bound on the number of iterations required for solving $\mathbf{A}^{-1}\mathbf{x}$ with such methods.

3 Notation and linear algebra preliminaries

Notation We use bold uppercase letters for matrices and lowercase letters for vectors (i.e. matrices with multiple rows, 1 column). A lowercase letter with a subscript is used to denote a particular column vector in the corresponding matrix. E.g. $\mathbf{q}_5$ denotes the 5^{th} column in the matrix $\mathbf{Q}$. Non-bold letters denote scalar quantities. A superscript ^{T} denotes the transpose of a matrix or vector. $\mathbf{e}_i$ denotes the i^{th} standard basis vector, i.e. a vector with a 1 at position i and 0's elsewhere. Its length will be clear from context. We use $\mathbf{I}_{k\times k}$ to denote the $k \times k$ identity matrix, removing the subscript when it is clear from context. When discussing runtimes, we occasionally use $\tilde{O}(x)$ as shorthand for $O(x \log^c x)$, where c is a fixed positive constant.

Matrix Functions The main subject of this work is matrix functions and their approximation by matrix polynomials. We define matrix functions in the standard way, via the eigendecomposition:

Definition 1 (Matrix Function)**.** *For any function $f : \mathbb{R} \to \mathbb{R}$, for any real symmetric matrix $\mathbf{M}$, which can be diagonalized as $\mathbf{M} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$, we define the matrix function $f(\mathbf{M})$ as:*

$$f(\mathbf{M}) \stackrel{\text{def}}{=} \mathbf{V} f(\mathbf{\Lambda})\mathbf{V}^T,$$

where $f(\mathbf{\Lambda})$ is a diagonal matrix obtained by applying f independently to each eigenvalue on the diagonal of $\mathbf{\Lambda}$ (including any 0 eigenvalues).

Other For a vector $\mathbf{x}$, $\|\mathbf{x}\|$ denotes the Euclidean norm. For a matrix $\mathbf{M}$, $\|\mathbf{M}\|$ denotes the spectral norm and $\kappa(\mathbf{M}) = \|\mathbf{M}\|\|\mathbf{M}^{-1}\|$ the condition number. We denote the eigenvalues of a symmetric matrix $\mathbf{M} \in \mathbb{R}^{n\times n}$ by $\lambda_1(\mathbf{M}) \geq \lambda_2(\mathbf{M}) \geq \ldots \geq \lambda_n(\mathbf{M})$, often writing $\lambda_{\max}(\mathbf{M}) \stackrel{\text{def}}{=} \lambda_1(\mathbf{M})$ and $\lambda_{\min}(\mathbf{M}) \stackrel{\text{def}}{=} \lambda_n(\mathbf{M})$. $\text{nnz}(\mathbf{M})$ denotes the number of non-zero entries in $\mathbf{M}$.

4 The Lanczos method in exact arithmetic

We begin by presenting the classic Lanczos method and demonstrate how it can be used to approximate $f(\mathbf{A})\mathbf{x}$ for any function f and vector $\mathbf{x}$ when computations are performed in *exact arithmetic*. While the results in this section are well known, we include an analysis that will mirror and inform our eventual finite precision analysis.

We study the standard implementation of Lanczos described in Algorithm 1. In exact arithmetic, the algorithm computes an orthonormal matrix $\mathbf{Q}$ with $\mathbf{q}_1 = \mathbf{x}/\|\mathbf{x}\|$ as its first column such that for all $j \leq k$, $[\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_j]$ spans the rank-j Krylov subspace:

$$\mathcal{K}_j = [\mathbf{x}, \mathbf{A}\mathbf{x}, \mathbf{A}^2\mathbf{x}, \ldots, \mathbf{A}^{j-1}\mathbf{x}]. \tag{4}$$

The algorithm also computes symmetric tridiagonal $\mathbf{T} \in \mathbb{R}^{k\times k}$ such that $\mathbf{T} = \mathbf{Q}^T\mathbf{A}\mathbf{Q}$.[3]

While the Krylov subspace interpretation of the Lanczos method is useful in understanding the function approximation guarantees that we will eventually prove, there is a more succinct way of characterizing the algorithm's output that doesn't use the notion of Krylov subspaces. It has been quite useful in analyzing the algorithm since the work of Paige [Pai71], and will be especially useful when we study the algorithm's behavior in finite arithmetic.

[3] For conciseness, we ignore the case when the algorithm terminates early because $\beta_{i+1} = 0$. In this case, either $\mathbf{A}$ has rank i or $\mathbf{x}$ only has a non-zero projection onto i eigenvectors of $\mathbf{A}$. Accordingly, for any $j \geq 1$ $\mathcal{K}_j$ is spanned by $\mathcal{K}_i$ so there is no need to compute additional vectors beyond $\mathbf{q}_i$: any polynomial $p(\mathbf{A})\mathbf{x}$ can be formed by recombining vectors in $[\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_i]$. It is tedious but not hard to check that our proofs go through in this case.

Algorithm 1 Lanczos Method for Computing Matrix Functions

input: symmetric $\mathbf{A} \in \mathbb{R}^{n \times n}$, # of iterations k, vector $\mathbf{x} \in \mathbb{R}^n$, function $f : \mathbb{R} \to \mathbb{R}$
output: vector $\mathbf{y} \in \mathbb{R}^n$ which approximates $f(\mathbf{A})\mathbf{x}$

$\mathbf{q}_0 = \mathbf{0}$, $\mathbf{q}_1 = \mathbf{x}/\|\mathbf{x}\|$, $\beta_1 = 0$
for $i \in 1, \ldots, k$ **do**
  $\mathbf{q}_{i+1} \leftarrow \mathbf{A}\mathbf{q}_i - \beta_i \mathbf{q}_{i-1}$
  $\alpha_i \leftarrow \langle \mathbf{q}_{i+1}, \mathbf{q}_i \rangle$
  $\mathbf{q}_{i+1} \leftarrow \mathbf{q}_{i+1} - \alpha_i \mathbf{q}_i$
  $\beta_{i+1} \leftarrow \|\mathbf{q}_{i+1}\|$
  if $\beta_{i+1} == 0$ **then**
    break loop
  end if
  $\mathbf{q}_{i+1} \leftarrow \mathbf{q}_{i+1}/\beta_{i+1}$
end for

$$\mathbf{T} \leftarrow \begin{bmatrix} \alpha_1 & \beta_2 & & 0 \\ \beta_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_k \\ 0 & & \beta_k & \alpha_k \end{bmatrix}, \qquad \mathbf{Q} \leftarrow \begin{bmatrix} \mathbf{q}_1 & \ldots & \mathbf{q}_k \end{bmatrix},$$

return $\mathbf{y} = \|\mathbf{x}\| \cdot \mathbf{Q} f(\mathbf{T}) \mathbf{e}_1$

Claim 5 – Exact Arithmetic (Lanczos Output Guarantee)**.** *Run for $k \leq n$ iterations using exact arithmetic operations, the Lanczos algorithm (Algorithm 1) computes $\mathbf{Q} \in \mathbb{R}^{n \times k}$, an additional column vector $\mathbf{q}_{k+1} \in \mathbb{R}^n$, a scalar β_{k+1}, and a symmetric tridiagonal matrix $\mathbf{T} \in \mathbb{R}^{k \times k}$ such that:*

$$\mathbf{A}\mathbf{Q} = \mathbf{Q}\mathbf{T} + \beta_{k+1}\mathbf{q}_{k+1}\mathbf{e}_k^T, \tag{5}$$

and

$$[\mathbf{Q}, \mathbf{q}_{k+1}]^T [\mathbf{Q}, \mathbf{q}_{k+1}] = \mathbf{I}. \tag{6}$$

Together (5) and (6) also imply that:

$$\lambda_{\min}(\mathbf{T}) \geq \lambda_{\min}(\mathbf{A}) \qquad \textit{and} \qquad \lambda_{\max}(\mathbf{T}) \leq \lambda_{\max}(\mathbf{A}). \tag{7}$$

When run for $k \geq n$ iterations, the algorithm terminates at the n^{th} iteration with $\beta_{n+1} = 0$.

We include a brief proof in Appendix E for completeness. The formulation of Claim 5 is valuable because it allows use to analyze how Lanczos applies polynomials via the following identity:

$$\mathbf{A}^q\mathbf{Q} - \mathbf{Q}\mathbf{T}^q = \sum_{i=1}^{q} \mathbf{A}^{q-i} \left(\mathbf{A}\mathbf{Q} - \mathbf{Q}\mathbf{T}\right) \mathbf{T}^{i-1}. \tag{8}$$

In particular, (5) gives an explicit expression for $(\mathbf{A}\mathbf{Q} - \mathbf{Q}\mathbf{T})$. Ultimately, our finite precision analysis is based on a similar expression for this central quantity.

4.1 Function approximation in exact arithmetic

We first show that Claim 5 can be used to prove (1): Lanczos approximates matrix functions essentially as well as the best degree k polynomial approximates the corresponding scalar function on the range of $\mathbf{A}$'s eigenvalues. We begin with a statement that applies for any function $f(x)$:

Theorem 6 – Exact Arithmetic (Approximate Application of Matrix Functions)**.** *Suppose* $\mathbf{Q} \in \mathbb{R}^{n\times k}$, $\mathbf{T} \in \mathbb{R}^{k\times k}$, β_{k+1}, *and* $\mathbf{q}_{k+1}$ *are computed by the Lanczos algorithm (Algorithm 1), run with exact arithmetic on inputs* $\mathbf{A}$ *and* $\mathbf{x}$. *Let*

$$\delta_k = \min_{\substack{\text{polynomial } p\\ \text{w/ degree} < k}} \left(\max_{x\in[\lambda_{\min}(\mathbf{A}),\lambda_{\max}(\mathbf{A})]} |f(x) - p(x)| \right).$$

Then the output $\mathbf{y} = \|\mathbf{x}\| \cdot \mathbf{Q}f(\mathbf{T})\mathbf{e}_1$ *satisfies:*

$$\|f(\mathbf{A})\mathbf{x} - \mathbf{y}\| \leq 2\delta_k\|\mathbf{x}\|. \tag{9}$$

Theorem 6 is proven from the following lemma, which says that the Lanczos algorithm run for k iterations can *exactly* apply any matrix polynomial with degree $< k$.

Lemma 7 – Exact Arithmetic (Exact Application of Polynomials)**.** *If* $\mathbf{A}$, $\mathbf{Q}$, $\mathbf{T}$, β_{k+1}, *and* $\mathbf{q}_{k+1}$ *satisfy* (5) *of Claim 5 (e.g. because they are computed with the Lanczos method), then for any polynomial* p *with degree* $< k$*:*

$$p(\mathbf{A})\mathbf{q}_1 = \mathbf{Q}p(\mathbf{T})\mathbf{e}_1.$$

Recall that in Algorithm 1, we set $\mathbf{q}_1 = \mathbf{x}/\|\mathbf{x}\|$, so the above trivially gives $p(\mathbf{A})\mathbf{x} = \|\mathbf{x}\|\mathbf{Q}p(\mathbf{T})\mathbf{e}_1$.

Proof. We show that for any integer $1 \leq q < k$:

$$\mathbf{A}^q\mathbf{q}_1 = \mathbf{Q}\mathbf{T}^q\mathbf{e}_1. \tag{10}$$

The lemma then follows by linearity as any polynomial p with degree $< k$ can be written as the sum of these monomial terms. To prove (10), we appeal to the telescoping sum in (8). Specifically, since $\mathbf{q}_1 = \mathbf{Q}\mathbf{e}_1$, (10) is equivalent to:

$$\left(\mathbf{A}^q\mathbf{Q} - \mathbf{Q}\mathbf{T}^q\right)\mathbf{e}_1 = \mathbf{0}. \tag{11}$$

For $q \geq 1$, (8) let's us write:

$$\left(\mathbf{A}^q\mathbf{Q} - \mathbf{Q}\mathbf{T}^q\right)\mathbf{e}_1 = \left(\sum_{i=1}^{q} \mathbf{A}^{q-i}\left(\mathbf{A}\mathbf{Q} - \mathbf{Q}\mathbf{T}\right)\mathbf{T}^{i-1}\right)\mathbf{e}_1.$$

Substituting in (5):

$$\left(\mathbf{A}^q\mathbf{Q} - \mathbf{Q}\mathbf{T}^q\right)\mathbf{e}_1 = \beta_{k+1}\sum_{i=1}^{q} \mathbf{A}^{q-i}\mathbf{q}_{k+1}\mathbf{e}_k^T\mathbf{T}^{i-1}\mathbf{e}_1. \tag{12}$$

Since $\mathbf{T}$ is tridiagonal, $\mathbf{T}^{i-1}\mathbf{e}_1$ is zero everywhere besides its first i entries. So, as long as $q < k$, $\mathbf{e}_k^T\mathbf{T}^{i-1}\mathbf{e}_1 = 0$ for all $i \leq q$. Accordingly, (12) evaluates to $\mathbf{0}$, proving (11) and Lemma 7. ∎

With Lemma 7 in place, Theorem 6 intuitively follows because Lanczos always applies the "low degree polynomial part" of $f(\mathbf{A})$. The proof is a simple application of triangle inequality.

Proof of Theorem 6.

$$\|f(\mathbf{A})\mathbf{x} - \mathbf{y}\| = \|f(\mathbf{A})\mathbf{q}_1 - \mathbf{Q}f(\mathbf{T})\mathbf{e}_1\| \cdot \|\mathbf{x}\| \tag{13}$$

For any polynomial p, we can write:

$$\begin{aligned} \|f(\mathbf{A})\mathbf{q}_1 - \mathbf{Q}f(\mathbf{T})\mathbf{e}_1\| &\leq \|p(\mathbf{A})\mathbf{q}_1 - \mathbf{Q}p(\mathbf{T})\mathbf{e}_1\| + \|\left[f(\mathbf{A}) - p(\mathbf{A})\right]\mathbf{q}_1 - \mathbf{Q}\left[f(\mathbf{T}) - p(\mathbf{T})\right]\mathbf{e}_1\| \\ &\leq 0 + \|\left[f(\mathbf{A}) - p(\mathbf{A})\right]\mathbf{q}_1\| + \|\mathbf{Q}\left[f(\mathbf{T}) - p(\mathbf{T})\right]\mathbf{e}_1\| \\ &\leq \|f(\mathbf{A}) - p(\mathbf{A})\| + \|\mathbf{Q}\|\|f(\mathbf{T}) - p(\mathbf{T})\|. \end{aligned} \tag{14}$$

In the second step we use triangle inequality, in the third we use Lemma 7 and triangle inequality, and in the fourth we use submultiplicativity of the spectral norm and the fact that $\|\mathbf{q}_1\| = \|\mathbf{e}_1\| = 1$.

$f(\mathbf{A}) - p(\mathbf{A})$ is symmetric and has an eigenvalue equal to $f(\lambda) - p(\lambda)$ for each eigenvalue λ of $\mathbf{A}$. Accordingly:

$$\|f(\mathbf{A}) - p(\mathbf{A})\| \leq \max_{x \in [\lambda_{\min}(\mathbf{A}), \lambda_{\max}(\mathbf{A})]} |f(x) - p(x)|.$$

Additionally, by (7) of Claim 5, for any eigenvalue $\lambda(\mathbf{T})$ of $\mathbf{T}$, $\lambda_{\min}(\mathbf{A}) \leq \lambda(\mathbf{T}) \leq \lambda_{\max}(\mathbf{A})$ so:

$$\|f(\mathbf{T}) - p(\mathbf{T})\| \leq \max_{x \in [\lambda_{\min}(\mathbf{A}), \lambda_{\max}(\mathbf{A})]} |f(x) - p(x)|.$$

Plugging both bounds into (14), along with the fact that $\|\mathbf{Q}\| = 1$ and that these statements hold for *any* polynomial with degree $< k$ gives $\|f(\mathbf{A})\mathbf{x} - \mathbf{y}\| \leq 2\delta_k\|\mathbf{x}\|$ after rescaling via (13). ☐

As discussed in the introduction, Theorem 6 can be tightened in certain special cases, including when $\mathbf{A}$ is positive definite and $f(\mathbf{A}) = \mathbf{A}^{-1}$. We defer consideration of this point to Section 7.

5 Finite precision preliminaries

Our goal is to understand how Theorem 6 and related bounds translate from exact arithmetic to finite precision. In particular, our results apply to machines that employ floating-point arithmetic. We use ϵ_{mach} to denote the relative precision of the floating-point system. An algorithm is generally considered "stable" if it runs accurately when $1/\epsilon_{\text{mach}}$ is bounded by some polynomial in the input parameters, i.e., when the number of bits required is logarithmic in these parameters.

We say a machine has precision ϵ_{mach} if it can perform computations to relative error ϵ_{mach}, which necessarily requires that it can represent numbers to relative precision ϵ_{mach} – i.e., it has $\geq \log_2(1/\epsilon_{\text{mach}})$ bits in its floating point significand. To be precise, we require:

Requirement 1 (Accuracy of floating-point arithmetic)**.** *Let $\circ$ denote any of the four basic arithmetic operations ($+, -, \times, \div$) and let $\text{fl}(x \circ y)$ denote the result of computing $x \circ y$. Then a machine with precision ϵ_{mach} must be able to compute:*

$$\text{fl}(x \circ y) = (1 + \delta)(x \circ y) \qquad \textit{where} \qquad |\delta| \leq \epsilon_{\text{mach}}$$

and

$$\text{fl}(\sqrt{x}) = (1 + \delta)\sqrt{x} \qquad \textit{where} \qquad |\delta| \leq \epsilon_{\text{mach}}.$$

Requirement 1 is satisfied by any computer implementing the IEEE 754 standard for floating-point arithmetic [IEE08] with $\geq \log_2(1/\epsilon_{\text{mach}})$ bits of precision, as long as operations do not overflow or underflow[4]. Underflow or overflow occur when $(1+\delta)(x \circ y)$ cannot be represented in finite precision for any δ with $|\delta| \leq \epsilon_{\text{mach}}$, either because $x \circ y$ is so large that it exceeds the maximum expressible number on the computer or because it is so small that expressing the number to relative precision would require a negative exponent that is larger in magnitude than that supported by the computer. As is typical in stability analysis, we will ignore the possibility of overflow and underflow because doing so significantly simplifies the presentation of our results [Hig02].

However, because the version of Lanczos studied normalizes vectors at each iteration, it is not hard to check that our proofs, and the results of Paige, and Gu and Eisenstat that we rely on, go through with overflow and underflow accounted for. To be more precise, overflow does not occur as long as all numbers in the input (and their squares) are at least a $\text{poly}(k, n, C)$ factor smaller than the maximum expressible number (recall that in Theorem 1, C is an upper bound on $|f(x)|$ over our eigenvalue range). That is, overflow is avoided if we assume the exponent in our floating-point system has $\Omega(\log\log(kn \cdot \max(C, 1)))$ bits overall and $\Omega(1)$ bits more than what is needed to express the input. This ensures, for example, that the computation of $\|\mathbf{x}\|$ does not overflow and that the multiplication $\mathbf{A}\mathbf{w}$ does not overflow for any unit norm $\mathbf{w}$.

To account of underflow, Requirement 1 can be modified by including additive error γ_{mach} for $\times$ and $\div$ operations, where γ_{mach} denotes the smallest expressible positive number on our floating-point machine. The additive error carries through all calculations, but will be swamped by multiplicative error as long as we assume that $\|\mathbf{A}\|$, $\|\mathbf{x}\|$, ϵ_{mach}, and our function upper bound C are larger than γ_{mach} by a $\text{poly}(k, n, 1/\epsilon_{\text{mach}})$ factor. This ensures, e.g., that $\mathbf{x}$ can be normalized stably and, as we will discuss, allows for accurate multiplication of the input matrix $\mathbf{A}$ any vector.

In addition to Requirement 1, we also require the following of matrix-vector multiplications involving our input matrix $\mathbf{A}$:

Requirement 2 (Accuracy of matrix multiplication)**.** *Let* $\text{fl}(\mathbf{A}\mathbf{w})$ *denote the result of computing* $\mathbf{A}\mathbf{w}$ *on our floating-point computer. Then a computer with precision* ϵ_{mach} *must be able to compute, for any* $\mathbf{w} \in \mathbb{R}^n$*,*

$$\|\,\text{fl}(\mathbf{A}\mathbf{w}) - \mathbf{A}\mathbf{w}\| \leq 2n^{3/2}\|\mathbf{A}\|\|\mathbf{w}\|\,\epsilon_{\text{mach}}\,.$$

If $\mathbf{A}\mathbf{w}$ is computed explicitly, as long as $n\,\epsilon_{\text{mach}} \leq \frac{1}{2}$ (which holds for all of our results), any computer satisfying Requirement 1 also satisfies Requirement 2 [Wil65, Hig02]. We list Requirement 2 separately to allow our analysis to apply in situations where $\mathbf{A}\mathbf{w}$ is computed approximately for reasons other than rounding error. For example, in many applications where $\mathbf{A}$ cannot be accessed explicitly, $\mathbf{A}\mathbf{w}$ is approximated with an iterative method [FMMS16, OSV12]. As long as this computation is performed to the precision specified in Requirement 2, then our analysis holds.

As mentioned, when $\mathbf{A}\mathbf{w}$ is computed explicitly, underflow could occur during intermediate steps on a finite precision computer. This will add an error term of $2n^{3/2}\gamma_{\text{mach}}$ to $\|\,\text{fl}(\mathbf{A}\mathbf{w}) - \mathbf{A}\mathbf{w}\|$. However, under our assumption that $\epsilon_{\text{mach}}\,\|\mathbf{A}\| \gg \gamma_{\text{mach}}$, this term is subsumed by the $2n^{3/2}\|\mathbf{A}\|\|\mathbf{w}\|\,\epsilon_{\text{mach}}$ term whenever $\|\mathbf{w}\|$ is not tiny (in Algorithm 1, $\|\mathbf{w}\|$ is always very close to 1).

Finally, we mention that, in our proofs, we typically show that operations incur error $\epsilon_{\text{mach}} \cdot F$ for some value F that depends on problem parameters. Ultimately, to obtain error $0 < \epsilon \leq 1$ we

[4]Underflow is only a concern for $\times$ and $\div$ operations. On any computer implementing gradual underflow and a guard bit, Requirement 1 always holds for $+$ and $-$, even when underflow occurs. $\sqrt{x}$ cannot underflow or overflow.

then require that $\epsilon_{\text{mach}} \leq \epsilon/F$. Accordingly, during the course of a proof we will often assume that $\epsilon \cdot F \leq 1$. Additionally, all runtime bounds are for the unit-cost RAM model: we assume that computing $\text{fl}(x \circ y)$ and $\text{fl}(\sqrt{x})$ require $O(1)$ time. For simplicity, we also assume that the scalar function f we are interested in applying to $\mathbf{A}$ can be computed to relative error ϵ_{mach} in $O(1)$ time.

6 Lanczos in finite precision

The most notable issue with the Lanczos algorithm in finite precision is that $\mathbf{Q}$'s column vectors lose the mutual orthogonality property of (6). In practice, this loss of orthogonality is quite severe: $\mathbf{Q}$ will often have numerical rank $\ll k$. Naturally, $\mathbf{Q}$'s column vectors will thus also fail to span the Krylov subspace $\mathcal{K}_k = [\mathbf{q}_1, \mathbf{A}\mathbf{q}_1, \ldots, \mathbf{A}^{k-1}\mathbf{q}_1]$, and so we do not expect to be able to accurately apply all degree $< k$ polynomials. Surprisingly, this does not turn out to be much of a problem!

6.1 Starting point: Paige's results

In particular, a seminal result of Paige shows that while (6) falls apart under finite precision calculations, (5) of Claim 5 still holds, up to small error. In particular, in [Pai76] he proves that:

Theorem 8 (Lanczos Output in Finite Precision, [Pai76])**.** *Run for k iterations on a computer satisfying Requirements 1 and 2 with relative precision ϵ_{mach}, the Lanczos algorithm (Algorithm 1) computes $\mathbf{Q} \in \mathbb{R}^{n\times k}$, an additional column vector $\mathbf{q}_{k+1} \in \mathbb{R}^n$, a scalar β_k, and a symmetric tridiagonal matrix $\mathbf{T} \in \mathbb{R}^{k\times k}$ such that:*

$$\mathbf{A}\mathbf{Q} = \mathbf{Q}\mathbf{T} + \beta_{k+1}\mathbf{q}_{k+1}\mathbf{e}_k^T + \mathbf{E}, \tag{15}$$

and

$$\|\mathbf{E}\| \leq k(2n^{3/2} + 7)\|\mathbf{A}\|\epsilon_{mach}, \tag{16}$$

$$|\|\mathbf{q}_i\| - 1| \leq (n+4)\,\epsilon_{\text{mach}} \quad \text{for all } i. \tag{17}$$

In [Pai80] (see equation 3.28), it is shown that together, the above bounds also imply:

$$\lambda_{\min}(\mathbf{A}) - \epsilon_1 \leq \lambda(\mathbf{T}) \leq \lambda_{\max}(\mathbf{A}) + \epsilon_1 \tag{18}$$

where $\epsilon_1 = k^{5/2}\|\mathbf{A}\|\left(68 + 17n^{3/2}\right)\epsilon_{mach}$.

Paige was interested in using Theorem 8 to understand how $\mathbf{T}$ and $\mathbf{Q}$ can be used to compute approximate eigenvectors and values for $\mathbf{A}$. His bounds are quite strong: for example, (18) shows that (7) still holds up to tiny additive error, even though establishing that result for exact arithmetic relied heavily on the orthogonality of $\mathbf{Q}$'s columns.

6.2 Finite precision lanczos for applying polynomials

Theorem 8 allows us to give a finite precision analog of Lemma 7 for polynomials with magnitude $|p(x)|$ bounded on a small extension of the eigenvalue range $[\lambda_{\min}(\mathbf{A}), \lambda_{\max}(\mathbf{A})]$.

Lemma 9 (Lanczos Applies Bounded Polynomials)**.** *Suppose* $\mathbf{Q} \in \mathbb{R}^{n\times k}$ *and* $\mathbf{T} \in \mathbb{R}^{k\times k}$ *are computed by the Lanczos algorithm on a computer satisfying Requirements 1 and 2 with relative precision* ϵ_{mach}*, and thus these matrices satisfy the bounds of Theorem 8. For any* $\eta \geq 85n^{3/2}k^{5/2}\|\mathbf{A}\|\,\epsilon_{\text{mach}}$*, if* p *is a polynomial with degree* $< k$ *and* $|p(x)| \leq C$ *for all* $x \in [\lambda_{\min}(\mathbf{A}) - \eta, \lambda_{\max}(\mathbf{A}) + \eta]$ *then:*

$$\|\mathbf{Q}p(\mathbf{T})\mathbf{e}_1 - p(\mathbf{A})\mathbf{q}_1\| \leq \frac{2Ck^3\|\mathbf{E}\|}{\eta} \tag{19}$$

where $\mathbf{E}$ *is the error matrix defined in Theorem 8.*

Finite precision Lanczos applies Chebyshev polynomials

It is not immediately clear how to modify the proof of Lemma 7 to handle the error $\mathbf{E}$ in (15). Intuitively, any bounded polynomial cannot have too large a derivative by the Markov brothers' inequality [Mar90], and so we expect $\mathbf{E}$ to have a limited effect. However, we are not aware of a way to make this reasoning formal for matrix polynomials and asymmetric error $\mathbf{E}$.

As illustrated in [OSV12], there is a natural way to prove (19) for the monomials $\mathbf{A}, \mathbf{A}^2, \ldots, \mathbf{A}^{k-1}$. The bound can then be extended to all polynomials via triangle inequality, but error is amplified by the coefficients of each monomial component in $p(\mathbf{A})$. Unfortunately, there are polynomials that are uniformly bounded by C (and thus have bounded derivative) even though their monomial components can have coefficients much larger than C. The ultimate effect is that the approach taken in [OSV12] would incur an exponential dependence on k on the right hand side of (19).

To obtain our stronger polynomial dependence, we proceed with a different two-part analysis. We first show that (19) holds for any *Chebyshev polynomial* with degree $< k$ that is appropriately stretched and shifted to the range $[\lambda_{\min}(\mathbf{A}) - \eta, \lambda_{\max}(\mathbf{A}) + \eta]$. Chebyshev polynomials have magnitude much smaller than that of their monomial components, but because they can be formed via a well-behaved recurrence, we can show that they are stable to the perturbation $\mathbf{E}$. We can then obtain the general result of Lemma 9 because *any* bounded polynomial can be written as a weighted sum of such Chebyshev polynomials, with bounded weights.

Let $T_0, T_1, \ldots, T_{k-1}$ be the first k Chebyshev polynomials of the first kind, defined recursively:

$$\begin{aligned}
&T_0(x) = 1,\\
&T_1(x) = x,\\
&T_i(x) = 2xT_{i-1}(x) - T_{i-2}(x).
\end{aligned} \tag{20}$$

The roots of the Chebyshev polynomials lie in $[-1, 1]$ and this is precisely the range where they remain "well behaved": for $|x| > 1$, $T_i(x)$ begins to grow quite quickly. Define

$$\mathrm{r}_{\max} \stackrel{\text{def}}{=} \lambda_{\max} + \eta \qquad \text{and} \qquad \mathrm{r}_{\min} \stackrel{\text{def}}{=} \lambda_{\min} - \eta$$

and

$$\delta = \frac{2}{\mathrm{r}_{\max} - \mathrm{r}_{\min}} \qquad \text{and} \qquad \overline{T}_i(x) = T_i\left(\delta(x - \mathrm{r}_{\min}) - 1\right). \tag{21}$$

$\overline{T}_i(x)$ is the i^{th} Chebyshev polynomial stretched and shifted so that $\overline{T}_i(\mathrm{r}_{\min}) = T_i(-1)$ and $\overline{T}_i(\mathrm{r}_{\max}) = T_i(1)$. We prove the following:

Lemma 10 (Lanczos Applies Chebyshev Polynomials Stably)**.** *Suppose* $\mathbf{Q} \in \mathbb{R}^{n\times k}$ *and* $\mathbf{T} \in \mathbb{R}^{k\times k}$ *are computed by the Lanczos algorithm on a computer satisfying Requirements 1 and 2 with relative precision* ϵ_{mach} *and thus these matrices satisfy the bounds of Theorem 8. For any* $\eta \geq 85n^{3/2}k^{5/2}\|\mathbf{A}\|\,\epsilon_{\text{mach}}$*, for all* $i < k$*,*

$$\|\mathbf{Q}\overline{T}_i(\mathbf{T})\mathbf{e}_1 - \overline{T}_i(\mathbf{A})\mathbf{q}_1\| \leq \frac{2i^2 \cdot \|\mathbf{E}\|}{\eta} \tag{22}$$

where $\mathbf{E}$ *is the error matrix in Theorem 8 and* $\overline{T}_i$ *is the* i^{th} *shifted Chebyshev polynomial of (21).*

Proof. Define $\bar{\mathbf{A}} \stackrel{\text{def}}{=} \delta(\mathbf{A} - \mathrm{r}_{\min}\mathbf{I}) - \mathbf{I}$ and $\bar{\mathbf{T}} \stackrel{\text{def}}{=} \delta(\mathbf{T} - \mathrm{r}_{\min}\mathbf{I}) - \mathbf{I}$. so (22) is equivalent to:

$$\|\mathbf{Q}T_i(\bar{\mathbf{T}})\mathbf{e}_1 - T_i(\bar{\mathbf{A}})\mathbf{q}_1\| \leq \frac{2i^2 \cdot \|\mathbf{E}\|}{\eta}. \tag{23}$$

So now we just focus on showing (23). We use the following notation:

$$\begin{aligned} \mathbf{t}_i &\stackrel{\text{def}}{=} T_i(\bar{\mathbf{A}})\mathbf{q}_1, & \tilde{\mathbf{t}}_i &\stackrel{\text{def}}{=} T_i(\bar{\mathbf{T}})\mathbf{e}_1, \\ \mathbf{d}_i &\stackrel{\text{def}}{=} \mathbf{t}_i - \mathbf{Q}\tilde{\mathbf{t}}_i, & \boldsymbol{\xi}_i &\stackrel{\text{def}}{=} \delta\mathbf{E}\tilde{\mathbf{t}}_{i-1}. \end{aligned}$$

Proving (23) is equivalent to showing $\|\mathbf{d}_i\| \leq \frac{2i^2\|\mathbf{E}\|}{\eta}$. From the Chebyshev recurrence (20) for all $i \geq 2$:

$$\begin{aligned} \mathbf{d}_i &= \left(2\,\bar{\mathbf{A}}\,\mathbf{t}_{i-1} - \mathbf{t}_{i-2}\right) - \mathbf{Q}\left(2\,\bar{\mathbf{T}}\,\tilde{\mathbf{t}}_{i-1} - \tilde{\mathbf{t}}_{i-2}\right) \\ &= 2(\bar{\mathbf{A}}\,\mathbf{t}_{i-1} - \mathbf{Q}\,\bar{\mathbf{T}}\,\tilde{\mathbf{t}}_{i-1}) - \mathbf{d}_{i-2}. \end{aligned}$$

Applying the perturbed Lanczos relation (15), we can write $\mathbf{Q}\,\bar{\mathbf{T}} = \bar{\mathbf{A}}\,\mathbf{Q} - \delta\beta_{k+1}\mathbf{q}_{k+1}\mathbf{e}_k^T - \delta\mathbf{E}$. Plugging this in above we then have:

$$\begin{aligned} \mathbf{d}_i &= 2\,\bar{\mathbf{A}}(\mathbf{t}_{i-1} - \mathbf{Q}\tilde{\mathbf{t}}_{i-1}) - \mathbf{d}_{i-2} + 2\delta\beta_{k+1}\mathbf{q}_{k+1}\mathbf{e}_k^T\tilde{\mathbf{t}}_{i-1} + 2\delta\mathbf{E}\tilde{\mathbf{t}}_{t-1} \\ &= (2\,\bar{\mathbf{A}}\,\mathbf{d}_{i-1} - \mathbf{d}_{i-2}) + 2\delta\beta_{k+1}\mathbf{q}_{k+1}\mathbf{e}_k^T\tilde{\mathbf{t}}_{i-1} + 2\boldsymbol{\xi}_i. \end{aligned}$$

Finally, we use as in Lemma 7, that $\mathbf{e}_k^T\tilde{\mathbf{t}}_{i-1} = \mathbf{e}_k^T T_{i-1}(\bar{\mathbf{T}})\mathbf{e}_1 = 0$ since $\bar{\mathbf{T}}$ (like $\mathbf{T}$) is tridiagonal. Thus, $\mathbf{T}^{q-1}\mathbf{e}_1$ is zero outside its first q entries and so for $i < k$, $T_{i-1}(\bar{\mathbf{T}})\mathbf{e}_1$ is zero outside of its first $k-1$ entries. This gives the error recurrence:

$$\mathbf{d}_i = (2\,\bar{\mathbf{A}}\,\mathbf{d}_{i-1} - \mathbf{d}_{i-2}) + 2\boldsymbol{\xi}_i. \tag{24}$$

As in standard stability arguments for the *scalar* Chebyshev recurrence, we can analyze (24) using Chebyshev polynomials of the second kind [Cle55]. The i^{th} Chebyshev polynomial of the second kind is denoted $U_i(x)$ and defined by the recurrence

$$\begin{aligned} &U_0(x) = 1, \\ &U_1(x) = 2x, \\ &U_i(x) = 2xT_{i-1}(x) - T_{i-2}(x). \end{aligned} \tag{25}$$

We claim that for any $i \geq 0$, defining $U_k(x) = 0$ for any $k < 0$ for convinience:

$$\mathbf{d}_i = U_{i-1}(\bar{\mathbf{A}})\boldsymbol{\xi}_1 + 2\sum_{j=2}^{i} U_{i-j}(\bar{\mathbf{A}})\boldsymbol{\xi}_j. \tag{26}$$

This follows by induction starting with the base cases:

$$\mathbf{d}_0 = \mathbf{0}, \text{ and } \mathbf{d}_1 = \boldsymbol{\xi}_1.$$

Using (24) and assuming by induction that (26) holds for all $j < i$,

$$\begin{aligned}
\mathbf{d}_i &= 2\boldsymbol{\xi}_i + \left(2\,\bar{\mathbf{A}}\,\mathbf{d}_{i-1} - \mathbf{d}_{i-2}\right) \\
&= 2\boldsymbol{\xi}_i + [2\bar{A}U_{i-2}(\bar{\mathbf{A}})\boldsymbol{\xi}_1 - U_{i-3}(\bar{\mathbf{A}})\boldsymbol{\xi}_1] + 4\,\bar{\mathbf{A}}\sum_{j=2}^{i-1} U_{i-1-j}(\bar{\mathbf{A}})\boldsymbol{\xi}_j - 2\sum_{j=2}^{i-2} U_{i-2-j}(\bar{\mathbf{A}})\boldsymbol{\xi}_j \\
&= 2\boldsymbol{\xi}_i + U_{i-1}(\bar{\mathbf{A}})\boldsymbol{\xi}_1 + \left(2\sum_{j=2}^{i-2} 2\,\bar{\mathbf{A}}\,U_{i-1-j}(\bar{\mathbf{A}})\boldsymbol{\xi}_j - U_{i-2-j}(\bar{\mathbf{A}})\boldsymbol{\xi}_j\right) + 4\,\bar{\mathbf{A}}\,U_0(\bar{\mathbf{A}})\boldsymbol{\xi}_{i-1} \\
&= U_{i-1}(\bar{\mathbf{A}})\boldsymbol{\xi}_1 + 2\sum_{j=2}^{i} U_{i-j}(\bar{\mathbf{A}})\boldsymbol{\xi}_j,
\end{aligned}$$

establishing (26). It follows from triangle inequality and submultiplicativity that

$$\|\mathbf{d}_i\| \leq 2\sum_{j=1}^{i} \|U_{i-j}(\bar{\mathbf{A}})\|\|\boldsymbol{\xi}_j\|.$$

Since $\bar{\mathbf{A}}$ is symmetric (it is just a shifted and scaled $\mathbf{A}$), $U_k(\bar{\mathbf{A}})$ is equivalent to the matrix obtained by applying $U_k(x)$ to each of $\bar{\mathbf{A}}$'s eigenvalues, which lie in the range $[-1, 1]$. It is well known that, for values in this range $U_k(x) \leq k+1$ [GST07]. Accordingly, $\|U_{i-j}(\bar{\mathbf{A}})\| \leq i - j + 1$, so

$$\|\mathbf{d}_i\| \leq 2\sum_{j=1}^{i}(i - j + 1)\|\boldsymbol{\xi}_j\| \leq 2\sum_{j=1}^{i} i\|\boldsymbol{\xi}_j\|. \tag{27}$$

We finally bound $\|\boldsymbol{\xi}_j\| = \delta\mathbf{E}\tilde{\mathbf{t}}_{j-1}$. Recall that $\tilde{\mathbf{t}}_{j-1} \stackrel{\text{def}}{=} T_{j-1}(\bar{\mathbf{T}})\mathbf{e}_1$ so:

$$\begin{aligned}
\|\boldsymbol{\xi}_j\| &= \|\delta\mathbf{E}T_{j-1}(\bar{\mathbf{T}})\mathbf{e}_1\| \\
&\leq \delta\,\|\mathbf{E}\|\,\left|T_{j-1}(\bar{\mathbf{T}})\right\| = \frac{2}{\mathrm{r}_{\max} - \mathrm{r}_{\min}} \cdot \|\mathbf{E}\|\,\|T_{j-1}\left(\delta\left(\mathbf{T} - \mathrm{r}_{\min}\,\mathbf{I}\right) - \mathbf{I}\right)\|\,,
\end{aligned} \tag{28}$$

where we used that $\|\mathbf{e}_1\| = 1$, and $\delta = \frac{2}{\mathrm{r}_{\max} - \mathrm{r}_{\min}}$. By (18) of Theorem 8 and our requirement that $\eta \geq 85n^{3/2}k^{5/2}\|\mathbf{A}\|\,\epsilon_{\text{mach}}$, $\mathbf{T}$ has all eigenvalues in $[\lambda_{\min} - \eta, \lambda_{\max} + \eta]$. Thus $\bar{\mathbf{T}} = \delta\left(\mathbf{T} - \mathrm{r}_{\min}\,\mathbf{I}\right) - \mathbf{I}$ has all eigenvalues in $[-1, 1]$. We have $T_{j-1}(x) \leq 1$ for $x \in [-1, 1]$, giving $\|T_{j-1}\left(\delta\left(\mathbf{T} - \mathrm{r}_{\min}\,\mathbf{I}\right) - \mathbf{I}\right)\| \leq 1$.

Plugging this back into (28), $\|\boldsymbol{\xi}_j\| \leq \frac{2\|\mathbf{E}\|}{\mathrm{r}_{\max} - \mathrm{r}_{\min}}$ and plugging into (27), $\|\mathbf{d}_i\| \leq \frac{4i^2\|\mathbf{E}\|}{\mathrm{r}_{\max} - \mathrm{r}_{\min}}$. This gives the lemma after noting that $\mathrm{r}_{\max} - \mathrm{r}_{\min} \geq 2\eta$. □

From Chebyshev polynomials to general polynomials

As discussed, with Lemma 10 in place, we can prove Lemma 9 by writing any bounded polynomial in the Chebyshev basis.

Proof of Lemma 9. Recall that we define $\mathrm{r}_{\min} = \lambda_{\min} - \eta$, $\mathrm{r}_{\max} = \lambda_{\max} + \eta$, and $\delta = \frac{2}{\mathrm{r}_{\max} - \mathrm{r}_{\min}}$. Let

$$\overline{p}(x) = p\left(\frac{x+1}{\delta} + \mathrm{r}_{\min}\right).$$

For any $x \in [-1,1]$, $\overline{p}(x) = p(y)$ for some $y \in [\mathrm{r}_{\min}, \mathrm{r}_{\max}]$. This immediately gives $|\overline{p}(x)| \leq C$ on $[-1,1]$ by the assumption that $|p(x)| \leq C$ on $[\lambda_{\min} - \eta, \lambda_{\max} + \eta] = [\mathrm{r}_{\min}, \mathrm{r}_{\max}]$.

Any polynomial with degree $< k$ can be written as a weighted sum of the first k Chebyshev polynomials (see e.g. [GST07]). Specifically we have:

$$\overline{p}(x) = c_0 T_0(x) + c_1 T_1(x) + \ldots + c_{k-1} T_{k-1}(x),$$

where the i^{th} coefficient is given by:

$$c_i = \frac{2}{\pi} \int_{-1}^{1} \frac{\overline{p}(x) T_i(x)}{\sqrt{1-x^2}}.$$

$|T_i(x)| \leq 1$ on $[-1,1]$ and $\int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} = \pi$, and since $|\overline{p}(x)| \leq C$ for $x \in [-1,1]$ we have for all i:

$$|c_i| \leq 2C. \tag{29}$$

By definition, $p(x) = \overline{p}\left(\delta(x - \mathrm{r}_{\min}) - 1\right)$. Letting $\bar{\mathbf{A}} = \delta(\mathbf{A} - \mathrm{r}_{\min}\mathbf{I}) - \mathbf{I}$ and $\bar{\mathbf{T}} = \delta(\mathbf{T} - \mathrm{r}_{\min}\mathbf{I}) - \mathbf{I}$ as in the proof of Lemma 10, we have

$$\|\mathbf{Q}p(\mathbf{T})\mathbf{e}_1 - p(\mathbf{A})\mathbf{q}_1\| = \left\|\mathbf{Q}\overline{p}\left(\bar{\mathbf{T}}\right)\mathbf{e}_1 - \overline{p}\left(\bar{\mathbf{A}}\right)\mathbf{q}_1\right\|$$

so need to upper bound the right hand side to prove the lemma. Applying triangle inequality:

$$\left\|\mathbf{Q}\overline{p}\left(\bar{\mathbf{T}}\right)\mathbf{e}_1 - \overline{p}(\bar{\mathbf{A}})\mathbf{q}_1\right\| \leq \sum_{i=0}^{k-1} c_i \left\|\mathbf{Q}T_i\left(\bar{\mathbf{T}}\right)\mathbf{e}_1 - T_i(\bar{\mathbf{A}})\mathbf{q}_1\right\|,$$

where for each i, $|c_i| \leq 2C$ by (29). Combining with Lemma 10, we thus have:

$$\left\|\mathbf{Q}\overline{p}(\bar{\mathbf{T}})\mathbf{e}_1 - \overline{p}(\bar{\mathbf{A}})\mathbf{q}_1\right\| \leq \frac{4C \cdot \|\mathbf{E}\|}{\eta} \sum_{i=0}^{k-1} i^2 \leq \frac{2Ck^3\|\mathbf{E}\|}{\eta}$$

which gives the lemma. ☐

6.3 Completing the analysis

With Lemma 9, we have nearly proven our main result, Theorem 1. We first show, using a proof mirroring our analysis in the exact arithmetic case, that Lemma 9 implies that $\mathbf{Q}f(\mathbf{T})\mathbf{e}_1$ well approximates $f(\mathbf{A})\mathbf{q}_1$. Thus the output $\mathbf{y} = \|\mathbf{x}\|\mathbf{Q}f(\mathbf{T})\mathbf{e}_1$ well approximates $f(\mathbf{A})\mathbf{x}$. With this bound, all that remains in proving Theorem 1 is to show that we can compute $\mathbf{y}$ accurately using known techniques (although with a tedious error analysis).

Lemma 11 (Stable function approximation via Lanczos)**.** *Suppose* $\mathbf{Q} \in \mathbb{R}^{n \times k}$ *and* $\mathbf{T} \in \mathbb{R}^{k \times k}$ *are computed by the Lanczos algorithm on a computer satisfying Requirements 1 and 2 with relative precision* ϵ_{mach} *and thus these matrices satisfy the bounds of Theorem 8. For degree* k *and any* η *with* $85n^{3/2}k^{5/2}\|\mathbf{A}\|\epsilon_{mach} \leq \eta \leq \|\mathbf{A}\|$ *define:*

$$\delta_k = \min_{\substack{\text{polynomial } p \\ \text{with degree} < k}} \left(\max_{x \in [\lambda_{\min}(\mathbf{A})-\eta, \lambda_{\max}(\mathbf{A})+\eta]} |f(x) - p(x)| \right) \tag{30}$$

and $C = \max_{x \in [\lambda_{\min}(\mathbf{A})-\eta, \lambda_{\max}(\mathbf{A})+\eta]} |f(x)|$*. Then we have:*

$$\|f(\mathbf{A})\mathbf{q}_1 - \mathbf{Q}f(\mathbf{T})\mathbf{e}_1\| \leq (k+2)\delta_k + \epsilon_{\mathrm{mach}} \cdot \frac{41Ck^4n^{3/2}\|\mathbf{A}\|}{\eta}. \tag{31}$$

Proof. Applying Lemma 9, letting p be the optimal degree $< k$ polynomial achieving δ_k, by (30) and our bound on $f(x)$ on this range:

$$\|\mathbf{Q}p(\mathbf{T})\mathbf{e}_1 - p(\mathbf{A})\mathbf{q}_1\| \leq \frac{2k^3(C+\delta_k)\|\mathbf{E}\|}{\eta}.$$

By triangle inequality, spectral norm submultiplicativity, and the fact that $\|\mathbf{q}_1\| \approx 1$ (certainly $\|\mathbf{q}_1\| \leq 2$ even if $\mathbf{x}$ is normalized in finite-precision) we have:

$$\begin{aligned} \|\mathbf{Q}f(\mathbf{T})\mathbf{e}_1 - f(\mathbf{A})\mathbf{q}_1\| &\leq \|\mathbf{Q}p(\mathbf{T})\mathbf{e}_1 - p(\mathbf{A})\mathbf{q}_1\| + \|\mathbf{Q}f(\mathbf{T})\mathbf{e}_1 - \mathbf{Q}p(\mathbf{T})\mathbf{e}_1\| + \|f(\mathbf{A})\mathbf{q}_1 - p(\mathbf{A})\mathbf{q}_1\| \\ &\leq 2k^3(C+\delta_k)\|\mathbf{E}\|/\eta + \|\mathbf{Q}\|\|f(\mathbf{T})\mathbf{e}_1 - p(\mathbf{T})\mathbf{e}_1\| + \|f(\mathbf{A})\mathbf{q}_1 - p(\mathbf{A})\mathbf{q}_1\| \\ &\leq 2k^3(C+\delta_k)\|\mathbf{E}\|/\eta + (\|\mathbf{Q}\|+2)\delta_k \end{aligned} \tag{32}$$

where the last inequality follows from the definition of δ_k in (30) and the fact that all eigenvalues of $\mathbf{T}$ lie in $[\lambda_{\min}(\mathbf{A}) - \eta, \lambda_{\max}(\mathbf{A}) + \eta]$ by (18) of Theorem 8 since $\eta > 85n^{3/2}k^{5/2}\|\mathbf{A}\|\,\epsilon_{\mathrm{mach}}$. By Theorem 8 we also have $\|\mathbf{q}_i\| \leq 1+(n+4)\,\epsilon_{\mathrm{mach}}$ for all i. This gives $\|\mathbf{Q}\| \leq \|\mathbf{Q}\|_F \leq k+k(n+4)\,\epsilon_{\mathrm{mach}}$. Further, $\|\mathbf{E}\| \leq k(2n^{3/2}+7)\|\mathbf{A}\|\,\epsilon_{\mathrm{mach}}$. Plugging back into (32), loosely bounding $\delta_k < C$ (since we could always set $p(x) = 0$), and using that $\eta \leq \|\mathbf{A}\|$, gives (31) and thus completes the lemma. □

After scaling by a $\|\mathbf{x}\|$ factor, Lemma 11 shows that the output $\mathbf{y} = \|\mathbf{x}\|\mathbf{Q}f(\mathbf{T})\mathbf{e}_1$ of Lanczos approximates $f(\mathbf{A})\mathbf{x}$ to within a $(k+2)\delta_k\|\mathbf{x}\|$ factor (plus a lower order term depending on ϵ_{mach}), where δ_k is the best approximation given by a degree $< k$ polynomial on the eigenvalue range. Of course, in finite precision, we cannot exactly compute $\mathbf{y}$. However, it is known that it is possible to stably compute an eigendecomposition of a symmetric tridiagonal $\mathbf{T}$ in $\tilde{O}(n^2)$ time. This allows us to explicitly approximate $f(\mathbf{T})$ and thus $\mathbf{y}$. The upshot is our main theorem:

Theorem 1 (Function Approximation via Lanczos in Finite Arithmetic)**.** *Given real symmetric* $\mathbf{A} \in \mathbb{R}^{n \times n}$*,* $\mathbf{x} \in \mathbb{R}^n$*,* $\eta \leq \|\mathbf{A}\|$*,* $\epsilon \leq 1$*, and any function* f *with* $|f(x)| < C$ *for* $x \in [\lambda_{\min}(\mathbf{A}) - \eta, \lambda_{\max}(\mathbf{A}) + \eta]$*, let* $B = \log\left(\frac{nk\|\mathbf{A}\|}{\epsilon\eta}\right)$*. Suppose Algorithm 1 is run for* k *iterations on a computer satisfying Requirements 1 and 2 with relative precision* $\epsilon_{\mathrm{mach}} = 2^{-\Omega(B)}$ *(e.g. on computer using* $\Omega(B)$ *bits of precision). If in Step 13,* $\mathbf{y}$ *is computed using the eigendecomposition algorithm of [GE95], it satisfies:*

$$\|f(\mathbf{A})\mathbf{x} - \mathbf{y}\| \leq (7k \cdot \delta_k + \epsilon C)\|\mathbf{x}\| \tag{33}$$

where

$$\delta_k \stackrel{\text{def}}{=} \min_{\substack{\text{polynomial } p \\ \text{with degree } < k}} \left(\max_{x \in [\lambda_{\min}(\mathbf{A})-\eta, \lambda_{\max}(\mathbf{A})+\eta]} |p(x) - f(x)| \right).$$

The algorithm's runtime is $O(\text{mv}(\mathbf{A})k + k^2 B + kB^2)$*, where* $\text{mv}(\mathbf{A})$ *is the time required to multiply* $\mathbf{A}$ *by a vector to the precision required by Requirement 2 (e.g.* $O(\text{nnz}(\mathbf{A})$ *time if* $\mathbf{A}$ *is given explicitly).*

We note that the dependence on η in our bound is typically mild. For example, it $\mathbf{A}$ is positive semi-definite, if it is possible to find a good polynomial approximation on $[\lambda_{\min}(\mathbf{A}), \lambda_{\max}(\mathbf{A})]$, it is possible to find an approximation with similar degree on, e.g., $[\frac{1}{2}\lambda_{\min}(\mathbf{A}), 2\lambda_{\max}(\mathbf{A})]$, in which case $\eta = \Theta(\lambda_{\min}(\mathbf{A})|)$. For some functions, we can get away with an even larger η (and thus fewer required bits). For example, in Section 8 our applications to the matrix step function and matrix exponential both set $\eta = \lambda_{\max}(\mathbf{A})$.

Proof. We can apply Lemma 11 to show that:

$$\|f(\mathbf{A})\mathbf{q}_1 - \mathbf{Q}f(\mathbf{T})\mathbf{e}_1\| \leq (k+2)\delta_k + \epsilon_{\text{mach}} \cdot \frac{41Ck^4 n^{3/2}\|\mathbf{A}\|}{\eta}. \tag{34}$$

The lemma requires $\epsilon_{\text{mach}} \leq \frac{\eta}{85n^{3/2}k^{5/2}\|\mathbf{A}\|}$, which holds since we require $\epsilon \leq 1$ and set $\epsilon_{\text{mach}} = 2^{-\Omega(B)}$ with $B = \log\left(\frac{nk\|\mathbf{A}\|}{\epsilon\eta}\right)$. This also ensures that the second term of (34) becomes very small, and so we can bound:

$$\|f(\mathbf{A})\mathbf{q}_1 - \mathbf{Q}f(\mathbf{T})\mathbf{e}_1\| \leq (k+2)\delta_k + \epsilon C/4. \tag{35}$$

We now show that a similar bound still holds when we compute $\mathbf{Q}f(\mathbf{T})\mathbf{e}_1$ approximately. Via an error analysis of the symmetric tridiagonal eigendecomposition algorithm of Gu and Eisenstat [GE95], contained in Lemma 23 of Appendix A, for any $\epsilon_1 \leq 1/2$ with

$$ck^3 \log k \cdot \epsilon_{\text{mach}} \leq \epsilon_1 \leq \frac{\eta}{4\|\mathbf{T}\|} \tag{36}$$

for large enough c, in $O(k^2 \log \frac{k}{\epsilon_1} + k \log^2 \frac{k}{\epsilon_1})$ time we can compute $\mathbf{z}$ satisfying:

$$\|f(\mathbf{T})\mathbf{e}_1 - \mathbf{z}\| \leq 2\delta_k + \epsilon_1 \cdot \left(\frac{8k^3 C\|\mathbf{T}\|}{\eta} + 16C\right). \tag{37}$$

By our restriction that $\epsilon \leq 1$ and $\eta \leq \|\mathbf{A}\|$, since $B = \log\left(\frac{nk\|\mathbf{A}\|}{\epsilon\eta}\right)$, we have $\epsilon_{\text{mach}} = 2^{-\Omega(B)} \leq \frac{1}{(nk)^c}$ for some large constant c. This gives $\|\mathbf{T}\| \leq \|\mathbf{A}\| + \epsilon_{\text{mach}} k^{5/2}\|\mathbf{A}\|(68 + 17n^{3/2}) \leq 2\|\mathbf{A}\|$ by (18) of Theorem 8. Thus, if we set $\epsilon_1 = \left(\frac{\epsilon\eta}{3nk\|\mathbf{A}\|}\right)^c$ for large enough c, by (37) we will have:

$$\|f(\mathbf{T})\mathbf{e}_1 - \mathbf{z}\| \leq 2\delta_k + \frac{\epsilon C}{4(k+1)}. \tag{38}$$

Furthermore $\|\mathbf{Q}\| \leq \|\mathbf{Q}\|_F \leq k + k(n+4)\,\epsilon_{\text{mach}} \leq k+1$ by Paige's bounds (Theorem 8) and the fact that $\epsilon_{\text{mach}} \leq \left(\frac{1}{nk}\right)^c$ for some large c. Using (38), this gives:

$$\|\mathbf{Q}f(\mathbf{T})\mathbf{e}_1 - \mathbf{Q}\mathbf{z}\| \leq (2k+2)\delta_k + \epsilon C/4. \tag{39}$$

As discussed in Section 5, if $\mathbf{Qz}$ is computed on a computer satisfying Requirement 1 then the output $\bar{\mathbf{y}}$ satisfies:

$$\|\mathbf{Qz} - \bar{\mathbf{y}}\| \leq 2\max(n,k)^{3/2}\, \epsilon_{\text{mach}}\, \|\mathbf{Q}\|\|\mathbf{z}\|.$$

By (38), $\|\mathbf{z}\| \leq \|f(\mathbf{T})\| + 2\delta_k + \frac{\epsilon C}{4(k+1)} = O(C + \delta_k) = O(C)$ since $\delta_k \leq C$. Accordingly, by our choice of ϵ_{mach} we can bound $\|\mathbf{Qz} - \bar{\mathbf{y}}\| \leq \epsilon C/4$. Combining with (35) and (39) we have:

$$\|f(\mathbf{A})\mathbf{q}_1 - \bar{\mathbf{y}}\| \leq (3k+4)\delta_k + 3\epsilon C/4. \tag{40}$$

This gives the final error bound of (33) after rescaling by a $\|\mathbf{x}\|$ factor. $\|\bar{\mathbf{y}}\| \leq \|f(\mathbf{A})\mathbf{q}_1\| + (3k+4)\delta_k + 3\epsilon C/4 = O(kC)$ and so, by our setting of ϵ_{mach}, we can compute $\mathbf{y} = \|\mathbf{x}\| \cdot \bar{\mathbf{y}}$ up to additive error $\frac{\epsilon C \cdot \|\mathbf{x}\|}{8}$. Similarly, we have $\|f(\mathbf{A})\mathbf{q}_1\|\mathbf{x}\| - f(\mathbf{A})\mathbf{x}\| = \frac{\epsilon C \cdot \|\mathbf{x}\|}{8}$ even when $\mathbf{q}_1 = \mathbf{x}/\|\mathbf{x}\|$ is computed approximately. Overall this lets us claim using (40):

$$\|f(\mathbf{A})\mathbf{x} - \mathbf{y}\| \leq [(3k+4)\delta_k + \epsilon C] \cdot \|\mathbf{x}\|$$

which gives our final error bound. The runtime follows from noting that each iteration of Lanczos requires $\text{mv}(\mathbf{A}) + O(n) = O(\text{mv}(\mathbf{A}))$ time. The stable eigendecomposition of $\mathbf{T}$ up to error ϵ_1 requires $O(k^2 \log \frac{k}{\epsilon_1} + k \log^2 \frac{k}{\epsilon_1}) = O(k^2 B + kB^2)$ time by our setting of ϵ_1. With this eigendecomposition in hand, computing $\mathbf{Q}f(\mathbf{T})\mathbf{e}_1$ takes an additional $O(nk) = O(\text{mv}(\mathbf{A})k)$ time. $\square$

7 Lower bound

In the previous section, we proved that finite precision Lanczos essentially matches the best known exact arithmetic iteration bounds for *general matrix functions*. These bounds depend on the degree needed to uniformly approximate of $f(x)$ over $[\lambda_{\min}(\mathbf{A}), \lambda_{\max}(\mathbf{A})]$. We now turn to the special case of positive definite linear systems, where tighter bounds can be shown.

Specifically, equation (3), proven in Theorem 24, shows that the error of Lanczos after k iterations matches the error of the best polynomial approximating $1/x$ at each of $\mathbf{A}$'s eigenvalues, rather than on the full range $[\lambda_{\min}(\mathbf{A}), \lambda_{\max}(\mathbf{A})]$. Greenbaum proved a natural extension of this bound to the finite precision CG method, showing that its performance matches the best polynomial approximating $1/x$ on tiny ranges around each of $\mathbf{A}$'s eigenvalues [Gre89]. Recall that "tiny" means essentially on the order of machine precision – the approximation need only be over ranges of width η as long as the bits of precision used is $\gtrsim \log(1/\eta)$. We state a simplified version of this result as Theorem 2 and provide a full discussion in Appendix B.

At first glance, Theorem 2 appears to be a very strong result – intuitively, approximating $1/x$ on small intervals around each eigenvalue seems much easier than uniform approximation.

7.1 Main theorem

Surprisingly, we show that this is not the case: Greenbaum's result can be much weaker than the exact arithmetic bounds of Theorem 24. We prove that for any κ and interval width η, there are matrices with condition number κ and just $O(\log \kappa \cdot \log 1/\eta)$ eigenvalues for which any 'stable approximating polynomial' of the form required by Theorem 2 achieving error $\bar{\delta}_k \stackrel{\text{def}}{=} \leq 1/6$ must have degree $\Omega(\kappa^c)$ for a fixed constant $c \geq 1/5$.

This result immediately implies a number of iteration lower bounds on Greenbaum's result, even when we just ask for constant factor approximation to $\mathbf{A}^{-1}\mathbf{x}$. See Corollary 4 and surrounding discussion for a full exposition. As a simple example, setting $\eta = 1/\operatorname{poly}(\kappa)$, our result shows the existence of matrices with $\log^2(\kappa)$ eigenvalues for which Theorem 2 requires $\Omega(\kappa^c)$ iterations for convergence if $O(\log\kappa)$ bits of precision are used. This is nearly exponentially worse than the $O(\log^2\kappa)$ iterations required for exact computation of $\mathbf{A}^{-1}\mathbf{x}$ in exact arithmetic by (3).

Theorem 3. *There exists a fixed constant $1/5 \le c \le 1/2$ such that for any $\kappa \ge 2$, $0 < \eta < \frac{1}{20\kappa^2}$, and $n \ge \lfloor \log_2 \kappa \rfloor \cdot \lceil \ln 1/\eta \rceil$, there is a positive definite $\mathbf{A} \in \mathbb{R}^{n\times n}$ with condition number $\le \kappa$, such that for any $k < \lfloor \kappa^c/377 \rfloor$*

$$\bar{\delta}_k \stackrel{\text{def}}{=} \min_{\substack{\text{polynomial } p \\ \text{with degree } < k}} \left(\max_{x \in \bigcup_{i=1}^n [\lambda_i(\mathbf{A})-\eta, \lambda_i(\mathbf{A})+\eta]} |p(x) - 1/x| \right) \ge 1/6.$$

We prove Theorem 3 by arguing that there is no polynomial $p(x)$ with degree $\le \kappa^c/377$ which has $p(0) = 1$ and $|p(x)| < 1/3$ for every $x \in \bigcup_{i=1}^n [\lambda_i(\mathbf{A}) - \eta, \lambda_i(\mathbf{A}) + \eta]$. Specifically, we show:

Lemma 12. *There exists a fixed constant $1/5 \le c \le 1/2$ and such that for any $\kappa \ge 2$, $0 < \eta \le \frac{1}{20\kappa^2}$ and $n \ge \lfloor \log_2 \kappa \rfloor \cdot \lceil \ln 1/\eta \rceil$, there are $\lambda_1, ..., \lambda_n \in [1/\kappa, 1]$, such that for any polynomial p with degree $k \le \kappa^c/377$ and $p(0) = 1$:*

$$\max_{x \in \bigcup_{i=1}^n [\lambda_i - \eta, \lambda_i + \eta]} |p(x)| \ge 1/3.$$

Lemma 12 can be viewed as an extension of the classic Markov brother's inequality [Mar90], which implies that any polynomial with $p(0) = 1$ and $|p(x)| \le 1/3$ for all $x \in [1/\kappa, 1]$ must have degree $\Omega(\sqrt{\kappa})$. Lemma 12 shows that even if we just restrict $|p(x)| \le 1/3$ on a few small subintervals of $[1/\kappa, 1]$, $\Omega(\kappa^c)$ degree is still required. We do not carefully optimize the constant c, although we believe it should be possible to improve to nearly 1/2 (see discussion in Appendix D). This would match the upper bound achieved by the Chebyshev polynomials of the first kind, appropriately shifted and scaled. Given Lemma 12 it is easy to show Theorem 3:

Proof of Theorem 3. Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be any matrix with eigenvalues equal to $\lambda_1, ..., \lambda_n$ – e.g. a diagonal matrix with these values as its entries. Assume by way of contradiction that there is a polynomial $p(x)$ with degree $k < \lfloor \kappa^c/377 \rfloor$ which satisfies:

$$\max_{x \in \bigcup_{i=1}^n [\lambda_i(\mathbf{A})-\eta, \lambda_i(\mathbf{A})+\eta]} |p(x) - 1/x| < 1/6.$$

Then if we set $\bar{p}(x) = 1 - xp(x)$, $\bar{p}(0) = 1$ and for any $x \in \bigcup_{i=1}^n [\lambda_i - \eta, \lambda_i + \eta]$,

$$|\bar{p}(x)| \le |xp(x) - 1| \le |x| \cdot |p(x) - 1/x| < \frac{|x|}{6} \le \frac{1}{3}$$

since $|x| \le 2$ when $\eta \le 1$. Since $\bar{p}(x)$ has degree $k + 1 \le \kappa^c/377$, it thus contradicts Lemma 12. $\square$

7.2 Hard instance construction

We begin by describing the "hard" eigenvalue distribution that is used to prove Lemma 12 for any given condition number $\kappa \ge 2$ and range radius η. Define $\lfloor \log_2(\kappa) \rfloor$ intervals:

$$I_i \stackrel{\text{def}}{=} \left[\frac{1}{2^i}, \frac{1}{2^{i-1}}\right] \text{ for } i = 1, \ldots, \lfloor \log_2(\kappa) \rfloor.$$

In each interval I_i we place z evenly spaced eigenvalues, where:

$$z = \lceil \ln 1/\eta \rceil.$$

That is, the eigenvalues in interval I_i are set to:

$$\lambda_{i,j} = \frac{1}{2^i} + \frac{j}{z 2^i} \text{ for } j = 1, \ldots, z. \tag{41}$$

Thus, our construction uses $\lfloor \log_2 \kappa \rfloor \cdot \lceil \ln 1/\eta \rceil$ eigenvalues total. The smallest is $> \frac{1}{\kappa}$ and the largest is ≤ 1, as required in the statement of Lemma 12. For convenience, we also define:

$$\mathcal{R}_{i,j} \stackrel{\text{def}}{=} [\lambda_{i,j} - \eta, \lambda_{i,j} + \eta] \qquad \mathcal{R}_i \stackrel{\text{def}}{=} \bigcup_j \mathcal{R}_{i,j} \qquad \text{and} \qquad \mathcal{R} \stackrel{\text{def}}{=} \bigcup_i \mathcal{R}_i.$$

By the assumption of Lemma 12 that $\eta \leq \frac{1}{20k^2}$, we have $\eta z = \eta \lceil \ln \frac{1}{\eta} \rceil \leq \sqrt{\eta} + \eta \leq \frac{1}{4\kappa}$. So none of the $\mathcal{R}_{i,j}$ overlap and in fact are distance at least $\frac{1}{2z\kappa}$ apart (since the eigenvalues themselves have spacing at least $\frac{1}{z\kappa}$ by (41)). An illustration is included in Figure 1.

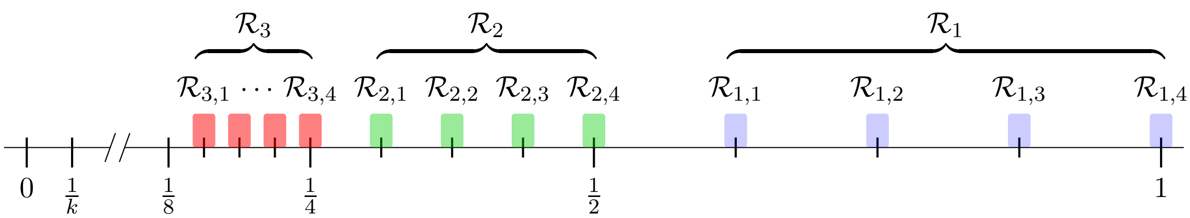

Figure 1: A sample "hard" distribution of eigenvalues with $z = 4$. The width of each range $\mathcal{R}_{i,j}$ is over-exaggerated for illustration – in reality each interval has width 2η, where $\eta \leq \frac{1}{4z\kappa}$.

7.3 Outline of the argument

Let p be any polynomial with degree k that satisfies $p(0) = 1$. To prove Lemma 12 we need to show that we cannot have $|p(x)| \leq 1/3$ for all $x \in \mathcal{R}$ unless k is relatively high (i.e. $\geq \kappa^c$). Let $r_1, \ldots, r_k$ denote p's k roots. So $|p(x)| = \prod_{i=1}^k |1 - \frac{x}{r_i}|$. Then define

$$g(x) \stackrel{\text{def}}{=} \ln |p(x)| = \sum_{i=1}^{k} \ln \left| 1 - \frac{x}{r_i} \right|. \tag{42}$$

To prove that $|p(x)| \geq 1/3$ for some $x \in \mathcal{R}$, it suffices to show that,

$$\max_{x \in \mathcal{R}} g(x) \geq -1. \tag{43}$$

We establish (43) via a potential function argument. For any positive weight function $w(x)$,

$$\max_{x \in \mathcal{R}} g(x) \geq \frac{\int_{\mathcal{R}} w(x) g(x) dx}{\int_{\mathcal{R}} w(x) dx}.$$

I.e., any weighted average lower bounds the maximum of a function. From (42), we have:

$$\frac{1}{k}\max_{x\in\mathcal{R}} g(x) \geq \frac{1}{k}\cdot\frac{\int_{\mathcal{R}} w(x)g(x)dx}{\int_{\mathcal{R}} w(x)dx} \geq \min_{r}\frac{\int_{\mathcal{R}} w(x)\ln|1-x/r|dx}{\int_{\mathcal{R}} w(x)dx}. \tag{44}$$

We focus on bounding this last quantity. More specifically, we set $w(x)$ to be:

$$w(x) \stackrel{\text{def}}{=} 2^{ic} \quad \text{for} \quad x\in\mathcal{R}_i.$$

The weight function increases from a minimum of $\sim 2^c$ to a maximum of $\sim \kappa^c$ as $x \in \mathcal{R}$ decreases from 1 towards $1/\kappa$. With this weight function, we will be able prove that (44) is lower bounded by $-O(\frac{1}{\kappa^c})$. It will then follow that (43) holds for any polynomial with degree $k = O(\kappa^c)$.

7.4 Initial Observations

Before giving the core argument, we make an initial observation that simplifies our analysis:

Claim 13. *If Lemma 12 holds for the hard instance described in Section 7.2 and all real rooted polynomials with roots on the range $[1/\kappa, 1+\eta]$, then it holds for all polynomials.*

Proof. We first show that we can consider just real rooted polynomials, before arguing that we can also assume their roots are within the range $[1/\kappa, 1+\eta]$.

Real rooted: If there is any polynomial equal to 1 at $x = 0$ with magnitude $\leq 1/3$ for $x \in \mathcal{R}$, then there must be a real polynomial (i.e. with real coefficients) of the same degree that only has smaller magnitude on $\mathcal{R}$. So we focus on $p(x)$ with real coefficients. Letting the roots of $p(x)$ be $r_1, \ldots, r_k$ and using that $p(0) = 1$, we can write:

$$p(x) = \prod_{i=1}^{k}(1 - x/r_i). \tag{45}$$

By the complex conjugate root theorem, any polynomial with real coefficients and a complex root must also have its conjugate as a root. Thus, if $p(x)$ has root $\frac{1}{a+bi}$ for some a, b, the above product contains a term of the form:

$$(1 - x(a - bi))(1 - x(a + bi)) = 1 - 2ax + a^2x^2 + b^2x^2.$$

If we just set $b = 0$ (i.e. take the real part of the root), $1 - 2ax + a^2x^2 + b^2x^2$ decreases for all $x > 0$. In fact, since $(1 - 2ax + a^2x^2) = (1 - ax)^2 > 0$, the absolute value $|1 - 2ax + a^2x^2 + b^2x^2|$ decreases if we set $b = 0$. Accordingly, by removing the complex part of p's complex root, we obtain a polynomial of the same degree that remains 1 at 0, but has smaller magnitude everywhere else.

Roots in eigenvalue range: First note that we can assume p doesn't have any negative roots: removing a term in (45) of the form $(1 - x/r_i)$ for $r_i < 0$ produces a polynomial with lower degree that is 1 at 0 but smaller in magnitude for all $x > 0$. It is not hard to see that by construction $\mathcal{R} \subseteq [1/\kappa, 1+\eta]$ and thus $x > 0$ for all $x \in \mathcal{R}$. Thus removing a negative root can only lead to smaller maximum magnitude over $\mathcal{R}$.

Now, suppose p has some root $0 < r < 1/\kappa$. For all $x \geq 1/\kappa$,

$$\left|1 - \frac{x}{1/\kappa}\right| < \left|1 - \frac{x}{r}\right|.$$

Accordingly, by replacing p's root at r with one at $(1/\kappa)$ we obtain a polynomial of the same degree that is smaller in magnitude for all $x \geq 1/\kappa$ and thus for all $x \in \mathcal{R} \subseteq [1/\kappa, 1+\eta]$.

Similarly, suppose p has some root $r > 1+\eta$. For all $x \leq 1+\eta$,

$$\left|1 - \frac{x}{1+\eta}\right| < \left|1 - \frac{x}{r}\right|.$$

So by replacing p's root at r with a root at $(1+\eta)$, we obtain a polynomial that has smaller magnitude everywhere in $\mathcal{R}$. $\square$

7.5 Main argument

With Claim 13, we are now ready to prove Lemma 12, which implies Theorem 3.

Proof of Lemma 12. Since we can restrict our attention to real rooted polynomials with each root $r_i \in [\frac{1}{\kappa}, 1+\eta]$, to prove (43) via (44) we just need to establish that:

$$\min_{r \in [\frac{1}{\kappa}, 1+\eta]} \frac{\int_{\mathcal{R}} w(x) \ln|1 - x/r| dx}{\int_{\mathcal{R}} w(x) dx} \geq -\frac{377}{\kappa^c}. \tag{46}$$

Consider the denominator of the left hand side:

$$\int_{\mathcal{R}} w(x) dx = \sum_{i=1}^{\lfloor \log_2(\kappa) \rfloor} \int_{\mathcal{R}_i} 2^{ic} dx = \sum_{i=1}^{\lfloor \log_2(\kappa) \rfloor} 2\eta z 2^{ic} \geq \eta z \kappa^c.$$

With this bound in place, to prove (46) we need to show:

$$\min_{r \in [\frac{1}{\kappa}, 1+\eta]} \int_{\mathcal{R}} w(x) \ln|1 - x/r| dx \geq -377\eta z.$$

Recalling our definition of $\mathcal{R}$, this is equivalent to showing that:

$$\text{For all } r \in \left[\frac{1}{\kappa}, 1+\eta\right], \qquad \sum_{i=1}^{\lfloor \log_2(\kappa) \rfloor} \int_{\mathcal{R}_i} w(x) \ln|1 - x/r| dx \geq -377\eta z. \tag{47}$$

To prove (47) we divide the sum into three parts. Letting $\lambda_{\ell,h}$ be the eigenvalue closest to r:

$$\sum_{i=1}^{\lfloor \log_2(\kappa) \rfloor} \int_{\mathcal{R}_i} w(x) \ln|1 - x/r| dx =$$

$$\sum_{i=1}^{\ell-2} \int_{\mathcal{R}_i} w(x) \ln|1 - x/r| dx \tag{48}$$

$$+ \sum_{i=\ell-1}^{\ell+1} \int_{\mathcal{R}_i} w(x) \ln|1 - x/r| dx \tag{49}$$

$$+ \sum_{i=\ell+2}^{\lfloor \log_2(\kappa) \rfloor} \int_{\mathcal{R}_i} w(x) \ln|1 - x/r| dx. \tag{50}$$

Note that when ℓ lies towards the limits of $\{1, \ldots, \lfloor \log_2(\kappa) \rfloor\}$, the sums in (50) and (48) may contain no terms and (49) may contain less than 3 terms.

To gain a better understanding of each of these terms, consider Figure 2, which plots $\ln|1-x/r|$ for an example value of r. (48) is a weighted integral over regions $\mathcal{R}_i$ that lie well above r. Specifically, for all $x \in \bigcup_{i=1}^{\ell-2} \mathcal{R}_i$, $x \geq 2r$ and thus $\ln|1-x/r|$ is *strictly positive*. Accordingly, (48) is a positive term and will help in our effort to lower bound (47).

On the other hand, (49) and (50) involve values of x which are close to r or lie below the root. For these values, $\ln|1-x/r|$ is *negative* and thus (49) and (50) will hurt our effort to lower bound (47). We need to show that the negative contribution cannot be too large.

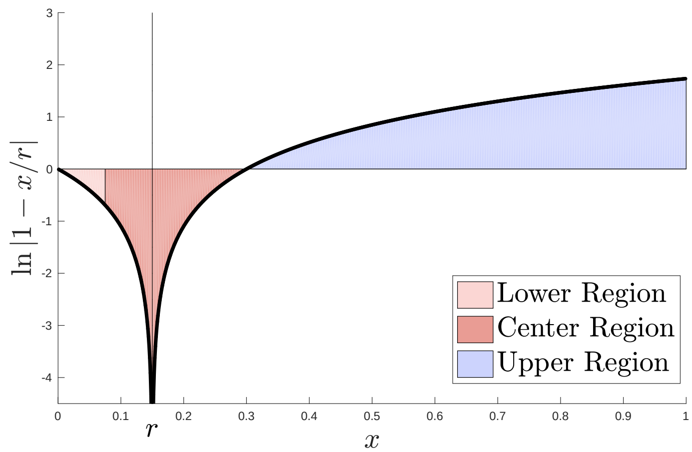

Figure 2: Plot of $\ln|1-x/r|$ for $r = 1/10$. Proving that (47) cannot be too small for any root r requires lower bounding a weighted integral of this function over $\mathcal{R} \subset [1/\kappa, 1+\eta]$.

Center region

We first evaluate (49), which is the range containing eigenvalues close to r. In particular, we start by just considering $\mathcal{R}_{\ell,h}$, the interval around the eigenvalue nearest to r.

$$\int_{\mathcal{R}_{\ell,h}} w(x) \ln|1-x/r| dx = 2^{\ell c} \int_{\lambda_{\ell,h}-\eta}^{\lambda_{\ell,h}+\eta} \ln|1-x/r| \geq 2^{\ell c} \int_{\lambda_{\ell,h}-\eta}^{\lambda_{\ell,h}+\eta} \ln|1-x/\lambda_{\ell,h}|.$$

The inequality follows because $\ln|1-x/r|$ strictly increases as x moves away from r. Accordingly, the integral takes on its maximum value when r is centered in the interval $[\lambda_{\ell,h}-\eta, \lambda_{\ell,h}+\eta]$.

$$2^{\ell c} \int_{\lambda_{\ell,h}-\eta}^{\lambda_{\ell,h}+\eta} \ln|1-x/\lambda_{\ell,h}| = 2^{\ell c+1} \int_0^{\eta} \ln \frac{x}{\lambda_{\ell,h}} = 2^{\ell c+1} \eta \left(\ln \eta - \ln \lambda_{\ell,h} - 1\right).$$

Since $\ln(\eta) \leq -1$ by the assumption that $\eta \leq \frac{1}{20\kappa^2}$ and since $-\ln \lambda_{\ell,h} \geq 0$ since $\lambda_{\ell,h} \leq 1$, we obtain:

$$\int_{\mathcal{R}_{\ell,h}} w(x) \ln|1-x/r| dx \geq 4 \cdot 2^{\ell c} \eta \ln \eta \geq -4 \cdot 2^{\ell c} \eta z. \tag{51}$$

Now we consider the integral over $\mathcal{R}_{\ell,i}$ for all $i \neq h$ and also over the entirety of $\mathcal{R}_{\ell+1}$ and $\mathcal{R}_{\ell-1}$. For all $x \in [\mathcal{R}_{\ell+1} \cup (\mathcal{R}_\ell \setminus \mathcal{R}_{\ell,h}) \cup \mathcal{R}_{\ell-1}]$, $w(x) \leq 2^{(\ell+1)c} \leq 2^{1/5} \cdot 2^{lc}$ since $c \geq 1/5$. So we have:

$$\begin{aligned}\int_{\mathcal{R}_{\ell+1}\cup(\mathcal{R}_\ell\setminus\mathcal{R}_{\ell,h})\cup\mathcal{R}_{\ell-1}} w(x)\ln|1-x/r|dx &\geq \int_{\mathcal{R}_{\ell+1}\cup(\mathcal{R}_\ell\setminus\mathcal{R}_{\ell,h})\cup\mathcal{R}_{\ell-1}} w(x)\min(\ln|1-x/r|,0)dx \\ &\geq 2^{1/5}\cdot 2^{lc}\int_{\mathcal{R}_{\ell+1}\cup(\mathcal{R}_\ell\setminus\mathcal{R}_{\ell,h})\cup\mathcal{R}_{\ell-1}} \min(\ln|1-x/r|,0)dx. \quad (52)\end{aligned}$$

where the last inequality holds by our bound on $w(x)$ and since $\min(\ln|1-x/r|, 0)$ is nonpositive.

The nearest eigenvalue to $\lambda_{\ell,h}$ is $\frac{1}{2^\ell z}$ away from it. Thus, the second closest eigenvalue to r besides $\lambda_{\ell,h}$ is at least $\frac{1}{2^{\ell+1}z}$ away from r. By our assumption that $\eta \leq \frac{1}{20\kappa^2}$, as discussed we have $\eta \leq \frac{1}{4\kappa z} \leq \frac{1}{2^{\ell+2}z}$. Thus, the closest interval to r besides $\mathcal{R}_{\ell,h}$ is at least $\frac{1}{2^{\ell+1}z} - \frac{1}{2^{\ell+2}z} \geq \frac{r}{8z}$ away.

Thus, using that $\ln|1-x/r|$ is strictly increasing as x moves away from r, that there are $3z-1$ eigenvalues in $\mathcal{R}_{\ell+1} \cup (\mathcal{R}_\ell \setminus \mathcal{R}_{\ell,h}) \cup \mathcal{R}_{\ell-1}$, and (52) we can lower bound the integral by:

$$\begin{aligned}&\int_{\mathcal{R}_{\ell+1}\cup(\mathcal{R}_\ell\setminus\mathcal{R}_{\ell,h})\cup\mathcal{R}_{\ell-1}} w(x)\ln|1-x/r|dx \\ &\qquad\geq 2^{1/5}\cdot 2^{lc}\cdot 2\eta \sum_{i\in\{-\lfloor 1.5z\rfloor,\ldots,\lfloor 1.5z\rfloor\}\setminus 0} \min\left(\ln\left|1-\frac{r(1+\frac{i}{8z})}{r}\right|, 0\right) \\ &\qquad\geq 4\cdot 2^{1/5}\eta\cdot 2^{lc}\sum_{i=1}^{\lfloor 1.5z\rfloor}\min(\ln(i/8z),0) \\ &\qquad\geq 4\cdot 2^{1/5}\eta\cdot 2^{lc}\int_{x=0}^{1.5z}\ln(x/8z)dx \\ &\qquad\geq -18.5\cdot 2^{lc}\eta z.\end{aligned}$$

This bound combines with (51) to obtain a final lower bound on (49) of:

$$\sum_{i=\ell-1}^{\ell+1}\int_{\mathcal{R}_i} w(x)\ln|1-x/r|dx \geq -22.5\cdot 2^{lc}\eta z. \quad (53)$$

Lower region

Next consider (50), which involves values that are at least a factor of 2 smaller than r. We have:

$$\text{For } j\geq 2 \text{ and } x\in\mathcal{R}_{\ell+j}, \quad \ln\left|1-\frac{x}{r}\right| \geq \ln\left(1-\frac{1}{2^{j-1}}\right) \geq -\frac{1.39}{2^{j-1}}.$$

For the last bound we use that $\frac{1}{2^{j-1}} \leq \frac{1}{2}$. It follows that:

$$\sum_{i=\ell+2}^{\lfloor\log_2(\kappa)\rfloor}\int_{\mathcal{R}_i} w(x)\ln|1-x/r|dx \geq \sum_{i=\ell+2}^{\lfloor\log_2(\kappa)\rfloor} -2.78\eta z\cdot\frac{2^{ic}}{2^{i-\ell-1}} = -5.56\cdot 2^{\ell c}\eta z\sum_{j=2}^{\lfloor\log_2(\kappa)\rfloor-\ell}\frac{1}{2^{j(1-c)}}. \quad (54)$$

Since we restrict $c \geq 1/5$, the sum above (which is positive) is at most:

$$\sum_{j=2}^{\lfloor\log_2(\kappa)\rfloor-\ell}\frac{1}{2^{j(1-c)}} \leq \frac{1}{2^{8/5}}\cdot\frac{1}{1-\frac{1}{2^{4/5}}} \leq .8$$

So we conclude using (54) that:

$$\sum_{i=\ell+2}^{\lfloor \log_2(\kappa) \rfloor} \int_{\mathcal{R}_i} w(x) \ln |1 - x/r| dx > -4.5 \cdot 2^{\ell c} \eta z. \tag{55}$$

Upper region

From (53) and (55), we see that (49) and (50) sum to $-O(2^{\ell c}\eta z)$. Recall that we wanted the entirety of (48) + (49) + (50) to sum to something greater than $-O(\eta z)$. For large values of ℓ (i.e., when r is small), the $2^{\ell c}$ term is problematic. It could be on the order $-\kappa^c$. If this is the case, we need to rely on a positive value of (48) to cancel out the negative contribution of (49) and (50). Fortunately, from the intuition provided by Figure 2, we expect (48) to increase as r decreases.

We start by noting that:

$$\text{For } j \geq 2 \text{ and } x \in \mathcal{R}_{\ell-j}, \quad \ln |1 - \frac{x}{r}| \geq \ln \left(2^{j-1} - 1\right) \geq \frac{j-2}{2}.$$

It follows that

$$\sum_{i=1}^{\ell-2} \int_{\mathcal{R}_i} w(x) \ln |1 - x/r| dx \geq \sum_{i=1}^{\ell-2} 2\eta \cdot 2^{ic} \cdot \frac{\ell - i - 2}{2} = 2^{\ell c} \eta z \sum_{i=1}^{\ell-2} \frac{\ell - i - 2}{2^{c(\ell-i)}}. \tag{56}$$

By our requirement that $c \geq 1/5$, as long as $\ell \geq 20$ we can explicitly compute:

$$\sum_{i=1}^{\ell-2} \frac{\ell - i - 2}{2^{c(\ell-i)}} = \frac{1}{2^{3c}} + \frac{2}{2^{4c}} + \ldots + \frac{\ell - 3}{2^{c(\ell-1)}} \geq 27.4 \tag{57}$$

which finally gives, using (56):

$$\sum_{i=1}^{\ell-2} \int_{\mathcal{R}_i} w(x) \ln |1 - x/r| dx \geq 27.4 \cdot 2^{\ell c} \eta z. \tag{58}$$

We note for the interested reader that (56) is the reason that we cannot set c too large (e.g. $c \geq 1/2$). If c is too large, the sum in (57) will be small, and will not be enough to cancel out the negative contributions from the center and lower regions.

7.6 Putting it all together

We can bound (47) using our bounds on the upper region (48) (given in (58)), the center region (49) (given in (53)) and the lower region (50) (given in (55)). As long as $\ell \geq 20$ we have:

$$\int_{\mathcal{R}} w(x) \ln |1 - x/r| dx \geq (-22.5 - 4.5 + 27.4) \cdot 2^{\ell c} \eta z \geq 0 \geq -\eta z.$$

It remains to handle the case of $\ell < 20$. In this case, the concerning $2^{\ell c}$ term is not a problem. Specifically, when $\ell < 20$ we have $2^{\ell c} \leq 2^{19/5}$. Even ignoring the positive contribution of (48), we can thus lower bound (47) using our center and lower region bounds by:

$$\int_{\mathcal{R}} w(x) \log |1 - x/r| dx \geq (-22.5 - 4.5) \cdot 2^{19/5} \cdot \eta z \geq -377 \eta z$$

and it follows that (46) is lower bounded by

$$\min_{r\in[\frac{1}{\kappa},1+\eta]} \frac{\int_{\mathcal{R}} w(x)\log|1-x/r|dx}{\int_{\mathcal{R}} w(x)dx} \geq -\frac{377}{\kappa^c}.$$

Then, by the argument outlined in Section 7.3, for any $k \leq \frac{\kappa^c}{377}$, there is no real rooted, degree k polynomial p with roots in $[\frac{1}{\kappa}, 1+\eta]$ such that:

$$p(0) = 1 \qquad \text{and} \qquad \log|p(x)| \leq -1 \text{ for all } x \in \mathcal{R}.$$

Finally, applying Claim 13 proves Lemma 12, as desired. □

8 Applications

In this section, we give example applications of Theorem 1 to matrix step function, matrix exponential, and top singular value approximation. We also show how Lanczos can be used to accelerate the computation of any function which is well approximated by a high degree polynomial with bounded coefficients. For each application, we show that Lanczos either improves upon or matches state-of-the-art runtimes, even when computations are performed with limited precision.

8.1 Matrix step function approximation

In many applications it is necessary to compute the matrix step function $s_\lambda(\mathbf{A})$ where

$$s_\lambda(x) \stackrel{\text{def}}{=} \begin{cases} 0 \text{ for } x < \lambda \\ 1 \text{ for } x \geq \lambda. \end{cases}$$

Computing $s_\lambda(\mathbf{A})\mathbf{x}$ is equivalent to projecting $\mathbf{x}$ onto the span of all eigenvectors of $\mathbf{A}$ with eigenvalue $\geq \lambda$. This projection is useful in data analysis algorithms that preprocess data points by projecting onto the top principal components of the data set – here $\mathbf{A}$ would be the data covariance matrix, whose eigenvectors correspond to principal components of the data. For example, as shown in [FMMS16] and [AZL17], an algorithm for approximating $s_\lambda(\mathbf{A})\mathbf{x}$ can be used to efficiently solve the *principal component regression* problem, a widely used form of regularized regression. A projection algorithm can also be used to accelerate spectral clustering methods [TPGV16].

The matrix step function $s_\lambda(\mathbf{A})$ is also useful because $\operatorname{tr}(s_\lambda(\mathbf{A}))$ is equal to the number of eigenvalues of $\mathbf{A}$ which are $\geq \lambda$. This trace can be estimated up to $(1 \pm \epsilon)$ relative error with probability $1-\delta$ by computing $\mathbf{x}^T s_\lambda(\mathbf{A})\mathbf{x}$ for $O(\log(1/\delta)/\epsilon^2)$ random sign vectors [Hut90]. By composing step functions at different thresholds and using this trace estimation technique, it is possible to estimate the number of eigenvalues of $\mathbf{A}$ in any interval $[a, b]$, which is a useful primitive in estimating numerical rank [US16], tuning eigensolvers and other algorithms [DNPS16], and estimating the value of matrix norms [MNS+17].

Soft step function application via Lanczos

Due to its discontinuity at λ, $s_\lambda(x)$ cannot be uniformly approximated on the range of $\mathbf{A}$'s eigenvalues by any polynomial. Thus, we cannot apply Theorem 1 directly. However, it typically suffices to apply a softened step function that is allowed to deviate from the true step function in a small range around λ. For simplicity, we focus on applying such a function with $\lambda = 0$. Specifically, we wish to apply $h(\mathbf{A})$ where:

$$h(x) \in \begin{cases} [0, \epsilon] \text{ for } x < -\gamma \\ [0, 1] \text{ for } x \in [-\gamma, \gamma] \\ [1-\epsilon, 1] \text{ for } x \geq \gamma. \end{cases} \tag{59}$$

For a positive semidefinite $\mathbf{A}$, by applying h to $\mathbf{B} = \mathbf{A}(\mathbf{A} + \lambda \mathbf{I})^{-1} - \frac{1}{2}\mathbf{I}$, we can recover a soft step function at λ, which, for example, provably suffices to solve principal component regression [FMMS16, AZL17] and to perform the norm estimation algorithms of [MNS+17]. We just need to apply $\mathbf{B}$ to the precision specified in Requirement 2, which can be done, for example, using a fast iterative linear system solver.

In [FMMS16], Corollary 5.4, it is proven that for $q = O(\gamma^{-2} \log(1/\epsilon))$ the polynomial:

$$p_q(x) = \sum_{i=0}^{q} \left(x(1-x^2)^i \prod_{j=1}^{i} \frac{2j-1}{2j} \right) \tag{60}$$

is a valid softened sign function satisfying (59). Additionally, it is shown in Lemma 5.5 that there is a lower degree polynomial $p^*(x)$ with degree $O(\gamma^{-1} \log(1/\epsilon\gamma))$ which uniformly approximates $p_q(x)$ to error ϵ on the range $[-1, 1]$. Combining these two results we can apply Theorem 1 to obtain:

Theorem 14 (Approximation of soft matrix sign function)**.** *Given $\mathbf{B} \in \mathbb{R}^{n \times n}$ with $\|\mathbf{B}\| \leq 1/2$, $\mathbf{x} \in \mathbb{R}^n$, and $\epsilon < 1$, let $B = \log\left(\frac{n}{\epsilon\gamma}\right)$ and $q = O(\gamma^{-2} \log(1/\epsilon))$. Suppose Algorithm 1 is run with $f(x) = p_q(x)$, which is a function satisfying (59). After $k = O(\gamma^{-1} \log(1/\epsilon\gamma))$ iterations on a computer satisfying Requirement 1 and Requirement 2 for applying $\mathbf{B}$ to precision $\epsilon_{\text{mach}} = 2^{-\Omega(B)}$ (e.g. a computer with $\Omega(B)$ bits of precision), the algorithm outputs $\mathbf{y}$ with $\|p_q(\mathbf{B})\mathbf{x} - \mathbf{y}\| \leq \epsilon\|\mathbf{x}\|$. The total runtime is $O(\text{mv}(\mathbf{B})k + k^2 B + kB^2)$.*

Note that the assumption $\|\mathbf{B}\| \leq \frac{1}{2}$ allows us to set $\eta = \Theta(\|\mathbf{B}\|)$ and still have $[\lambda_{\min}(\mathbf{B}) - \eta, \lambda_{\max}(\mathbf{B}) + \eta] \subseteq [-1, 1]$, so we can apply the uniform approximation bound of [FMMS16]. If we apply Theorem 14 to $\mathbf{B} = \mathbf{A}(\mathbf{A} + \lambda \mathbf{I})^{-1} - \frac{1}{2}\mathbf{I}$ for PSD $\mathbf{A}$ to compute the step function at λ, the assumption $\|\mathbf{B}\| \leq 1/2$ holds.

Comparision with prior work

[AZL17] shows how to directly apply a polynomial with degree $O(\gamma^{-1} \log(1/\epsilon\gamma))$ which approximates a softened sign function. Furthermore, this application can be made stable using the stable recurrence for Chebyshev polynomial computation, and thus matches Theorem 14. Both [FMMS16] and [AZL17] acknowledge Lanczos as a standard method for applying matrix sign functions, but avoid the method due to the lack of a complete theory for its approximation quality. Theorem 14 demonstrates that end-to-end runtime bounds can in fact be achieved for the Lanczos method, matching the state-of-the-art given in [AZL17].

8.2 Matrix exponential approximation

We next consider the matrix exponential, which is applied widely in numerical computation, theoretical computer science, and machine learning. For example, computing $\exp(-\mathbf{A})\mathbf{x}$ for a PSD

$\mathbf{A}$ is an important step in the matrix multiplicative weights method for semidefinite programming [AHK05, Kal07] and in the balanced separator algorithm of [OSV12]. When $\mathbf{A}$ is a graph adjacency matrix, $\text{tr}(\exp(\mathbf{A}))$ is known as the *Estrada index*. As in the case of the sign function, its value can be estimated to $(1 \pm \epsilon)$ multiplicative error with probability $1 - \delta$ if $\exp(\mathbf{A})$ is applied to $O(\log(1/\delta)\epsilon^{-2})$ random vectors [HMAS17].

Approximating the matrix exponential, including via the Lanczos method [Saa92, DGK98], has been widely studied – see [ML03] for a review. Here we use our results to give general end-to-end runtime bounds for this problem in finite precision, which as far as we know are state-of-the-art.

Approximation of $\exp(\mathbf{A})$ for general $\mathbf{A}$

We can apply Theorem 1 directly to the matrix exponential. $\exp(x)$ can be uniformly approximated up to error ϵ for $x \in [a, b]$ with a truncated Taylor series expansion at $(b + a)/2$ with degree $k = O((b - a) + \log(e^{b+a}/\epsilon))$ (see e.g. Lemma 7.5 of [OSV12] with δ set to $\delta = \epsilon \cdot e^{-(b+a)/2}$). Applying Theorem 1 with $\eta = \|\mathbf{A}\|$ we have:

Theorem 15 (General matrix exponential approximation)**.** *Given symmetric $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{x} \in \mathbb{R}^n$, and $\epsilon \leq 1$, let $B = \log\left(\frac{\max(n,\|\mathbf{A}\|)}{\epsilon}\right)$. If Algorithm 1 is run with $f(x) = \exp(x)$ for $k = O(\|\mathbf{A}\| + \log(1/\epsilon))$ iterations on a computer satisfying Requirements 1 and 2 for precision $\epsilon_{\text{mach}} = 2^{-\Omega(B)}$ (e.g. a computer using $\Omega(B)$ bits of precision), it outputs $\mathbf{y}$ satisfying $\|\exp(\mathbf{A})\mathbf{x} - \mathbf{y}\| \leq \epsilon C\|\mathbf{x}\|$ where $C = e^{2\|\mathbf{A}\|}$. The total runtime is $O(\text{mv}(\mathbf{A})k + k^2B + kB^2)$.*

Approximation of $\exp(-\mathbf{A})$ for positive semidefinite $\mathbf{A}$

In applications such as to the matrix multiplicative weights update method and the balanced separator algorithm of [OSV12], we are interesting in computing $\exp(-\mathbf{A})$ for positive semidefinite $\mathbf{A}$. In this case a better bound is achievable. Using Theorem 7.1 of [OSV12], the linear dependence on $\|\mathbf{A}\|$ in the iterations required for Theorem 15 can be improved to $\tilde{O}(\sqrt{\|\mathbf{A}\|})$. Additionally, since $-\mathbf{A}$ has only non-positive eigenvalues, we can set $C = O(1)$.

However, the runtime of Lanczos still has a $\tilde{O}(k^2)$ term. We can significantly reduce k and thus improve this cost via the rational approximation technique used in [OSV12]. Specifically, $\exp(\mathbf{A})$ can be approximated via a $k = O(\log(1/\epsilon))$ degree polynomial in $(\mathbf{I} + \frac{1}{k}\mathbf{A})^{-1}$. Further, our stability results immediately imply that it suffices to compute an approximation to this inverse, using e.g. the conjugate gradient method. Specifically we have:

Theorem 16 (Improved matrix exponential approximation)**.** *Given PSD $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{x} \in \mathbb{R}^n$, and $\epsilon < 1$, let $k = O(\log(1/\epsilon))$, $B = \log\left(\frac{n\max(\|\mathbf{A}\|,1)}{\epsilon}\right)$, and $\epsilon_1 = 2^{-\Omega(B)}$. Let $\mathcal{A}(\mathbf{A}, \mathbf{w}, k, \epsilon_1)$ be an algorithm returning $\mathbf{z}$ with $\|\left(\mathbf{I} + \frac{1}{k}\mathbf{A}\right)^{-1}\mathbf{w} - \mathbf{z}\| \leq \epsilon_1\|\mathbf{w}\|$ for any $\mathbf{w}$. There is an algorithm running on a computer with $\Omega(B)$ bits of precision that makes k calls to $\mathcal{A}(\mathbf{A}, \mathbf{w}, k, \epsilon_1)$ and uses $O(nk + kB^2)$ additional time to return $\mathbf{y}$ satisfying: $\|\exp(-\mathbf{A})\mathbf{x} - \mathbf{y}\| \leq \epsilon\|\mathbf{x}\|$.*

Theorem 16 can be compared to Theorem 6.1 of [OSV12]. It has an improved dependence on k since the modified Lanczos algorithm used in [OSV12] employs reorthogonalization at each iteration and thus incurs a cost of $O(nk^2)$. Additionally, [OSV12] focuses on handling error due to the approximate application of $(\mathbf{I} + \frac{1}{k}\mathbf{A})^{-1}$, but assumes exact arithmetic for all other operations.

Proof. We apply Theorem 1 with matrix $\mathbf{B} = (\mathbf{I} + \frac{1}{k}\mathbf{A})^{-1}$ and $f(x) = e^{-k/x+k}$. We can write $\mathbf{B} = g(\mathbf{A})$ where $g(x) = \frac{1}{1+x/k}$ and thus have $f(\mathbf{B}) = \exp(-\mathbf{A})$ since $f(g(x)) = e^{-x}$.

Additionally, $\mathbf{B}$'s eigenvalues all fall between $\frac{k}{\|\mathbf{A}\|}$ and 1. Set $\eta = \min(\frac{k}{2\|\mathbf{A}\|}, \frac{\epsilon}{ck^3}) \leq \|\mathbf{B}\|$ for sufficiently large constant c. Then for all $x \in [\lambda_{\min}(\mathbf{B}) - \eta, \lambda_{\max}(\mathbf{B}) + \eta]$, we can loosely bound:

$$|f(x)| \leq e^{-k/(1+\eta)+k} \leq e^{-k/(1+1/k)+k} \leq e. \tag{61}$$

By Corollary 6.9 of [OSV12], there is a degree k polynomial $p^*(x)$ satisfying $p^*(0) = 0$ and:

$$\sup_{x\in(0,1]} |f(x) - p^*(x)| = O(k \cdot 2^{-k}). \tag{62}$$

We need to bound the error of approximation on the slightly extended range $[\lambda_{\min}(\mathbf{B}) - \eta, \lambda_{\max}(\mathbf{B}) + \eta] \subset (0, 1 + \eta]$. We do this simply by arguing that $f(x)$ and $p^*(x)$ cannot diverge substantially over the range $[1, 1 + \eta]$. In this range we can bound $f(x)$:

$$1 < e^{-k/x+k} \leq e^{\frac{-k}{1+\epsilon/ck^3}+k} \leq e^{\frac{-ck^4+ck^4+k\epsilon}{ck^3+\epsilon}} \leq e^{\epsilon/(ck^2)} \leq 1 + O\left(\frac{\epsilon}{k^2}\right). \tag{63}$$

Additionally, by the Markov brother's inequality, any degree k polynomial $p(x)$ with $|p(x)| \leq 1$ for $x \in [-1, 1]$ has derivative $p'(x) \leq k^2$ on the same range. By (62), if we set $k = c\log(1/\epsilon)$ for large enough constant c, we have $\sup_{x\in(0,1]} |f(x) - p^*(x)| = O\left(\frac{\epsilon}{k}\right)$. Since $f(x) \leq 1$ on this range, we thus loosely have $|p^*(x)| \leq 2$ for $x \in [0, 1]$. We can then claim that p^* changes by at most $O\left(\frac{\epsilon}{k}\right)$ on $[1, 1+\eta]$, which has width $O\left(\frac{\epsilon}{k^3}\right)$. Otherwise, p^* would have derivative $\geq ck^2$ for some constant c at some point in this range, contradicting Markov's inequality after appropriately shifting and scaling p^* to have magnitude bounded by 1 on $[-1, 1]$. Overall, combined with (63) we have:

$$\delta_k \leq \max_{x\in[\lambda_{\min}(\mathbf{B})-\eta,\lambda_{\max}(\mathbf{B})+\eta]} |f(x) - p(x)| = O\left(\frac{\epsilon}{k}\right).$$

Theorem 1 applies with $C = e$ from (61), $k = O(\log(1/\epsilon))$ and $\eta = \min(\frac{k}{2\|\mathbf{A}\|}, \frac{\epsilon}{ck^3})$ as long as we use $\Omega\left(\log\left(\frac{nk\|\mathbf{B}\|}{\epsilon\eta}\right)\right) = \Omega\left(\log\left(\frac{n\max(\|\mathbf{A}\|,1)}{\epsilon}\right)\right)$ bits of precision (to satisfy Requirement 1) and can compute $\mathbf{Bw}$ up to error $\epsilon_1\|\mathbf{w}\|$ for any $\mathbf{w}$ (to satisfy Requirement 2). Accordingly,

$$\|f(\mathbf{B})\mathbf{x} - \mathbf{y}\| = \|\exp(-\mathbf{A})\mathbf{x} - \mathbf{y}\| \leq \epsilon\|\mathbf{x}\|.$$

□

For the balanced separator algorithm of [OSV12], the linear system solver $\mathcal{A}(\mathbf{A}, \mathbf{w}, k, \epsilon_1)$ can be implemented used a fast, near linear time Laplacian system solver. For general matrices, it can be implemented via the conjugate gradient method. Applying Theorem 2 to $\mathbf{B} = (\mathbf{I} + \frac{1}{k}\mathbf{A})$, setting $\eta = \lambda_{\min}(\mathbf{B})/2 \geq 1/2$ and $k = O\left(\log(\kappa(\mathbf{B})/\epsilon_1) \cdot \sqrt{\kappa(\mathbf{B})}\right)$ ensures that CG computes $\mathbf{y}$ satisfying $\|\left(\mathbf{I} + \frac{1}{k}\mathbf{A}\right)^{-1}\mathbf{w} - \mathbf{z}\| \leq \epsilon_1\|\mathbf{w}\|$ if $\Omega\left(\log(n\kappa(\mathbf{B})/\epsilon_1)\right)$ bits of precision are used. Additionally, we can multiply a vector by $\mathbf{B}$ in time $\mathrm{mv}(\mathbf{B}) = \mathrm{mv}(\mathbf{A}) + n$. Plugging in $\kappa(\mathbf{B}) \leq 1 + \|\mathbf{A}\|/k$ gives:

Corollary 17. *Given PSD $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{x} \in \mathbb{R}^n$, and $\epsilon < 1$, there exists an algorithm running on a computer with $B = \Omega\left(\log\left(\frac{n\max(\|\mathbf{A}\|,1)}{\epsilon}\right)\right)$ bits of precision which returns $\mathbf{y}$ satisfying $\|\exp(-\mathbf{A})\mathbf{x} - \mathbf{y}\| \leq \epsilon\|\mathbf{x}\|$ in $O\left(\left[(\mathrm{mv}(\mathbf{A}) + n)\log\left(\frac{n\max(\|\mathbf{A}\|,1)}{\epsilon}\right)\sqrt{\frac{\|\mathbf{A}\|}{\log(1/\epsilon)} + 1} + \log^2\left(\frac{n\max(\|\mathbf{A}\|,1)}{\epsilon}\right)\right] \cdot \log\frac{1}{\epsilon}\right)$ time.*

8.3 Top singular value approximation

Beyond applications to matrix functions, the Lanczos method and related Krylov subspace methods are the most common iterative algorithms for computing approximate eigenvectors and eigenvalues of symmetric matrices. Once $\mathbf{Q}$ and $\mathbf{T}$ are obtained by Algorithm 1 (or a variant) the *Rayleigh-Ritz method* can be used to find approximate eigenpairs for $\mathbf{A}$. Specifically, $\mathbf{T}$'s eigenvalues are taken as approximate eigenvalues and $\mathbf{Q}\mathbf{v}_i$ is taken as an approximate eigenvector for each eigenvector $\mathbf{v}_i$ of $\mathbf{T}$. For a *non-symmetric matrix* $\mathbf{B}$, the Lanczos method can be used to find approximate *singular vectors and values* since these correspond to eigenpairs of $\mathbf{B}^T\mathbf{B}$ and $\mathbf{B}\mathbf{B}^T$.

Substantial literature studies the accuracy of these approximations, both under exact arithmetic and finite precision. While addressing the stability of the Rayleigh-Ritz method is beyond the scope of this work, it turns out that, unmodified, our Theorem 1 can prove the stability of a related algorithm for the common problem of approximating just the top singular value of a matrix. In particular, for error parameter Δ, our goal is to find some vector $\mathbf{u}$ such that:

$$\frac{\|\mathbf{B}\mathbf{u}\|}{\|\mathbf{u}\|} \geq (1-\Delta)\max_{\mathbf{x}} \frac{\|\mathbf{B}\mathbf{x}\|}{\|\mathbf{x}\|}. \tag{64}$$

Here $\max_{\mathbf{x}} \frac{\|\mathbf{B}\mathbf{x}\|}{\|\mathbf{x}\|} = \|\mathbf{B}\| = \sigma_{\max}(\mathbf{B})$ is $\mathbf{B}$'s top singular value. In addition to being a fundamental problem in its own right, via deflation techniques, an algorithm for approximating the top singular vector of a matrix can also be used for the important problem of finding a nearly optimal low-rank matrix approximation to $\mathbf{B}$ [AZL16].

Suppose we have $\mathbf{B} \in \mathbb{R}^{m\times n}$ that we can multiply on the right by a vector in $\mathrm{mv}(\mathbf{B})$ time. In *exact arithmetic* a vector $\mathbf{u}$ satisfying (64) can be found in time (see e.g. [SV14]):

$$O\left(\mathrm{mv}(\mathbf{B})\sqrt{1/\Delta}\log\frac{n}{\Delta} + \left(\sqrt{1/\Delta}\log\frac{n}{\Delta}\right)^2\right).$$

Note that, unlike other commonly stated bounds for singular vector approximation with the Lanczos method, this runtime does not have a dependence on the gaps between $\mathbf{B}$'s singular values – i.e. it does not require a sufficiently large gap to obtain high accuracy. Since the second term is typically dominated by the first, it is an improvement over the $\tilde{O}(\mathrm{nnz}(\mathbf{B})/\Delta)$ gap-independent runtime required, for example, by the standard power method.

We can use Theorem 1 to prove, to the best of our knowledge, the first rigorous gap-independent bound for Lanczos that holds in finite precision. It essentially matches the algorithm's exact arithmetic runtime for singular vector approximation.

Theorem 18 (Approximating the top singular vector and value)**.** *Suppose we are given $\mathbf{B} \in \mathbb{R}^{m\times n}$ and error parameter $\Delta \leq 1/2$. Let $q = \frac{4}{\Delta}\log\frac{n}{\Delta}$, $B = \log\frac{n}{\Delta}$, and let $\mathbf{z} \in \{-1,1\}^n$ be chosen randomly by selecting each entry to be 1 with probability 1/2 and −1 otherwise. If Algorithm 1 is run with $f(x) = x^q$ on $\mathbf{B}^T\mathbf{B}$ and input vector $\mathbf{z}$ for $O(\sqrt{1/\Delta}\log\frac{n}{\Delta})$ iterations on a computer satisfying Requirement 1 and Requirement 2 with precision $\epsilon_{\mathrm{mach}} = 2^{-\Omega(B)}$ (e.g. a computer with $\Omega(B)$ bits of precision), then with probability $\geq 1/2$, $\mathbf{y} = \mathbf{Q}\mathbf{T}^q\mathbf{e}_1$ satisfies*

$$\frac{\|\mathbf{B}\mathbf{y}\|}{\|\mathbf{y}\|} \geq (1-\Delta)\max_{\mathbf{x}}\frac{\|\mathbf{B}\mathbf{x}\|}{\|\mathbf{x}\|} = (1-\Delta)\|\mathbf{B}\|.$$

$\mathbf{y}$ takes $O(\mathrm{mv}(\mathbf{B})\sqrt{1/\Delta}\log\frac{n}{\Delta} + \left(\sqrt{1/\Delta}\log\frac{n}{\Delta}\right)^2 B + \left(\sqrt{1/\Delta}\log\frac{n}{\Delta}\right)B^2)$ time to compute. Note that if this randomized procedure is repeated $O(\log(1/\delta))$ times and the $\mathbf{y}$ maximizing $\|\mathbf{B}\mathbf{y}\|/\|\mathbf{y}\|$ is selected, then it will satisfy the guarantee with probability $(1-\delta)$.

Before applying our results on function approximation under finite precision, to prove the theorem we first need to argue that, if computed *exactly*, $\hat{\mathbf{y}} = \left(\mathbf{B}^T\mathbf{B}\right)^q \mathbf{z}$ provides a good approximate top eigenvector. Doing so amounts to a standard analysis of the power method with a random starting vector, which we include below:

Lemma 19 (Power method)**.** *For any $\mathbf{B} \in \mathbb{R}^{m\times n}$, $\mathbf{z} \in \{-1,1\}^n$ a random sign vector as described in Theorem 18, $\Delta \leq 1/2$, and $q = \frac{4}{\Delta}\log\frac{n}{\Delta}$, with probability $\geq 1/2$, $\hat{\mathbf{y}} = \left(\mathbf{B}^T\mathbf{B}\right)^q \mathbf{z}$ satisfies $\|\hat{\mathbf{y}}\| \geq \frac{\|\mathbf{B}^T\mathbf{B}\|^q}{n}$ and:*

$$\frac{\|\mathbf{B}\hat{\mathbf{y}}\|}{\|\hat{\mathbf{y}}\|} \geq (1-\Delta/2)\max_{\mathbf{x}} \frac{\|\mathbf{B}\mathbf{x}\|}{\|\mathbf{x}\|}.$$

Proof. Let $\mathbf{B}^T\mathbf{B} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$ be an eigendecomposition of the PSD matrix $\mathbf{B}^T\mathbf{B}$. $\mathbf{V}$ is orthonormal with columns $\mathbf{v}_1, \ldots, \mathbf{v}_n$ and $\mathbf{\Lambda}$ is a positive diagonal matrix with entries $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$.

$$\|\hat{\mathbf{y}}\|^2 = \mathbf{z}^T\mathbf{V}\mathbf{\Lambda}^q\mathbf{V}^T\mathbf{z} = \sum_{i=1}^n \lambda_i^q \left(\mathbf{z}^T\mathbf{v}_i\right)^2 \quad \text{and} \quad \|\mathbf{B}\hat{\mathbf{y}}\|^2 = \mathbf{z}^T\mathbf{V}\mathbf{\Lambda}^{q+1}\mathbf{V}^T\mathbf{z} = \sum_{i=1}^n \lambda_i^{q+1} \left(\mathbf{z}^T\mathbf{v}_i\right)^2.$$

Let λ_r be the smallest eigenvalue with $\lambda_r \geq (1-\Delta/2)\lambda_1$. Then note that $\|\mathbf{B}\hat{\mathbf{y}}\|^2 = \sum_{i=1}^n \lambda_i^{q+1}\left(\mathbf{z}^T\mathbf{v}_i\right)^2 \geq \sum_{i=1}^r \lambda_i^{q+1}\left(\mathbf{z}^T\mathbf{v}_i\right)^2 \geq (1-\Delta/2)\lambda_1 \sum_{i=1}^r \lambda_i^q \left(\mathbf{z}^T\mathbf{v}_i\right)^2$. It follows that:

$$\frac{\|\mathbf{B}\hat{\mathbf{y}}\|^2}{\|\hat{\mathbf{y}}\|^2} \geq (1-\Delta/2)\lambda_1 \frac{\sum_{i=1}^r \lambda_i^q \left(\mathbf{z}^T\mathbf{v}_i\right)^2}{\sum_{i=1}^r \lambda_i^q \left(\mathbf{z}^T\mathbf{v}_i\right)^2 + \sum_{i=r+1}^n \lambda_i^q \left(\mathbf{z}^T\mathbf{v}_i\right)^2}. \tag{65}$$

We want to show that $\sum_{i=r+1}^n \lambda_i^q \left(\mathbf{z}^T\mathbf{v}_i\right)^2$ is small in comparison to $\sum_{i=1}^r \lambda_i^q \left(\mathbf{z}^T\mathbf{v}_i\right)^2$ so that the entire fraction in (65) is not much smaller than 1. In fact, we will show that the first quantity is small in comparison to $\lambda_1^q \left(\mathbf{z}^T\mathbf{v}_i\right)^2$, which is sufficient.

Since $q = \frac{4}{\Delta}\log\frac{n}{\Delta}$ and $\frac{\lambda_i}{\lambda_1} < (1-\Delta/2)$ for $i \geq r+1$, using the fact that $(1-x)^{1/x} \leq 1/e$ for $x \in [0,1]$, it is not hard to check that:

$$\left(\frac{\lambda_i}{\lambda_1}\right)^q \leq \frac{\Delta^2}{n^2}. \tag{66}$$

Additionally, since $\mathbf{z}$ is chosen randomly, with good probability, we don't expect that $\sum_{i=r+1}^n \left(\mathbf{z}^T\mathbf{v}_i\right)^2$ will be much larger than $\left(\mathbf{z}^T\mathbf{v}_1\right)^2$. Since $\|\mathbf{v}_1\|$ is a unit vector, it must have some entry i with absolute value $\geq \frac{1}{\sqrt{n}}$. For any randomly drawn sign vector $\mathbf{z}$, let $\bar{\mathbf{z}}$ be the same vector, but with the sign of this i^{th} entry flipped. Since $\mathbf{v}_1$'s i^{th} entry has magnitude $\geq \frac{1}{\sqrt{n}}$, it holds that:

$$|\mathbf{z}^T\mathbf{v}_1 - \bar{\mathbf{z}}^T\mathbf{v}_1| \geq \frac{2}{\sqrt{n}}.$$

Accordingly, for any $\mathbf{z}$, either $|\mathbf{z}^T\mathbf{v}_1|$ or $|\bar{\mathbf{z}}^T\mathbf{v}_1|$ is $\geq \frac{1}{\sqrt{n}}$. We conclude that for a *randomly drawn* $\mathbf{z}$, with probability $\geq 1/2$,

$$\left(\mathbf{z}^T\mathbf{v}_1\right)^2 \geq \frac{1}{n}.$$

This immediately gives by our formula for $\|\hat{\mathbf{y}}\|$ our first claim that $\|\hat{\mathbf{y}}\| \geq \lambda_1^q/n$. Furthermore,

$$\sum_{i=r+1}^{n} \left(\mathbf{z}^T \mathbf{v}_i\right)^2 \leq \sum_{i=1}^{n} \left(\mathbf{z}^T \mathbf{v}_i\right)^2 = \|\mathbf{V}\mathbf{z}\|^2 = \|\mathbf{z}\|^2 = n$$

so we can conclude that

$$\left(\mathbf{z}^T \mathbf{v}_1\right)^2 \geq \frac{1}{n^2} \sum_{i=r+1}^{n} \left(\mathbf{z}^T \mathbf{v}_i\right)^2. \tag{67}$$

Combining (66) and (67) and noting that $\frac{\Delta^2}{n^2} \leq \frac{\Delta}{2n^2}$ for $\Delta \leq \frac{1}{2}$, we have that:

$$\sum_{i=r+1}^{n} \lambda_i^q \left(\mathbf{z}^T \mathbf{v}_i\right)^2 \leq n^2 \cdot \frac{\Delta}{2n^2} \lambda_1^q \left(\mathbf{z}^T \mathbf{v}_1\right)^2 \leq \frac{\Delta}{2} \sum_{i=1}^{r} \lambda_i^q \left(\mathbf{z}^T \mathbf{v}_i\right)^2$$

Plugging into (65) , we have that

$$\frac{\|\mathbf{B}\hat{\mathbf{y}}\|^2}{\|\hat{\mathbf{y}}\|^2} \geq (1-\Delta/2)\lambda_1 \frac{\sum_{i=1}^{r} \lambda_i^q \left(\mathbf{z}^T \mathbf{v}_i\right)^2}{\sum_{i=1}^{r} \lambda_i^q \left(\mathbf{z}^T \mathbf{v}_i\right)^2 + \frac{\Delta}{2} \sum_{i=1}^{r} \lambda_i^q \left(\mathbf{z}^T \mathbf{v}_i\right)^2} \geq \frac{1-\Delta/2}{1+\Delta/2} \lambda_1.$$

Since $\sqrt{\frac{1-\Delta/2}{1+\Delta/2}} \geq (1-\Delta/2)$, we conclude that $\frac{\|\mathbf{B}\hat{\mathbf{y}}\|}{\|\hat{\mathbf{y}}\|} \geq (1-\Delta/2)\sqrt{\lambda_1}$ and the lemma follows since $\sqrt{\lambda_1} = \max_{\mathbf{x}} \frac{\|\mathbf{B}\mathbf{x}\|}{\|\mathbf{x}\|}$. □

With Lemma 19 in place, we prove our main result on approximating the top singular vector.

Proof of Theorem 18. We begin with Theorem 3.3 of [SV14], which says that for any q, there is a polynomial p with degree $k = \left\lceil \sqrt{2q \log(2/\delta)} \right\rceil$ such that:

$$\text{For all } x \in [-1,1] \qquad\qquad |x^q - p(x)| \leq \delta. \tag{68}$$

Denote $\lambda_{\max} \stackrel{\text{def}}{=} \lambda_{\max}(\mathbf{B}^T\mathbf{B}) = \|\mathbf{B}^T\mathbf{B}\|$. If we set $\eta = \frac{\lambda_{\max}}{q}$, after scaling, (68) shows that there exists a degree k polynomial $p'(x)$ satisfying:

$$|x^q - p'(x)| \leq [(1+1/q)\,\lambda_{\max}]^q \cdot \delta \leq e\delta\lambda_{\max}^q$$

on the range $[\lambda_{\min} - \eta, \lambda_{\max} + \eta]$.

Set $q = \frac{4}{\Delta} \log \frac{n}{\Delta}$ as in Theorem 19 and $k = \Theta\left(\min\left(\sqrt{q \log(2qn/\Delta)}, q\right)\right) = O(\sqrt{1/\Delta} \log \frac{n}{\Delta})$. Theorem 1 applied with $\mathbf{A} = \mathbf{B}^T\mathbf{B}$, $\eta = \frac{\lambda_{\max}}{q}$, $\delta_k = O\left(\frac{\Delta}{n^{3/2}k}\lambda_{\max}^q\right)$, and $C = \lambda_{\max}^q$ shows that a computer with $B = \Omega\left(\log\left(\frac{nk\lambda_{\max}}{\Delta\eta}\right)\right) = \Omega\left(\log \frac{n}{\Delta}\right)$ bits of precision can compute $\mathbf{y}$ satisfying:

$$\|(\mathbf{B}^T\mathbf{B})^q \mathbf{z} - \mathbf{y}\| \leq \frac{\Delta \lambda_{\max}^q}{4n^{3/2}} \|\mathbf{z}\| \leq \frac{\Delta \lambda_{\max}^q}{4n}, \tag{69}$$

since $\|\mathbf{z}\| = \sqrt{n}$. We note that we can multiply by $\mathbf{B}^T\mathbf{B}$ accurately, so Requirement 2 holds as required by Theorem 1. Specifically, it is easy to check that if Requirement 2 holds for $\mathbf{B}$ with precision 2^{-cB} for some constant c, it holds with precision $O(2^{-cB})$ for $\mathbf{B}^T\mathbf{B}$.

Combining (69) with the bound of Theorem 19 that $\|\hat{\mathbf{y}}\| \geq \frac{\lambda_{\max}^q}{n}$ we first have:

$$\|\hat{\mathbf{y}} - \mathbf{y}\| \leq \frac{\Delta}{4}\|\hat{\mathbf{y}}\|.$$

It follows that

$$\begin{aligned}
\frac{\|\mathbf{B}\mathbf{y}\|}{\|\mathbf{y}\|} &\geq \frac{\|\mathbf{B}\hat{\mathbf{y}}\| - \frac{\Delta}{4}\|\mathbf{B}\|\|\hat{\mathbf{y}}\|}{(1+\frac{\Delta}{4})\|\hat{\mathbf{y}}\|} \\
&\geq \left(1 - \frac{\Delta}{4}\right)\frac{\|\mathbf{B}\hat{\mathbf{y}}\|}{\|\hat{\mathbf{y}}\|} - \frac{\Delta}{4}\|\mathbf{B}\| \\
&\geq (1-\Delta)\,\|\mathbf{B}\|,
\end{aligned}$$

where the last step follows from Theorem 19's claim that $\frac{\|\mathbf{B}\hat{\mathbf{y}}\|}{\|\hat{\mathbf{y}}\|} \geq \left(1 - \frac{\Delta}{2}\right)\|\mathbf{B}\|$. This proves the theorem, with runtime bounds following from Theorem 1. $\square$

Remark. As discussed in Section 5, computations in Algorithm 1 won't overflow or lose accuracy due to underflow as long as the exponent in our floating point system has at $\Omega(\log\log(knC))$ bits. This is typically a very mild assumption. However, in Theorem 18, $C = \lambda_{\max}(\mathbf{B}^T\mathbf{B})^q$ so we need $O(\log q + \log\log\|\mathbf{B}\|) = \Omega(\log\frac{1}{\Delta})$ bits for our exponent. This may not be a reasonable assumption for some computers – we might want $\Delta \approx \epsilon_{\text{mach}}$ and in typical floating point systems fewer bits are allocated for the exponent than for the significand. This issue can be avoided in a number of ways. One simple approach is to instead apply $f(x) = \frac{1}{\lambda_{\max}(\mathbf{T})^q}x^q$, which also satisfies the guarantees of Lemma 19. Doing so avoids overflow or problematic underflow in Algorithm 1 as long as we have $\Omega(\log\log(kn))$ exponent bits. It could lead to underflow when applying $f(x)$ to $\mathbf{T}$'s smaller eigenvalues, but this won't affect the outcome of the theorem – as discussed in Section 5 the tiny additive error incurred from underflow when applying $f(\mathbf{T})$ is swamped by multiplicative error terms.

8.4 Generic polynomial acceleration

Our applications to the matrix step function and to approximating the top singular vector in Sections 8.1 and 8.3 share a common approach: in both cases, Lanczos is used to apply a function that is *itself* a polynomial, one that is simple to describe and evaluate, but has high degree. We then claim that this polynomial can be approximated by a lower degree polynomial, and the number of iterations required by Lanczos depends on this lower degree. In both cases, it is possible to improve a degree q polynomial to degree roughly $\sqrt{q}$ – a significant gain for the applications. To use the common term from convex optimization, Lanczos provides a way of "accelerating" the computation of some high degree matrix polynomials.

In fact, it turns out that *any* degree q polynomial with bounded coefficients in the monomial basis (or related simple bases) can be accelerated in a similar way to our two examples. To see this, we begin with the following result of [FMMS16]:

Lemma 20 (Polynomial Acceleration)**.** *Let p be an $O(k)$ degree polynomial that can be written as*

$$p(x) = \sum_{i=0}^{k} f_i(x)\,(g_i(x))^i$$

where $f_i(x)$ and $g_i(x)$ are $O(1)$ degree polynomials satisfying $|f_i(x)| \leq a_i$ and $|g_i(x)| \leq 1$ for all $x \in [-1, 1]$. Then, there exists polynomial $q(x)$ of degree $O(\sqrt{k \log(A/\epsilon)})$ where $A = \sum_{i=0}^{k} a_i$ such that $|p(x) - q(x)| \leq \epsilon$ for all $x \in [-1, 1]$.

Setting $g_i(x) = x$ and $f_i(x) = c_i$ for example, lets us accelerate any degree k polynomial $p(x) = c_0 + c_1 x + \ldots + c_k x^k$ with bounded coefficients. Lemma 20 yields the following result, which generalizes Theorems 14 and 18:

Theorem 21 (Application of Accelerated Polynomial)**.** *Consider $\mathbf{A} \in \mathbb{R}^{n\times n}$ with $\|\mathbf{A}\| \leq 1$, $\mathbf{x} \in \mathbb{R}^n$, $\epsilon \leq 1$, and any degree $O(k)$ polynomial $p(x)$ which can be written as in Lemma 20. Let $B = \log\left(\frac{nk}{\epsilon}\right)$. If Algorithm 1 is run with $f(x) = p(x)$ for $q = O(\sqrt{k \log(kA/\epsilon)})$ iterations on a computer satisfying Requirements 1 and 2 for precision $\epsilon_{\text{mach}} = 2^{-\Omega(B)}$ (e.g. a computer using $\Omega(B)$ bits of precision), it outputs $\mathbf{y}$ satisfying $\|p(\mathbf{A})\mathbf{x} - \mathbf{y}\| \leq \epsilon A \|\mathbf{x}\|$. The total runtime is $O(\text{mv}(\mathbf{A})q + q^2 B + qB^2)$.*

Proof. The proof follows from Theorem 1. Since $p(x)$ can be written as in Lemma 20, it is not hard to see that $|p(x)| \leq A$ for $x \in [-1, 1]$. If we set $\eta = \Theta\left(\min(\|\mathbf{A}\|, \frac{1}{k^2})\right)$ we can also claim that $|p(x)| = O(A)$ on $[-1-\eta, 1+\eta] \supseteq [\lambda_{\min}(\mathbf{A}) - \eta, \lambda_{\max}(\mathbf{A}) + \eta]$ (we bound $\eta \leq \|\mathbf{A}\|$ to satisfy the requirement of Theorem 1). This is a consequence of the Markov Brother's inequality. Let $p(x)$ have degree ck and choose $\eta < \frac{1}{2c^2k^2}$. Suppose, for the sake of contradiction that $p(x) \geq 2A$ for some $x \in [1, 1+\eta]$. Then $p(x)$ must have derivative $> 2Ac^2k^2$ somewhere in $[1, x]$. This would contradict the Markov Brother's inequality. An identical bound can be shown for the range $[-1-\eta, -1]$, overall allowing us to set $C = O(A)$ in applying Theorem 1.

By Lemma 20 there is an $O(\sqrt{k \log(kA/\epsilon)})$ polynomial $q(x)$ with $|p(x) - q(x)| = O(\epsilon/k)$ for all $x \in [-1, 1] \supseteq [\lambda_{\min}(\mathbf{A}), \lambda_{\max}(\mathbf{A})]$. We need to extend this approximation guarantee to all $x \in [-1-\eta, 1+\eta]$. To do so, we first note that $p(x) - q(x)$ is an $O(k)$ degree polynomial – we can assume that q has degree at most that of p or else we can just set $q(x) = p(x)$ achieving $\delta_k = 0$. Then, again by using the Markov brother's inequality, since $\eta = O(1/k^2)$ we have $|p(x) - q(x)| = O(\epsilon/k)$ for all $x \in [-1-\eta, 1+\eta]$. We can thus apply Theorem 1 with $\delta_k = O(\epsilon/k)$, giving the result. □

9 Conclusions and future work

In this work we study the stability of the Lanczos method for approximating general matrix functions and solving positive definite linear systems. We show that the method's performance for general function approximation in finite arithmetic essentially matches the strongest known exact arithmetic bounds. At the same time, we show that for linear systems, known techniques give finite precision bounds which are much weaker than what is known in exact arithmetic.

The most obvious question we leave open is understanding if our lower bound against Greenbaum's results for approximating $\mathbf{A}^{-1}\mathbf{x}$ in fact gives a lower bound on the number of iterations required by the Lanczos and CG algorithms. Alternatively, it is possible that an improved analysis could lead to stronger error bounds for finite precision Lanczos that actually match the guarantees available in exact arithmetic. It seems likely that such an analysis would have to go beyond the view of Lanczos as applying a single near optimal approximating polynomial, and thus could provide significant new insight into the behavior of the algorithm.

Understanding whether finite precision Lanczos can match the performance of non-uniform approximating polynomials is also interesting beyond the case of positive definite linear systems. For a number of other functions, it is possible to prove stronger bounds than Theorem 6 in exact arithmetic. In some of these cases, including for the matrix exponential, such results can be extended to finite precision in an analogous way to Greenbaum's work on linear systems [GS94, DGK98]. It would be interesting to explore the strength of these bounds for functions besides $1/x$.

Finally, investigating the stability of Lanczos method for other tasks besides of the widely studied problem of eigenvector computation would be interesting. Block variants of Lanczos, or Lanczos with reorthogonalization, have recently been used to give state-of-the-art runtimes for low-rank matrix approximation [RST09, MM15]. The analysis of these methods relies on the ability of Lanczos to apply optimal approximating polynomials and understanding the stability of this analysis is an interesting question. It has already been addressed for the closely related but slower block power method [HP14, BDWY16].

Acknowledgements

We would like to thank Roy Frostig for valuable guidance at all stages of this work and for his close collaboration on the projects that lead to this research. We would also like to thank Michael Cohen for the initial idea behind the lower bound proof and Jon Kelner and Richard Peng for a number of helpful conversations.

References

[AHK05] Sanjeev Arora, Elad Hazan, and Satyen Kale. Fast algorithms for approximate semidefinite programming using the multiplicative weights update method. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 339–348, 2005.

[AK07] Sanjeev Arora and Satyen Kale. A combinatorial, primal-dual approach to semidefinite programs. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 227–236, 2007.

[AL86] Owe Axelsson and Gunhild Lindskog. On the rate of convergence of the preconditioned conjugate gradient method. *Numerische Mathematik*, 48(5):499–523, 1986.

[AZL16] Zeyuan Allen-Zhu and Yuanzhi Li. Lazysvd: Even faster svd decomposition yet without agonizing pain. In *Advances in Neural Information Processing Systems 29 (NIPS)*, pages 974–982. 2016.

[AZL17] Zeyuan Allen-Zhu and Yuanzhi Li. Faster principal component regression and stable matrix Chebyshev approximation. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

[BDWY16] Maria-Florina Balcan, Simon Shaolei Du, Yining Wang, and Adams Wei Yu. An improved gap-dependency analysis of the noisy power method. In *Proceedings of the 29th Annual Conference on Computational Learning Theory (COLT)*, volume 49, pages 284–309, 2016.

[BNS78] James R. Bunch, Christopher P. Nielsen, and Danny C. Sorensen. Rank-one modification of the symmetric eigenproblem. *Numerische Mathematik*, 31(1):31–48, 1978.

[CKP+16] Michael B. Cohen, Jonathan Kelner, John Peebles, Richard Peng, Aaron Sidford, and Adrian Vladu. Faster algorithms for computing the stationary distribution, simulating random walks, and more. In *Proceedings of the 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 583–592, 2016.

[Cle55] C. W. Clenshaw. A note on the summation of chebyshev series. *Mathematics of Computation*, 9(51):118, 1955.

[DGK98] Vladimir Druskin, Anne Greenbaum, and Leonid Knizhnerman. Using nonorthogonal Lanczos vectors in the computation of matrix functions. *SIAM Journal on Scientific Computing*, 19(1):38–54, 1998.

[DH07] John Dunagan and Nicholas J. A. Harvey. Iteratively constructing preconditioners via the conjugate gradient method. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 207–216, 2007.

[DK91] Vladimir Druskin and Leonid Knizhnerman. Error bounds in the simple Lanczos procedure for computing functions of symmetric matrices and eigenvalues. *U.S.S.R. Comput. Math. Math. Phys.*, 31(7):970–983, 1991.

[DK95] Vladimir Druskin and Leonid Knizhnerman. Krylov subspace approximation of eigenpairs and matrix functions in exact and computer arithmetic. *Numerical Linear Algebra with Applications*, 2(3):205–217, 1995.

[DNPS16] Edoardo Di Napoli, Eric Polizzi, and Yousef Saad. Efficient estimation of eigenvalue counts in an interval. *Numerical Linear Algebra with Applications*, 2016.

[DPSX17] Kevin Deweese, Richard Peng, Serban A. Stan, and Hao Ran Xu. Evaluating the precision of tree PCG for graph laplacians. Technical report, 2017.

[FMMS16] Roy Frostig, Cameron Musco, Christopher Musco, and Aaron Sidford. Principal component projection without principal component analysis. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 2349–2357, 2016.

[GE95] Ming Gu and Stanley C. Eisenstat. A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem. *SIAM Journal on Matrix Analysis and Applications*, 16(1):172–191, 1995.

[GR87] Leslie Greengard and Vladimir Rokhlin. A fast algorithm for particle simulations. *Journal of computational physics*, 73(2):325–348, 1987.

[Gre89] Anne Greenbaum. Behavior of slightly perturbed Lanczos and conjugate-gradient recurrences. *Linear Algebra and its Applications*, 113:7–63, 1989.

[GS94] Gene H. Golub and Zdeněk Strakoš. Estimates in quadratic formulas. *Numerical Algorithms*, 8(2):241–268, 1994.

[GST07] A. Gil, J. Segura, and N. Temme. *Numerical Methods for Special Functions.* Society for Industrial and Applied Mathematics, 2007.

[Hig02] Nicholas J. Higham. *Accuracy and stability of numerical algorithms.* SIAM, 2002.

[Hig08] Nicholas J. Higham. *Functions of Matrices.* Society for Industrial and Applied Mathematics, 2008.

[HMAS17] Insu Han, Dmitry Malioutov, Haim Avron, and Jinwoo Shin. Approximating the spectral sums of large-scale matrices using stochastic Chebyshev approximations. *SIAM Journal on Scientific Computing*, 2017.

[HMS15] Insu Han, Dmitry Malioutov, and Jinwoo Shin. Large-scale log-determinant computation through stochastic Chebyshev expansions. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 908–917, 2015.

[HP14] Moritz Hardt and Eric Price. The noisy power method: A meta algorithm with applications. In *Advances in Neural Information Processing Systems 27 (NIPS)*, pages 2861–2869. 2014.

[HS52] Magnus R Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6), 1952.

[Hut90] Michael F. Hutchinson. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 19(2):433–450, 1990.

[IEE08] IEEE standard for floating-point arithmetic. *IEEE Std 754-2008*, pages 1–70, 2008.

[Kal07] Satyen Kale. *Efficient algorithms using the multiplicative weights update method*. PhD thesis, 2007.

[Lan50] Cornelius Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, 45(4), 1950.

[Mar90] A. Markov. On a question by D.I. Mendeleev. *Zap. Imp. Akad. Nauk*, 62, 1890.

[ML03] Cleve Moler and Charles Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, 45(1):3–49, 2003.

[MM15] Cameron Musco and Christopher Musco. Randomized block Krylov methods for stronger and faster approximate singular value decomposition. In *Advances in Neural Information Processing Systems 28 (NIPS)*, pages 1396–1404, 2015.

[MNS+17] Cameron Musco, Praneeth Netrapalli, Aaron Sidford, Shashanka Ubaru, and David P Woodruff. Spectrum approximation beyond fast matrix multiplication: Algorithms and hardness. *arXiv:1704.04163*, 2017.

[MS06] Gérard Meurant and Zdeněk Strakoš. The Lanczos and conjugate gradient algorithms in finite precision arithmetic. *Acta Numerica*, 15:471–542, 2006.

[OSV12] Lorenzo Orecchia, Sushant Sachdeva, and Nisheeth K. Vishnoi. Approximating the exponential, the Lanczos method and an $\tilde{\mathrm{O}}(\mathrm{m})$-time spectral algorithm for balanced separator. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1141–1160, 2012.

[Pai71] Christopher C. Paige. *The computation of eigenvalues and eigenvectors of very large sparse matrices.* PhD thesis, University of London, 1971.

[Pai76] Christopher C. Paige. Error analysis of the Lanczos algorithm for tridiagonalizing a symmetric matrix. *IMA Journal of Applied Mathematics*, 18(3):341–349, 1976.

[Pai80] Christopher C. Paige. Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem. *Linear Algebra and its Applications*, 34:235–258, 1980.

[Par98] Beresford N. Parlett. *The symmetric eigenvalue problem.* SIAM, 1998.

[PS79] Beresford N. Parlett and David S. Scott. The Lanczos algorithm with selective orthogonalization. *Mathematics of Computation*, 33(145):217–238, 1979.

[RST09] Vladimir Rokhlin, Arthur Szlam, and Mark Tygert. A randomized algorithm for principal component analysis. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1100–1124, 2009.

[Saa92] Yousef Saad. Analysis of some Krylov subspace approximations to the matrix exponential operator. *SIAM Journal on Numerical Analysis*, 29(1):209–228, 1992.

[Saa03] Yousef Saad. *Iterative methods for sparse linear systems.* SIAM, 2003.

[Saa11] Yousef Saad. *Numerical Methods for Large Eigenvalue Problems: Revised Edition.* SIAM, 2011.

[Sim84] Horst D. Simon. The Lanczos algorithm with partial reorthogonalization. *Mathematics of Computation*, 42(165):115–142, 1984.

[SV14] Sushant Sachdeva and Nisheeth K Vishnoi. Faster algorithms via approximation theory. *Foundations and Trends in Theoretical Computer Science*, 9(2):125–210, 2014.

[SW09] Daniel A. Spielman and Jaeoh Woo. A note on preconditioning by low-stretch spanning trees. *arXiv preprint arXiv:0903.2816*, 2009.

[TPGV16] Nicolas Tremblay, Gilles Puy, Rémi Gribonval, and Pierre Vandergheynst. Compressive spectral clustering. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 20–22, 2016.

[US16] Shashanka Ubaru and Yousef Saad. Fast methods for estimating the numerical rank of large matrices. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 468–477, 2016.

[Wil65] James H. Wilkinson. *Rounding Errors in Algebraic Processes.* 1965.

A Stability of post-processing for Lanczos

In this section, we show that the final step in Algorithm 1, computing $\mathbf{Q}f(\mathbf{T})\mathbf{e}_1$, can be performed stably in $\tilde{O}(k^2)$ time since $\mathbf{T}$ is a $k \times k$ symmetric tridiagonal matrix. This claim relies on a $\tilde{O}(k^2)$ time backwards stable algorithm for computing a tridiagonal eigendecomposition, which was developed by Gu and Eisenstat [GE95]. Given an accurate eigendecomposition of $\mathbf{T}$, we can explicitly compute an approximation to $f(\mathbf{T})$ and thus to $\mathbf{Q}f(\mathbf{T})\mathbf{e}_1$. Of course, since small error in computing the eigendecomposition can be amplified, this technique only gives an accurate result when f is sufficiently smooth. In particular, we will show that as long as $f(x)$ is well approximated by a degree k polynomial, then $f(\mathbf{T})$ can be applied stably. This characterization of smoothness is convenient because our accuracy bounds for Lanczos already depend on the degree to which $f(x)$ can be approximated by a polynomial.

A.1 Stable symmetric tridiagonal eigendecomposition

We first characterize the performance of Gu and Eisenstat's divide-and-conquer eigendecomposition algorithm for symmetric tridiagonal $\mathbf{T}$. We work through the error analysis carefully here, however we refer readers to [GE95] for a full discussion of the computations involved.

Lemma 22 (Divide-and-Conquer Algorithm of [GE95])**.** *Given symmetric tridiagonal $\mathbf{T} \in \mathbb{R}^{k\times k}$ and error parameter ϵ with $ck^3 \log k \cdot \epsilon_{\text{mach}} \leq \epsilon \leq 1/2$ for fixed constant c, there is an algorithm running in $O(k^2 \log \frac{k}{\epsilon} + k \log^2 \frac{k}{\epsilon})$ time on a computer satisfying Requirements 1 and 2 with machine precision ϵ_{mach} which outputs $\tilde{\mathbf{V}} \in \mathbb{R}^{k\times k}$ and diagonal $\tilde{\mathbf{\Lambda}} \in \mathbb{R}^{k\times k}$ satisfying:*

$$\|\tilde{\mathbf{V}}\tilde{\mathbf{\Lambda}}\tilde{\mathbf{V}}^T - \mathbf{T}\| \leq \epsilon\|\mathbf{T}\| \text{ and } \|\tilde{\mathbf{V}}^T\tilde{\mathbf{V}} - \mathbf{I}\| \leq \epsilon.$$

Proof. The algorithm of [GE95] is recursive, partitioning $\mathbf{T}$ into two blocks $\mathbf{T}_1 \in m \times m$ and $\mathbf{T}_2 \in (N-m-1)\times(N-m-1)$ where $m = \lfloor k/2 \rfloor$ such that:

$$\mathbf{T} = \begin{bmatrix} \mathbf{T}_1 & \beta_{m+1}\mathbf{e}_m & \mathbf{0} \\ \beta_{m+1}\mathbf{e}_m^T & \alpha_{m+1} & \beta_{m+2}\mathbf{e}_1^T \\ \mathbf{0} & \beta_{m+2}\mathbf{e}_1 & \mathbf{T}_2 \end{bmatrix}.$$

Note that α_i, β_j are the corresponding entries of $\mathbf{T}$ in the notation of Algorithm 1. Let $\mathbf{T}_i = \mathbf{V}_i\mathbf{\Lambda}_i\mathbf{V}_i$ be the eigendecomposition of $\mathbf{T}_i$ for $i = 1, 2$. We can see that $\mathbf{T} = \mathbf{Z}\mathbf{H}\mathbf{Z}^T$ where:

$$\mathbf{H} = \begin{bmatrix} \alpha_{m+1} & \beta_{m+1}\mathbf{l}_1^T & \beta_{m+2}\mathbf{f}_2^T \\ \beta_{m+1}\mathbf{l}_1 & \mathbf{\Lambda_1} & \mathbf{0} \\ \beta_{m+2}\mathbf{f}_2 & \mathbf{0} & \mathbf{\Lambda}_2 cycle \end{bmatrix} \quad \text{and} \quad \mathbf{Z} = \begin{bmatrix} \mathbf{0} & \mathbf{V}_1 & \mathbf{0} \\ 1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{V}_2 \end{bmatrix}.$$

Here $\mathbf{l}_1^T$ is the last row of $\mathbf{V}_1$ and $\mathbf{f}_2^T$ is the first row of $\mathbf{V}_2$. $\mathbf{H}$ is a symmetric arrowhead matrix and a primary contribution of [GE95] is showing that it can be eigendecomposed stably in $\tilde{O}(k^2)$ time. Writing the eigendecomposition $\mathbf{H} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, the eigendecomposition of $\mathbf{T}$ is then given by $\mathbf{T} = \mathbf{Z}\mathbf{U}\mathbf{\Lambda}\mathbf{U}^T\mathbf{Z}^T$.

We now proceed with the error analysis of this method. Assume by induction that for $\mathbf{T}_i$ we compute an approximate eigendecomposition $\tilde{\mathbf{V}}_i\tilde{\mathbf{\Lambda}}_i\tilde{\mathbf{V}}_i^T$ satisfying:

$$\tilde{\mathbf{V}}_i\tilde{\mathbf{\Lambda}}_i\tilde{\mathbf{V}}_i^T = \tilde{\mathbf{T}}_i \text{ where } \|\mathbf{T}_i - \tilde{\mathbf{T}}_i\| \leq \delta_T \text{ and } \|\tilde{\mathbf{V}}_i^T\tilde{\mathbf{V}}_i - \mathbf{I}\| \leq \delta_I. \tag{70}$$

In the base case, $\mathbf{T}_i$ is just a single entry and so (70) holds trivially for $\delta_T = \delta_I = 0$. We define:

$$\tilde{\mathbf{T}} = \begin{bmatrix} \tilde{\mathbf{T}}_1 & \beta_{m+1}\mathbf{e}_m & \mathbf{0} \\ \beta_{m+1}\mathbf{e}_m^T & \alpha_{m+1} & \beta_{m+2}\mathbf{e}_1^T \\ \mathbf{0} & \beta_{m+2}\mathbf{e}_1 & \tilde{\mathbf{T}}_2 \end{bmatrix}$$

and have $\|\mathbf{T} - \tilde{\mathbf{T}}\| \leq \delta_T$ by (70) since $\mathbf{T} - \tilde{\mathbf{T}}$ is block diagonal with blocks $\mathbf{T}_i - \tilde{\mathbf{T}}_i$. Define:

$$\tilde{\mathbf{H}}_{\text{exact}} = \begin{bmatrix} \alpha_{m+1} & \beta_{m+1}\tilde{\mathbf{l}}_1^T & \beta_{m+2}\tilde{\mathbf{f}}_2^T \\ \beta_{m+1}\tilde{\mathbf{l}}_1 & \tilde{\mathbf{\Lambda}}_1 & \mathbf{0} \\ \beta_{m+2}\tilde{\mathbf{f}}_2 & \mathbf{0} & \tilde{\mathbf{\Lambda}}_2 \end{bmatrix} \qquad \text{and} \qquad \tilde{\mathbf{Z}} = \begin{bmatrix} \mathbf{0} & \tilde{\mathbf{V}}_1 & \mathbf{0} \\ 1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \tilde{\mathbf{V}}_2 \end{bmatrix}$$

where $\tilde{\mathbf{l}}_1$ and $\tilde{\mathbf{f}}_2$ are the last and first rows of $\tilde{\mathbf{V}}_1$ and $\tilde{\mathbf{V}}_2$ respectively. Let $\tilde{\mathbf{H}} = \text{fl}(\tilde{\mathbf{H}}_{\text{exact}})$ be the result of computing $\tilde{\mathbf{H}}_{\text{exact}}$ in finite precision. By inspection we see that:

$$\tilde{\mathbf{T}} - \tilde{\mathbf{Z}}\tilde{\mathbf{H}}_{\text{exact}}\tilde{\mathbf{Z}}^T = \begin{bmatrix} \mathbf{0} & \beta_{m+1}(\mathbf{e}_m - \tilde{\mathbf{V}}_1\tilde{\mathbf{l}}_1) & \mathbf{0} \\ \beta_{m+1}(\mathbf{e}_m - \tilde{\mathbf{V}}_1\tilde{\mathbf{l}}_1)^T & 0 & \beta_{m+2}(\mathbf{e}_1 - \tilde{\mathbf{V}}_2\tilde{\mathbf{f}}_2)^T \\ \mathbf{0} & \beta_{m+2}(\mathbf{e}_1 - \tilde{\mathbf{V}}_2\tilde{\mathbf{f}}_2) & \mathbf{0} \end{bmatrix}.$$

By our inductive assumption that $\|\tilde{\mathbf{V}}_i^T\tilde{\mathbf{V}}_i - \mathbf{I}\| \leq \delta_I$ (70), we have $\|\tilde{\mathbf{Z}}^T\tilde{\mathbf{Z}} - \mathbf{I}\| \leq \delta_I$ and further:

$$\begin{aligned} \|\tilde{\mathbf{T}} - \tilde{\mathbf{Z}}\tilde{\mathbf{H}}_{\text{exact}}\tilde{\mathbf{Z}}^T\| &\leq \beta_{m+1}\|\mathbf{e}_m - \tilde{\mathbf{V}}_1\tilde{\mathbf{l}}_1\| + \beta_{m+2}\|\mathbf{e}_1 - \tilde{\mathbf{V}}_2\tilde{\mathbf{f}}_2\| \\ &\leq \beta_{m+1}\|\tilde{\mathbf{V}}_1\tilde{\mathbf{V}}_1^T - \mathbf{I}\|\|\mathbf{e}_k\| + \beta_{m+2}\|\tilde{\mathbf{V}}_2\tilde{\mathbf{V}}_2^T - \mathbf{I}\|\|\mathbf{e}_1\| \\ &\leq \delta_I(\beta_{m+1} + \beta_{m+2}). \end{aligned}$$

Then, by the triangle inequality we can loosely bound

$$\begin{aligned} \|\mathbf{T} - \tilde{\mathbf{Z}}\tilde{\mathbf{H}}_{\text{exact}}\tilde{\mathbf{Z}}^T\| &\leq \|\mathbf{T} - \tilde{\mathbf{T}}\| + \|\tilde{\mathbf{T}} - \tilde{\mathbf{Z}}\tilde{\mathbf{H}}_{\text{exact}}\tilde{\mathbf{Z}}^T\| \\ &\leq \delta_T + \delta_I(\beta_{m+1} + \beta_{m+2}) \\ &\leq \delta_T + 2\delta_I\|\mathbf{T}\|. \end{aligned} \tag{71}$$

Finally, using Requirement 1, $\tilde{\mathbf{H}}$'s entries are within relative error ϵ_{mach} of the entries in $\tilde{\mathbf{H}}_{\text{exact}}$ so

$$\|\tilde{\mathbf{H}} - \tilde{\mathbf{H}}_{\text{exact}}\| \leq \|\tilde{\mathbf{H}} - \tilde{\mathbf{H}}_{\text{exact}}\|_F \leq \epsilon_{\text{mach}}\,\|\tilde{\mathbf{H}}_{\text{exact}}\|_F \leq \epsilon_{\text{mach}}\sqrt{k}\|\tilde{\mathbf{H}}_{\text{exact}}\|. \tag{72}$$

Using (71) and submultiplicativity we can obtain a loose bound on $\|\tilde{\mathbf{H}}_{\text{exact}}\|$ of

$$\begin{aligned} \|\tilde{\mathbf{H}}_{\text{exact}}\| &\leq \|\tilde{\mathbf{Z}}^{-1}\|^2\,(\|\mathbf{T}\| + \delta_T + 2\delta_I\|\mathbf{T}\|) \\ &\leq 8\|\mathbf{T}\| + 4\delta_T \end{aligned} \tag{73}$$

as long as $\delta_I \leq 1/2$ and so $\|\tilde{\mathbf{Z}}^{-1}\| \leq 2$. We finally conclude that:

$$\|\tilde{\mathbf{Z}}\tilde{\mathbf{H}}_{\text{exact}}\tilde{\mathbf{Z}}^T - \tilde{\mathbf{Z}}\tilde{\mathbf{H}}\tilde{\mathbf{Z}}^T\| \leq \|\tilde{\mathbf{Z}}\|^2\|\tilde{\mathbf{H}}_{\text{exact}} - \tilde{\mathbf{H}}\| \leq 18\sqrt{k}\,\epsilon_{\text{mach}}\,\|\mathbf{T}\| + 9\sqrt{k}\,\epsilon_{\text{mach}}\,\delta_T.$$

Combined with (71), we have that:

$$\|\mathbf{T} - \tilde{\mathbf{Z}}\tilde{\mathbf{H}}\tilde{\mathbf{Z}}^T\| \leq (1 + 9\sqrt{k}\,\epsilon_{\text{mach}})\delta_T + 2\delta_I\|\mathbf{T}\| + 18\sqrt{k}\,\epsilon_{\text{mach}}\,\|\mathbf{T}\|. \tag{74}$$

We now discuss the eigendecomposition of $\tilde{\mathbf{H}}$. Gu and Eisenstat show that for some error parameter ϵ_1 with $\epsilon_{\text{mach}} \leq \epsilon_1 \leq \frac{1}{2k}$, it is possible to compute $\widehat{\lambda}_i$ satisfying

$$\left|\widehat{\lambda}_i - \lambda_i(\tilde{\mathbf{H}})\right| \leq \epsilon_1 k\|\tilde{\mathbf{H}}\| \text{ for all } i \tag{75}$$

in $O(k \log^2(1/\epsilon_1))$ time. To do this they assume the existence of a root finder giving relative accuracy solutions to the roots of the characteristic polynomial of $\tilde{\mathbf{H}}$ (its eigenvalues). While significant work has studied efficient root finders for these polynomials (see e.g. [BNS78]), we can just use of a simple bisection method that converges to relative accuracy $(1 \pm \epsilon_1)$ in $\log(1/\epsilon_1)$ iterations, as it will not significantly affect our final asymptotic runtime bounds. The second $\log(1/\epsilon_1)$ factor in the runtime bound above comes from the use of the Fast Multipole Method [GR87] to evaluate the characteristic polynomial efficiently at each iteration of the bisection method.

Gu and Eisenstat next show that given $\widehat{\lambda}_i$ satisfying (75), and letting $\hat{\mathbf{\Lambda}} = \mathrm{diag}(\widehat{\lambda}_1, ..., \widehat{\lambda}_k)$, it is possible to compute $\hat{\mathbf{U}}$ such that $\hat{\mathbf{H}} \stackrel{\text{def}}{=} \hat{\mathbf{U}}\hat{\mathbf{\Lambda}}\hat{\mathbf{U}}^T$ approximates $\tilde{\mathbf{H}}$ up to additive error $O(\epsilon_1 k^2 \|\tilde{\mathbf{H}}\|)$ on each entry. This gives:

$$\|\tilde{\mathbf{H}} - \hat{\mathbf{H}}\| = c_1 \epsilon_1 k^3 \|\tilde{\mathbf{H}}\| \tag{76}$$

for some constant c_1. They further show that $\|\hat{\mathbf{U}}^T\hat{\mathbf{U}} - \mathbf{I}\| \leq c_2\, \epsilon_{\text{mach}}\, k^2$ for constant c_2.

Using the fast multipole method, they show how to compute $\tilde{\mathbf{V}}$ which approximates $\tilde{\mathbf{Z}}\hat{\mathbf{U}}$ to entrywise relative accuracy $\Theta(\epsilon_1)$ in $O(k^2 \log(1/\epsilon_1))$ time. Note that as long as $\delta_I \leq 2$ and $\epsilon_{\text{mach}} \leq \frac{1}{ck^2}$ for sufficiently large c, then each entry $(\tilde{\mathbf{Z}}\hat{\mathbf{U}})^{(i,j)}$ is upper bounded by a fixed constant since these matrices are both near orthogonal. Thus $\tilde{\mathbf{V}}$ actually approximates $\tilde{\mathbf{Z}}\hat{\mathbf{U}}$ up to $O(\epsilon_1)$ entrywise additive error and correspondingly $\tilde{\mathbf{V}}^T\tilde{\mathbf{V}}$ approximates $\hat{\mathbf{U}}^T\tilde{\mathbf{Z}}^T\tilde{\mathbf{Z}}\hat{\mathbf{U}}$ up to entrywise additive error $O(\epsilon_1 k)$. Thus $\|\tilde{\mathbf{V}}^T\tilde{\mathbf{V}} - \hat{\mathbf{U}}^T\tilde{\mathbf{Z}}^T\tilde{\mathbf{Z}}\hat{\mathbf{U}}\| = O(\epsilon_1 k^2)$. This gives:

$$\begin{aligned}
\|\tilde{\mathbf{V}}^T\tilde{\mathbf{V}} - \mathbf{I}\| &\leq \|\tilde{\mathbf{V}}^T\tilde{\mathbf{V}} - \hat{\mathbf{U}}^T\tilde{\mathbf{Z}}^T\tilde{\mathbf{Z}}\hat{\mathbf{U}}\| + \|\hat{\mathbf{U}}^T\tilde{\mathbf{Z}}^T\tilde{\mathbf{Z}}\hat{\mathbf{U}} - \mathbf{I}\| \\
&\leq \|\tilde{\mathbf{V}}^T\tilde{\mathbf{V}} - \hat{\mathbf{U}}^T\tilde{\mathbf{Z}}^T\tilde{\mathbf{Z}}\hat{\mathbf{U}}\| + \|\hat{\mathbf{U}}^T\hat{\mathbf{U}} - \mathbf{I}\| + \|\hat{\mathbf{U}}\|^2\|\tilde{\mathbf{Z}}^T\tilde{\mathbf{Z}} - \mathbf{I}\| \\
&\leq c_3\epsilon_1 k^2 + (1 + c_2\, \epsilon_{\text{mach}}\, k^2)\delta_I
\end{aligned} \tag{77}$$

for some constants c_2, c_3. In the last step we use that $\|\hat{\mathbf{U}}\|^2 \leq (1 + c_2\, \epsilon_{\text{mach}}\, k^2)$, since as mentioned $\|\hat{\mathbf{U}}^T\hat{\mathbf{U}} - \mathbf{I}\| \leq c_2\, \epsilon_{\text{mach}}\, k^2$.

$\tilde{\mathbf{V}}\hat{\mathbf{\Lambda}}\tilde{\mathbf{V}}^T$ approximates $\tilde{\mathbf{Z}}\hat{\mathbf{U}}\hat{\mathbf{\Lambda}}\hat{\mathbf{U}}^T\tilde{\mathbf{Z}}^T$ to entrywise additive error $O(\epsilon_1 k\|\hat{\mathbf{\Lambda}}\|) = O(\epsilon_1 k \|\tilde{\mathbf{H}}\|)$ giving:

$$\|\tilde{\mathbf{V}}\hat{\mathbf{\Lambda}}\tilde{\mathbf{V}}^T - \tilde{\mathbf{Z}}\hat{\mathbf{U}}\hat{\mathbf{\Lambda}}\hat{\mathbf{U}}^T\tilde{\mathbf{Z}}^T\| \leq c_4\epsilon_1 k^2 \|\tilde{\mathbf{H}}\| \tag{78}$$

for some constant c_4. Combining (76) and (78) with the recursive error bound (74):

$$\begin{aligned}
&\|\mathbf{T} - \tilde{\mathbf{V}}\hat{\mathbf{\Lambda}}\tilde{\mathbf{V}}^T\| \leq \|\mathbf{T} - \tilde{\mathbf{Z}}\tilde{\mathbf{H}}\tilde{\mathbf{Z}}^T\| + \|\tilde{\mathbf{Z}}\tilde{\mathbf{H}}\tilde{\mathbf{Z}}^T - \tilde{\mathbf{V}}\hat{\mathbf{\Lambda}}\tilde{\mathbf{V}}^T\| \\
&\quad \leq (1 + 9\sqrt{k}\, \epsilon_{\text{mach}})\delta_T + 2\|\mathbf{T}\|\delta_I + 18\sqrt{k}\, \epsilon_{\text{mach}}\, \|\mathbf{T}\| + \|\tilde{\mathbf{Z}}(\tilde{\mathbf{H}} - \hat{\mathbf{U}}\hat{\mathbf{\Lambda}}\hat{\mathbf{U}}^T)\tilde{\mathbf{Z}}^T\| + \|\tilde{\mathbf{Z}}\hat{\mathbf{U}}\hat{\mathbf{\Lambda}}\hat{\mathbf{U}}^T\tilde{\mathbf{Z}}^T - \tilde{\mathbf{V}}\hat{\mathbf{\Lambda}}\tilde{\mathbf{V}}^T\| \\
&\quad \leq (1 + 9\sqrt{k}\, \epsilon_{\text{mach}})\delta_T + 2\|\mathbf{T}\|\delta_I + 18\sqrt{k}\, \epsilon_{\text{mach}}\, \|\mathbf{T}\| + c_1\epsilon_1 k^3 \|\tilde{\mathbf{H}}\|\|\tilde{\mathbf{Z}}\|^2 + c_4\epsilon_1 k^2\|\tilde{\mathbf{H}}\| \\
&\quad \leq (1 + 9\sqrt{k}\, \epsilon_{\text{mach}} + c_5\epsilon_1 k^3)\delta_T + c_6(\delta_I + \epsilon_1 k^3)\|\mathbf{T}\|
\end{aligned} \tag{79}$$

for fixed constants c_5, c_6. In the last bound we use that $\|\tilde{\mathbf{H}}\| = O(\|\mathbf{T}\| + \delta_T)$, which follows from combining (72) and (73) and assuming $\epsilon_{\text{mach}} \leq \frac{1}{\sqrt{k}}$. We also use that $\|\tilde{\mathbf{Z}}\| = O(1)$. Both of these bounds hold assuming $\delta_T, \delta_I \leq 1/2$.

Finally, over $\log k$ levels of recursion, as long as we set $\epsilon_1 \leq \frac{\epsilon}{ck^3 \log k}$ for sufficiently large c, in the end, by (77) and (79) we will have $\|\tilde{\mathbf{V}}^T\tilde{\mathbf{V}} - \mathbf{I}\| \leq \epsilon$ and $\|\mathbf{T} - \tilde{\mathbf{V}}\hat{\mathbf{\Lambda}}\tilde{\mathbf{V}}^T\| \leq \epsilon\|\mathbf{T}\|$.

Specifically, by (77), δ_I just grows by a $c_3\epsilon_1 k^2$ additive factor and a $(1+c_2\, \epsilon_{\text{mach}}\, k^2)$ multiplicative factor at each level. Assuming $\epsilon_{\text{mach}} < \frac{1}{ck^2 \log k}$, the multiplicative factor, even after compounding over $\log k$ levels, can be bounded by $(1 + c_2\,\epsilon_{\text{mach}}\,k^2)^{\log k} \leq 2$ if c is set large enough. Along with the the setting of ϵ_1, this ensures that $\delta_I \leq \frac{2c_3\epsilon}{ck}$ at each level. Thus $\delta_I \leq \epsilon$ if c is set large enough.

Similarly, δ_T grows according to (79), increasing by an additive factor of $c_6(\delta_I + \epsilon_1 k^3)\|\mathbf{T}\|$ and multiplicative factor of $(1+9\sqrt{k}\,\epsilon_{\text{mach}} + c_5\epsilon_1 k^3)$. By our setting of ϵ_1 and assuming $\epsilon_{\text{mach}} < \frac{1}{c\sqrt{k}\log k}$, the multiplicative factor can be bounded by $(1+\frac{9+c_5}{c\log k})^{\log k} \leq 2$ if we set c large enough. Additionally, the additive factor can be upper bounded by $c_6\left(\frac{2c_3\epsilon}{ck} + \frac{\epsilon}{c\log k}\right)\|\mathbf{T}\|$ which is less than $\frac{\epsilon}{\log k}\|\mathbf{T}\|$ by any constant factor if c is set large enough. Thus, even when accounting for the multiplicative error and summing over $\log k$ levels, we have $\delta_T \leq \epsilon\|\mathbf{T}\|$.

Our final runtime bound follows from adding the $O(k^2\log(1/\epsilon_1))$ cost of computing $\tilde{\mathbf{V}}$ to the $O(k\log^2(1/\epsilon_1))$ cost of computing $\hat{\mathbf{\Lambda}}$ and setting $\epsilon_1 = \frac{\epsilon}{ck^3\log k}$ so $\log(1/\epsilon_1) = O(\log(k/\epsilon))$. $\square$

A.2 Stable function application

With Lemma 22 in place we can complete the analysis of the Lanczos post processing step.

Lemma 23 (Stable Post-Processing)**.** *Suppose we are given a symmetric tridiagonal* $\mathbf{T} \in \mathbb{R}^{k\times k}$, $\mathbf{x} \in \mathbb{R}^k$ *with* $\|\mathbf{x}\| = 1$[5]*, function* f, $\eta \geq 0$, *and error parameter* $\epsilon \leq 1/2$ *with* $ck^3\log k \cdot \epsilon_{\text{mach}} \leq \epsilon \leq \frac{\eta}{4\|\mathbf{T}\|}$ *for sufficiently large constant* c. *Define* $C = \max_{x\in[\lambda_{\min}(\mathbf{T})-\eta,\lambda_{\max}(\mathbf{T})+\eta]}|f(x)|$ *and let*

$$\delta_q = \min_{\substack{\text{polynomial } p\\ \text{with degree} < q}}\left(\max_{x\in[\lambda_{\min}(\mathbf{T})-\eta,\lambda_{\max}(\mathbf{T})+\eta]}|f(x)-p(x)|\right).$$

There is an algorithm running in $O(k^2\log\frac{k}{\epsilon} + k\log^2\frac{k}{\epsilon})$ *time on a computer satisfying Requirements 1 and 2 with relative precision* ϵ_{mach} *which returns* $\mathbf{y}$ *satisfying:*

$$\|f(\mathbf{T})\mathbf{x} - \mathbf{y}\| \leq 2\delta_q + \epsilon\cdot\left(\frac{8q^3C\|\mathbf{T}\|}{\eta} + 16C\right).$$

Proof. Assume that we have $\tilde{\mathbf{V}}, \tilde{\mathbf{\Lambda}}$ satisfying the guarantees of Lemma 22 for error ϵ. Let $\tilde{\mathbf{T}} = \tilde{\mathbf{V}}\tilde{\mathbf{\Lambda}}\tilde{\mathbf{V}}^T$ and let $\mathbf{y} = \mathrm{fl}(\tilde{\mathbf{V}}f(\tilde{\mathbf{\Lambda}})\tilde{\mathbf{V}}^T\mathbf{x})$ be the result of computing $\tilde{\mathbf{V}}f(\tilde{\mathbf{\Lambda}})\tilde{\mathbf{V}}^T\mathbf{x}$ in finite precision.

We introduce an orthonormal matrix $\bar{\mathbf{V}}$ such that $\|\tilde{\mathbf{V}} - \bar{\mathbf{V}}\| \leq \epsilon$. To see that such a $\bar{\mathbf{V}}$ exists, by the condition of Lemma 22 we can write $\tilde{\mathbf{V}}^T\tilde{\mathbf{V}} = \mathbf{I} + \mathbf{\Delta}$ for some symmetric $\mathbf{\Delta}$ with $\|\mathbf{\Delta}\| \leq \epsilon$. Writing the eigendecomposition $\mathbf{\Delta} = \mathbf{Z}\mathbf{S}\mathbf{Z}^T$, we have $\tilde{\mathbf{V}}^T\tilde{\mathbf{V}} = \mathbf{Z}(\mathbf{I}+\mathbf{S})\mathbf{Z}^T$. So for some orthonormal matrix $\mathbf{M}$, $\tilde{\mathbf{V}} = \mathbf{M}(\mathbf{I}+\mathbf{S})^{1/2}\mathbf{Z}^T$ and thus:

$$\tilde{\mathbf{V}} = \mathbf{M}\mathbf{Z}^T + \mathbf{M}\hat{\mathbf{S}}\mathbf{Z}^T$$

where $\|\hat{\mathbf{S}}\| \leq \epsilon$. We can then define the orthonormal matrix $\bar{\mathbf{V}} \overset{\text{def}}{=} \mathbf{M}\mathbf{Z}^T$ and have

$$\|\tilde{\mathbf{V}} - \bar{\mathbf{V}}\| \leq \epsilon. \tag{80}$$

Let $\bar{\mathbf{T}} = \bar{\mathbf{V}}\tilde{\mathbf{\Lambda}}\bar{\mathbf{V}}^T$. Using Lemma 22, we have that:

$$\begin{aligned}
\|\mathbf{T} - \bar{\mathbf{T}}\| &\leq \|\mathbf{T} - \tilde{\mathbf{T}}\| + \|\tilde{\mathbf{T}} - \bar{\mathbf{T}}\| \\
&\leq \epsilon\|\mathbf{T}\| + \|\tilde{\mathbf{V}}\tilde{\mathbf{\Lambda}}\tilde{\mathbf{V}}^T - \bar{\mathbf{V}}\tilde{\mathbf{\Lambda}}\tilde{\mathbf{V}}^T\| + \|\bar{\mathbf{V}}\tilde{\mathbf{\Lambda}}\tilde{\mathbf{V}}^T - \bar{\mathbf{V}}\tilde{\mathbf{\Lambda}}\bar{\mathbf{V}}^T\| \\
&\leq \epsilon\|\mathbf{T}\| + (\|\tilde{\mathbf{V}}\| + 1)\|\tilde{\mathbf{V}} - \bar{\mathbf{V}}\|\|\tilde{\mathbf{\Lambda}}\| \\
&\leq 4\epsilon\|\mathbf{T}\|.
\end{aligned}$$

[5]The theorem also holds with different constant factors when $\|\mathbf{x}\| = O(1)$. We prove it for the case when the norm is exactly 1 because our broader analysis only applies it to truly unit norm vectors (i.e. to the basis vector $\mathbf{e}_1$).

We now have by triangle inequality:

$$\|f(\mathbf{T})\mathbf{x} - \mathbf{y}\| \leq \|f(\mathbf{T})\mathbf{x} - f(\bar{\mathbf{T}})\mathbf{x}\| + \|f(\bar{\mathbf{T}})\mathbf{x} - \mathbf{y}\|. \tag{81}$$

The first norm is small since $\|\mathbf{T} - \bar{\mathbf{T}}\| \leq 4\epsilon\|\mathbf{T}\| \leq \eta$. Specifically, we can apply Lemma 26 in Appendix C, which uses an argument similar to that in Lemma 9 to prove that for any p with degree $< q$ and $|p(x)| \leq C$ on $[\lambda_{\min}(\mathbf{T}) - \eta, \lambda_{\max}(\mathbf{T}) + \eta]$:

$$\|p(\mathbf{T})\mathbf{x} - p(\bar{\mathbf{T}})\mathbf{x}\| \leq \frac{2q^3 C\epsilon\|\mathbf{T} - \bar{\mathbf{T}}\|}{\eta} \leq \frac{8q^3 C\epsilon\|\mathbf{T}\|}{\eta}.$$

Using triangle inequality and again that all eigenvalues of $\bar{\mathbf{T}}$ lie in $[\lambda_{\min}(\mathbf{T}) - 4\epsilon\|\mathbf{T}\|, \lambda_{\max}(\mathbf{T}) + 4\epsilon\|\mathbf{T}\|]$ and thus in $[\lambda_{\min}(\mathbf{T}) - \eta, \lambda_{\max}(\mathbf{T}) + \eta]$ by our assumption that $\epsilon \leq \frac{\eta}{4\|\mathbf{T}\|}$ gives:

$$\begin{aligned}\|f(\mathbf{T})\mathbf{x} - f(\bar{\mathbf{T}})\mathbf{x}\| &\leq \|f(\mathbf{T})\mathbf{x} - p(\mathbf{T})\mathbf{x}\| + \|f(\bar{\mathbf{T}})\mathbf{x} - p(\bar{\mathbf{T}})\mathbf{x}\| + \|p(\mathbf{T})\mathbf{x} - p(\bar{\mathbf{T}})\mathbf{x}\| \\ &\leq 2\delta_q + \frac{8q^3 C\epsilon\|\mathbf{T}\|}{\eta}. \end{aligned} \tag{82}$$

We now bound the second term of (81): $\|f(\bar{\mathbf{T}})\mathbf{x} - \mathbf{y}\|$. First note that, but our assumption in Section 5 that f can be computed to relative error ϵ_{mach},

$$\|\operatorname{fl}(f(\tilde{\mathbf{\Lambda}})) - f(\tilde{\mathbf{\Lambda}})\| \leq \epsilon_{\text{mach}} \|f(\tilde{\mathbf{\Lambda}})\| \leq \epsilon_{\text{mach}} C \tag{83}$$

since $\tilde{\mathbf{\Lambda}}$'s eigenvalues lie in $[\lambda_{\min}(\mathbf{T}) - \eta, \lambda_{\max}(\mathbf{T}) + \eta]$. Additionally, by Requirement 2, for any square matrix $\mathbf{B}$ and vector $\mathbf{w}$, there is some matrix $\mathbf{E}_B$ such that:

$$\operatorname{fl}(\mathbf{B}\mathbf{w}) = (\mathbf{B} + \mathbf{E})\,\mathbf{w} \qquad \text{and} \qquad \|\mathbf{E}\| \leq 2k^{3/2}\,\epsilon_{\text{mach}}\,\|\mathbf{B}\|. \tag{84}$$

Accordingly, we can simply write $\mathbf{y} = \operatorname{fl}(\tilde{\mathbf{V}} f(\tilde{\mathbf{\Lambda}})\tilde{\mathbf{V}}^T\mathbf{x})$ as:

$$\mathbf{y} = \left(\bar{\mathbf{V}} + \mathbf{E}_1\right)\left(f(\tilde{\mathbf{\Lambda}}) + \mathbf{E}_2\right)\left(\bar{\mathbf{V}}^T + \mathbf{E}_3\right)\mathbf{x}.$$

We can show that $\|\mathbf{E}_1\|$ and $\|\mathbf{E}_3\|$ are upper bounded by $\epsilon + 4k^{3/2}\,\epsilon_{\text{mach}}$ using (80), (84), and the loose bound $\|\tilde{\mathbf{V}}\| \leq 2$. We can show that $\|\mathbf{E}_2\|$ is upper bounded by $4\,\epsilon_{\text{mach}}\,Ck^{3/2}$ using (83) and (84). Since $4\,\epsilon_{\text{mach}}\,k^{3/2} \leq \epsilon$, $\|\mathbf{E}_1\|, \|\mathbf{E}_3\| \leq 2\epsilon$ and $\|\mathbf{E}_2\| \leq \epsilon C$. Using that $\epsilon \leq 1/2$, it follows that:

$$\begin{aligned}\|f(\bar{\mathbf{T}})\mathbf{x} - \mathbf{y}\| &= \|\bar{\mathbf{V}} f(\tilde{\mathbf{\Lambda}})\bar{\mathbf{V}}^T\mathbf{x} - \mathbf{y}\| \\ &\leq \|\mathbf{E}_1\|(C + \epsilon C)(1 + 2\epsilon) + \|\mathbf{E}_3\|(C + \epsilon C)(1 + 2\epsilon) + \|\mathbf{E}_2\|(1 + 2\epsilon)^2 \\ &\leq 16\epsilon C.\end{aligned}$$

Plugging this bound and (82) into (81) gives the lemma. □

B Tighter results for linear systems

In this section we discuss how bounds on function approximation via Lanczos can be improved for the special case of $f(\mathbf{A}) = \mathbf{A}^{-1}$ when $\mathbf{A}$ is positive definite, both in exact arithmetic and finite precision. In particular, we provide a short proof of the exact arithmetic bound presented in (3) and discuss Greenbaum's analogous result for finite precision conjugate gradient (Theorem 2) in full detail. Ultimately, our lower bound in Section 7 shows that, while Theorem 2 is a natural extension of (3) to finite precision, it actually gives much weaker iteration bounds.

One topic which we do not discuss in depth is that, besides tighter approximation bounds, the computational cost of the Lanczos method can be somewhat improved when solving linear systems. Specifically, it is possible to compute the approximation $\mathbf{y} = \|\mathbf{x}\|\mathbf{Q}\mathbf{T}^{-1}\mathbf{e}_1$ "on-the-fly", without explicitly storing $\mathbf{Q}$ or $\mathbf{T}$. While this does not improve on the asymptotic runtime complexity of Algorithm 1, it improves the space complexity from $O(kn)$ to simply order $O(n)$.

Such space-optimized methods yield the popular conjugate gradient algorithm (CG) and its relatives. In fact, Greenbaum's analysis applies to a variant of CG (Algorithm 2). Like all variants, it computes an approximation to $\mathbf{A}^{-1}\mathbf{x}$ that, at least in exact arithmetic, is equivalent to $\|\mathbf{x}\|\mathbf{Q}\mathbf{T}^{-1}\mathbf{e}_1$, the approximation obtained from the Lanczos method (Algorithm 1). The finite precision behavior of Greenbaum's conjugate gradient implementation is also very similar to the finite precision behavior of the Lanczos method we study. In fact, her work is based on the same basic results of Paige that we depend on in Section 6.

B.1 Linear systems in exact arithmetic

We begin by proving (3), showing that the approximation quality of Lanczos in exact arithmetic (Theorem 6) can be improved when our goal is to approximate $\mathbf{A}^{-1}\mathbf{x}$ for positive definite $\mathbf{A}$. It is not hard to see that an identical bound holds when $\mathbf{A}$ is positive semidefinite (i.e. may be singular) and $f(\mathbf{A}) = \mathbf{A}^{+}$ is the pseudoinverse. That is, $f(x) = 1/x$ for $x > 0$ and 0 for $x = 0$. However, we restrict our attention to full rank matrices for simplicity, and since it is for these matrices which Greenbaum's finite precision bounds hold.

Theorem 24 – Exact Arithmetic (Approximate Application of $\mathbf{A}^{-1}$)**.** *Suppose $\mathbf{Q} \in \mathbb{R}^{n\times k}$, $\mathbf{T} \in \mathbb{R}^{k\times k}$, β_{k+1}, and $\mathbf{q}_{k+1}$ are computed by the Lanczos algorithm (Algorithm 1), run with exact arithmetic on positive definite $\mathbf{A} \in \mathbb{R}^{n\times n}$ and $\mathbf{x} \in \mathbb{R}^n$ for $k \leq n$ iterations. Let*

$$\bar{\delta}_k = \min_{\substack{\text{polynomial } p \\ \text{w/ degree} < k}} \left(\max_{x\in\{\lambda_1(\mathbf{A}),\lambda_2(\mathbf{A}),\ldots,\lambda_n(\mathbf{A})\}} |1/x - p(x)| \right).$$

Then if we approximate $\mathbf{A}^{-1}\mathbf{x}$ by $\mathbf{y}_k = \|\mathbf{x}\|\mathbf{Q}\mathbf{T}^{-1}\mathbf{e}_1$, we are guaranteed that:

$$\|\mathbf{A}^{-1}\mathbf{x} - \mathbf{y}_k\| \leq \sqrt{\kappa(\mathbf{A})}\bar{\delta}_k\|\mathbf{x}\|, \tag{85}$$

where $\kappa(\mathbf{A})$ is the condition number $\lambda_{\max}(\mathbf{A})/\lambda_{\min}(\mathbf{A})$.

Proof. Let $\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$ be $\mathbf{A}$'s eigendecomposition. Let $\mathbf{A}^{1/2} = \mathbf{V}\mathbf{\Lambda}^{1/2}\mathbf{V}^T$ and $\mathbf{A}^{-1/2} = \mathbf{V}\mathbf{\Lambda}^{-1/2}\mathbf{V}^T$. Since $\mathbf{A}$ is positive semidefinite, $\mathbf{\Lambda}$ has no negative entries, so both of these matrices are real. Recall that $\mathbf{q}_1 = \mathbf{x}/\|\mathbf{x}\|$ and consider the minimization problem:

$$\mathbf{y}^* = \arg\min_{\mathbf{y}} \|\mathbf{A}^{-1/2}\mathbf{q}_1 - \mathbf{A}^{1/2}\mathbf{Q}\mathbf{y}\|.$$

This is a standard linear regression problem and is solved by

$$\mathbf{y}^* = \left(\mathbf{Q}^T\mathbf{A}\mathbf{Q}\right)^{-1}\left(\mathbf{Q}^T\mathbf{A}^{1/2}\right)\mathbf{A}^{-1/2}\mathbf{x} = \left(\mathbf{Q}^T\mathbf{A}\mathbf{Q}\right)^{-1}\mathbf{Q}^T\mathbf{q}_1.$$

From Claim 5 we have that $\mathbf{Q}^T\mathbf{A}\mathbf{Q} = \mathbf{T}$ and that $\mathbf{q}_1$ is orthogonal to all other columns in $\mathbf{Q}$. Thus,

$$\mathbf{y}^* = \mathbf{T}^{-1}\mathbf{e}_1.$$

Since $p(\mathbf{A})\mathbf{q}_1$ can be written as $\mathbf{Q}\mathbf{y}$ for any polynomial p with degree $< k$, it follows that

$$\|\mathbf{A}^{-1/2}\mathbf{q}_1 - \mathbf{A}^{1/2}\mathbf{Q}\mathbf{T}^{-1}\mathbf{e}_1\| \leq \min_{\substack{\text{polynomial } p\\ \text{w/ degree } < k}} \|\mathbf{A}^{-1/2}\mathbf{q}_1 - \mathbf{A}^{1/2}p(\mathbf{A})\mathbf{q}_1\|. \tag{86}$$

As an aside, if we scale by $\|\mathbf{x}\|$ and define the $\mathbf{A}$-norm $\|\mathbf{v}\|_{\mathbf{A}} \stackrel{\text{def}}{=} \mathbf{v}^T\mathbf{A}\mathbf{v}$, then this can be rewritten:

$$\|\mathbf{A}^{-1}\mathbf{x} - \left(\|\mathbf{x}\|\mathbf{Q}\mathbf{T}^{-1}\mathbf{e}_1\right)\|_{\mathbf{A}} \leq \min_{\substack{\text{polynomial } p\\ \text{w/ degree } < k}} \|\mathbf{A}^{-1}\mathbf{x} - p(\mathbf{A})\mathbf{x}\|_{\mathbf{A}}.$$

So, (86) is equivalent to the perhaps more familiar statement that, "the Lanczos approximation to $\mathbf{A}^{-1}\mathbf{x}$ is optimal with respect to the $\mathbf{A}$-norm amongst all degree $< k$ matrix polynomials $p(\mathbf{A})\mathbf{x}$." Returning to (86),

$$\begin{aligned}\|\mathbf{A}^{-1/2}\mathbf{q}_1 - \mathbf{A}^{1/2}\mathbf{Q}\mathbf{T}^{-1}\mathbf{e}_1\| &= \|\mathbf{A}^{1/2}\left(\mathbf{A}^{-1}\mathbf{q}_1 - \mathbf{Q}\mathbf{T}^{-1}\mathbf{e}_1\right)\| \\ &\geq \sqrt{\lambda_{\min}(\mathbf{A})}\|\mathbf{A}^{-1}\mathbf{q}_1 - \mathbf{Q}\mathbf{T}^{-1}\mathbf{e}_1\|. \end{aligned}\tag{87}$$

Additionally,

$$\begin{aligned}\|\mathbf{A}^{-1/2}\mathbf{q}_1 - \mathbf{A}^{1/2}p(\mathbf{A})\mathbf{q}_1\| &= \|\mathbf{A}^{1/2}\left(\mathbf{A}^{-1}\mathbf{q}_1 - p(\mathbf{A})\mathbf{q}_1\right)\| \\ &\leq \sqrt{\lambda_{\max}(\mathbf{A})}\|\mathbf{A}^{-1}\mathbf{q}_1 - p(\mathbf{A})\mathbf{q}_1\|. \end{aligned}\tag{88}$$

Plugging (87) and (88) into (86), we see that

$$\begin{aligned}\|\mathbf{A}^{-1}\mathbf{q}_1 - \mathbf{Q}\mathbf{T}^{-1}\mathbf{e}_1\| \;&\leq \sqrt{\kappa(\mathbf{A})} \min_{\substack{\text{polynomial } p\\ \text{w/ degree } < k}} \|\mathbf{A}^{-1}\mathbf{q}_1 - p(\mathbf{A})\mathbf{q}_1\| \\ &\leq \sqrt{\kappa(\mathbf{A})} \min_{\substack{\text{polynomial } p\\ \text{w/ degree } < k}} \left(\max_{x\in\{\lambda_1(\mathbf{A}),\ldots,\lambda_n(\mathbf{A})\}} |1/x - p(x)|\right).\end{aligned}$$

Theorem 24 follows from scaling both sides by $\|\mathbf{x}\|$.

$\square$

B.2 Linear systems in finite precision: Greenbaum's Analysis

As discussed in the Section 2.2, Greenbaum proves a natural extension of Theorem 24 for finite precision computations in [Gre89]. She studies a standard implementation of the conjugate gradient method, included here as Algorithm 2. This method only requires $O(n)$ space, in contrast to the $O(nk)$ space required by the more general Lanczos method.

Although it's not computed explicitly, just as in the Lanczos algorithm, the changing coefficients α and β generated by Algorithm 2 can be used to form a tridiagonal matrix $\mathbf{T}$. Furthermore, since each β shows how the norm of the residual $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{y}$ decreases over time, $\mathbf{T}$'s entries uniquely determine the error of the conjugate gradient iteration at any step k.

Algorithm 2 Conjugate Gradient Method

input: positive semidefinite $\mathbf{A} \in \mathbb{R}^{n \times n}$, # of iterations k, vector $\mathbf{x} \in \mathbb{R}^n$
output: vector $\mathbf{y} \in \mathbb{R}^n$ that approximates $\mathbf{A}^{-1}\mathbf{x}$

$\mathbf{y} = \mathbf{0}$, $\mathbf{r} = \mathbf{b}$, $\mathbf{p} = \mathbf{b}$
for $i \in 1, \dots, k$ **do**
  $\alpha \leftarrow \|\mathbf{r}\| / \langle \mathbf{r}, \mathbf{A}\mathbf{p} \rangle$
  $\mathbf{y} \leftarrow \mathbf{y} + \alpha \mathbf{p}$
  $\mathbf{r}_{new} \leftarrow \mathbf{r} - \alpha \mathbf{A}\mathbf{p}$
  $\beta \leftarrow -\|\mathbf{r}_{new}\| / \|\mathbf{r}\|$
  if $\beta == 0$ **then**
    break loop
  end if
  $\mathbf{p} \leftarrow \mathbf{r}_{new} - \beta \mathbf{p}$
  $\mathbf{r} \leftarrow \mathbf{r}_{new}$
end for
return $\mathbf{y}$

At a high level, Greenbaum shows that the $\mathbf{T}$ produced by a finite precision CG implementation is *equivalent* to the $\mathbf{T}$ that would be formed by a running CG on a larger matrix, $\bar{\mathbf{A}}$, who's eigenvalues all lie in small intervals around the eigenvalues of $\mathbf{A}$. She can thus characterize the performance of CG in finite precision on $\mathbf{A}$ by the performance of CG in exact arithmetic on $\bar{\mathbf{A}}$. In particular, Theorem 3 in [Gre89] gives:

Theorem 25 (Theorem 3 in [Gre89], simplified)**.** *Let $\mathbf{y}$ be the output of Algorithm 2 run for k iterations on positive definite $\mathbf{A} \in \mathbb{R}^{n \times n}$ and $\mathbf{x} \in \mathbb{R}^n$, with computations performed with $\Omega\left(\log \frac{nk(\|\mathbf{A}\|+1)}{\min(\eta, \lambda_{\min}(\mathbf{A}))}\right)$ bits of precision. Let $\Delta = \min(\eta, \lambda_{\min}(\mathbf{A})/5)$ There exists a matrix $\bar{\mathbf{A}}$ who's eigenvalues all lie in $\bigcup_{i=1}^n [\lambda_i(\mathbf{A}) - \Delta, \lambda_i(\mathbf{A}) + \Delta]$ and a vector $\bar{\mathbf{x}}$ with $\|\bar{\mathbf{x}}\|_{\bar{\mathbf{A}}} = \|\mathbf{x}\|_{\mathbf{A}}$ such that, if Algorithm 2 is run for k iterations on $\bar{\mathbf{A}}$ and $\bar{\mathbf{x}}$ in* exact arithmetic *to produce $\bar{\mathbf{y}}$, then:*

$$\|\mathbf{A}^{-1}\mathbf{x} - \mathbf{y}\|_{\mathbf{A}} \leq 1.2 \|\bar{\mathbf{A}}^{-1}\bar{\mathbf{x}} - \bar{\mathbf{y}}\|_{\bar{\mathbf{A}}}. \tag{89}$$

Note that for any positive definite $\mathbf{M}$, and $\mathbf{z}$ we define $\|\mathbf{z}\|_{\mathbf{M}} \stackrel{\text{def}}{=} \mathbf{z}^T \mathbf{M} \mathbf{z}$. $\bar{\mathbf{A}}$ is positive definite since $\Delta \leq \lambda_{\min}(\mathbf{A})/5$.

Theorem 25 implies the version of Greenbaum's results stated in Theorem 2.

Proof of Theorem 2. From our proof of Theorem 24 we have that:

$$\|\bar{\mathbf{A}}^{-1}\bar{\mathbf{x}} - \bar{\mathbf{y}}\|_{\bar{\mathbf{A}}} \leq \sqrt{\lambda_{\max}(\bar{\mathbf{A}})} \min_{\substack{\text{polynomial } p \\ \text{with degree } < k}} \left(\max_{x \in \bigcup_{i=1}^n [\lambda_i(\mathbf{A}) - \Delta, \lambda_i(\mathbf{A}) + \Delta]} |p(x) - 1/x| \right) \|\bar{\mathbf{x}}\|.$$

Additionally,

$$\|\mathbf{A}^{-1}\mathbf{x} - \mathbf{y}\|_{\mathbf{A}} \geq \sqrt{\lambda_{\min}(\mathbf{A})} \|\mathbf{A}^{-1}\mathbf{x} - \mathbf{y}\|.$$

Since $\Delta \leq \lambda_{\min}(\mathbf{A})/5$, $\lambda_{\max}(\bar{\mathbf{A}}) \leq 1.2\, \lambda_{\max}(\mathbf{A})$. Accordingly, (89) simplifies to

$$\|\mathbf{A}^{-1}\mathbf{x} - \mathbf{y}_{\mathbf{A}}\| \leq 1.44 \sqrt{\kappa(\mathbf{A})} \min_{\substack{\text{polynomial } p \\ \text{with degree } < k}} \left(\max_{x \in \bigcup_{i=1}^n [\lambda_i(\mathbf{A}) - \Delta, \lambda_i(\mathbf{A}) + \Delta]} |p(x) - 1/x| \right) \|\bar{\mathbf{x}}\|. \tag{90}$$

Finally,

$$\|\bar{\mathbf{x}}\| \leq \sqrt{\frac{1}{\lambda_{\min}(\bar{\mathbf{A}})}}\|\bar{\mathbf{x}}\|_{\bar{\mathbf{A}}} = \sqrt{\frac{1}{\lambda_{\min}(\bar{\mathbf{A}})}}\|\mathbf{x}\|_{\mathbf{A}} \leq \sqrt{\frac{\lambda_{\max}\mathbf{A}}{\lambda_{\min}(\bar{\mathbf{A}})}}\|\mathbf{x}\| \leq 1.25\sqrt{\kappa(\mathbf{A})}\|\mathbf{x}\|.$$

Plugging in (90) yields Theorem 2. □

C General polynomial perturbation bounds

Here we prove that bounded polynomials are generally stable under small perturbations of the input matrix $\mathbf{A}$, even if these perturbations are asymmetric. This result is used in proving Lemma 23, which guarantees that the final step in Algorithm 1 can be performed stably in finite precision. We start by providing the natural definition of matrix polynomials for asymmetric matrices:

Definition 2 (Matrix Polynomial). *For a degree q univariate polynomial $p(x) = c_0 + c_1 x + \ldots + c_q x^q$, for any square (not necessarily symmetric) matrix $\mathbf{M}$ we define the matrix polynomial $p(\mathbf{M})$ as:*

$$p(\mathbf{M}) \stackrel{\text{def}}{=} c_0\mathbf{I} + c_1\mathbf{M} + c_2\mathbf{M}^2 + \ldots + c_q\mathbf{M}^q.$$

When $\mathbf{M}$ is real symmetric the definition is equivalent to Defintion 1. If $\mathbf{M}$ can be diagonalized as $\mathbf{M} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$, $p(\mathbf{M}) = \mathbf{V}p(\mathbf{\Lambda})\mathbf{V}^T$, where $p(\mathbf{\Lambda})$ applies p to each eigenvalue on the diagonal of $\mathbf{\Lambda}$.

We can now prove the following result, which is similar to Lemma 9, but a bit simpler since we have an explicit expression for the additive error on $\mathbf{A}$.

Lemma 26. *Given symmetric $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{E} \in \mathbb{R}^{n\times n}$, $\mathbf{x}$ with $\|\mathbf{x}\| = 1$, and $\eta \geq \|\mathbf{E}\|$ if p is a polynomial with degree $< k$ and $|p(x)| \leq C$ for all $x \in [\lambda_{\min}(\mathbf{A}) - \eta, \lambda_{\max}(\mathbf{A}) + \eta]$ then:*

$$\|p(\mathbf{A} - \mathbf{E})\mathbf{x} - p(\mathbf{A})\mathbf{x}\| \leq \frac{2k^3 C}{\eta}\|\mathbf{E}\|. \tag{91}$$

This results again follows by writing p in the Chebyshev basis. Letting T_i be the i^{th} Chebyshev polynomial of the first kind (see definition in (20)), define as in Lemma 9 $\mathrm{r}_{\max} = \lambda_{\max} + \eta$, $\mathrm{r}_{\min} = \lambda_{\min} - \eta$, and:

$$\delta = \frac{2}{\mathrm{r}_{\max} - \mathrm{r}_{\min}} \qquad \text{and} \qquad \overline{T}_i(x) = T_i\left(\delta(x - \mathrm{r}_{\min}) - 1\right). \tag{92}$$

In this way, we have $\overline{T}(\mathrm{r}_{\min}) = T(-1)$ and $\overline{T}(\mathrm{r}_{\max}) = T(1)$.

Lemma 27. *Given symmetric $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{E} \in \mathbb{R}^{n\times n}$, $\mathbf{x}$ with $\|\mathbf{x}\| = 1$, and $\eta \geq \|\mathbf{E}\|$, for all $i < k$,*

$$\|\overline{T}_i(\mathbf{A} - \mathbf{E})\mathbf{x} - \overline{T}_i(\mathbf{A})\mathbf{x}\| \leq \frac{2i^2}{\eta}\|\mathbf{E}\|. \tag{93}$$

Proof. Define: $\bar{\mathbf{A}} \stackrel{\text{def}}{=} \delta(\mathbf{A} - \mathrm{r}_{\min}\mathbf{I}) - \mathbf{I}$ so (93) is equivalent to:

$$\|T_i(\bar{\mathbf{A}} - \delta\mathbf{E})\mathbf{x} - T_i(\bar{\mathbf{A}})\mathbf{x}\| \leq \frac{2i^2}{\mathrm{r}_{\max} - \mathrm{r}_{\min}}\|\mathbf{E}\|. \tag{94}$$

We use the following notation, mirroring that of Lemma 10:

$$\mathbf{t}_i \stackrel{\text{def}}{=} T_i(\bar{\mathbf{A}})\mathbf{x}, \qquad \tilde{\mathbf{t}}_i \stackrel{\text{def}}{=} T_i(\bar{\mathbf{A}} + \delta\mathbf{E})\mathbf{x},$$
$$\mathbf{d}_i \stackrel{\text{def}}{=} \mathbf{t}_i - \tilde{\mathbf{t}}_i, \qquad \boldsymbol{\xi}_i \stackrel{\text{def}}{=} \delta\mathbf{E}\tilde{\mathbf{t}}_{i-1}.$$

Obtaining (94) requires showing $\|\mathbf{d}_i\| \leq \frac{2i^2\|\mathbf{E}\|}{\mathrm{r_{max}} - \mathrm{r_{min}}}$. From the Chebyshev recurrence (20) for all $i \geq 2$:

$$\begin{aligned}\mathbf{d}_i &= \left(2\,\bar{\mathbf{A}}\,\mathbf{t}_{i-1} - \mathbf{t}_{i-2}\right) - \left(2\left(\bar{\mathbf{A}} + \delta\mathbf{E}\right)\tilde{\mathbf{t}}_{i-1} - \tilde{\mathbf{t}}_{i-2}\right)\\ &= 2\boldsymbol{\xi}_i + \left(2\,\bar{\mathbf{A}}\,\mathbf{d}_{i-1} - \mathbf{d}_{i-2}\right).\end{aligned} \tag{95}$$

Let U_i be the i^{th} Chebyshevy polynomial of the second kind (see definition in (25)). Using the same argument as used to show (26) in the proof of Lemma 9 we have, for any $i \geq 0$,

$$\mathbf{d}_i = U_{i-1}(\bar{\mathbf{A}})\boldsymbol{\xi}_1 + 2\sum_{j=2}^{i} U_{i-j}(\bar{\mathbf{A}})\boldsymbol{\xi}_j. \tag{96}$$

where for convenience we define $U_k(x) = 0$ for any $k < 0$. It follows that:

$$\|\mathbf{d}_i\| \leq 2\sum_{j=1}^{i} \|U_{i-j}(\bar{\mathbf{A}})\|\|\boldsymbol{\xi}_j\|.$$

Since all of $\bar{\mathbf{A}}$'s eigenvalues lie in $[-1,1]$ and for values in this range $U_k(x) \leq k+1$ [GST07]:

$$\|\mathbf{d}_i\| \leq 2\sum_{j=1}^{i}(i-j+1)\|\boldsymbol{\xi}_j\| \leq 2\sum_{j=1}^{i} i\|\boldsymbol{\xi}_j\|. \tag{97}$$

We finally bound $\|\boldsymbol{\xi}_j\| = \delta\mathbf{E}\tilde{\mathbf{t}}_{j-1}$ by:

$$\|\boldsymbol{\xi}_j\| \leq \delta\|\mathbf{E}\|\|T_{j-1}(\bar{\mathbf{A}} - \delta\mathbf{E})\|. \tag{98}$$

By our requirement that $\eta \geq \|\mathbf{E}\|$, $\bar{\mathbf{A}} - \delta\mathbf{E} = \delta\left(\mathbf{A} - \mathbf{E} - \mathrm{r_{min}}\,\mathbf{I}\right) - \mathbf{I}$ is a symmetric matrix with all eigenvalues in $[-1,1]$. Therefore,

$$\left\|T_{j-1}(\bar{\mathbf{A}} - \delta\mathbf{E})\right\| \leq \max_{x\in[-1,1]} |T_{j-1}(x)| \leq 1. \tag{99}$$

Plugging this back into (98), we have $\|\boldsymbol{\xi}_j\| \leq \delta\|\mathbf{E}\|$ and plugging into (97), $\|\mathbf{d}_i\| \leq \frac{4i^2}{\mathrm{r_{max}} - \mathrm{r_{min}}}\|\mathbf{E}\|$, which gives the lemma after noting that $\mathrm{r_{max}} - \mathrm{r_{min}} \geq 2\eta$. □

Using Lemma 27, we can prove Lemma 26. The argument is omitted, as it is identical to the proof of Lemma 9.

D Potential function proof of Chebyshev polynomial optimality

Our lower bound in Section 7 (specifically Lemma 12) shows that there is a matrix $\mathbf{A}$ with condition number κ, such that any polynomial which has $p(0) = 1$ and $|p(x)| < 1/3$ for x in a small range around each of $\mathbf{A}$'s eigenvalues must have degree $\Omega(\kappa^c)$ for some constant $1/5 \leq c \leq 1/2$. We do not carefully optimize the value of c, as any poly(κ) iteration bound demonstrates that Greenbaum's finite precision bound (Theorem 2) is significantly weaker than the exact arithmetic bound of (3) (proven in Theorem 24).

Here we demonstrate that a continuous version of our potential function argument can prove that any polynomial which is small on the *entire* interval $[1/\kappa, 1]$ but has $p(0) = 1$ must have degree $\Omega(\sqrt{\kappa}/\log \kappa)$. This matches the optimal bound achievable via the Chebyshev polynomials up to an $O(\log \kappa)$. While there are many alternative proofs of this fact, we include the argument because we believe that a more careful discretization could lead to a tighter version of Lemma 12.

Theorem 28. *There exists a fixed constant $c > 0$ such that for any $\kappa > 1$, any polynomial p with $p(0) = 1$ and $|p(x)| \leq 1/3$ for all $x \in [1/\kappa, 1]$ must have degree $k \geq c\sqrt{\kappa}/\log \kappa$.*

Proof. As in Section 7, we can again assume that all roots of $p(x)$, $r_1, \ldots, r_k$, are real and lie in $[1/\kappa, 1]$. Moving a root that is outside this range to the nearest boundary of the range or taking the real part of an imaginary root can only decrease the magnitude of p at $x \in [1/\kappa, 1]$. We write, using that $p(0) = 1$:

$$g(x) = \log(|p(x)|) = \sum_{i=1}^{k} \log|1 - x/r_i|.$$

We want to lower bound $\max_{x \in [1/\kappa, 1]} g(x)$. To do so, we first note that for any positive weight function $w(x)$:

$$\max_{x \in [1/\kappa, 1]} g(x) \geq \frac{\int_{1/\kappa}^{1} w(x) g(x) dx}{\int_{1/\kappa}^{1} w(x) dx}. \tag{100}$$

I.e. any weighted average lower bounds the maximum of a function. We also note that:

$$\frac{1}{k} \frac{\int_{1/\kappa}^{1} w(x) g(x) dx}{\int_{1/\kappa}^{1} w(x) dx} \geq \min_{r \in [1/\kappa, 1]} \frac{\int_{1/\kappa}^{1} w(x) \log|1 - x/r| dx}{\int_{1/\kappa}^{1} w(x) dx}, \tag{101}$$

So we focus on bounding this second quantity. We set:

$$w(x) = \frac{1}{x^{1.5}}.$$

Under this weighting, the denominator in (101) evaluates to:

$$\int_{1/\kappa}^{1} \frac{1}{x^{1.5}} dx = \left.\frac{-2}{\sqrt{x}}\right|_{1/\kappa}^{1} = 2\left(\sqrt{\kappa} - 1\right). \tag{102}$$

We now consider the numerator, which we denote as N.

$$N = \int_{1/\kappa}^{1} \frac{1}{x^{1.5}} \log|1 - x/r| dx = \int_{1/\kappa}^{r} \frac{1}{x^{1.5}} \log(1 - x/r) dx + \int_{r}^{1} \frac{1}{x^{1.5}} \log(x/r - 1) dx. \tag{103}$$

We will ultimately show that $\min_{r\in[1/\kappa,1]}[N] \geq -c\log\kappa$ for some fixed constant c. Then when we divide by the denominator of $2\left(\sqrt{\kappa}-1\right)$ computed in (102), we see that any root makes very little progress – just $O(\frac{\log \kappa}{\sqrt{\kappa}})$ – towards decreasing the weighted average of $g(x)$. Ultimately, we will thus require $k = O(\sqrt{\kappa}/\log\kappa)$ roots for $|p(x)|$ to be bounded for all $x \in [1/\kappa, 1]$ (formally shown by applying (101)). The first term of the split integral in (103) can be evaluated as:

$$\begin{aligned}
\int_{1/\kappa}^{r} \frac{1}{x^{1.5}} \log(1-x/r)dx &\geq \int_{0}^{r} \frac{1}{x^{1.5}} \log(1-x/r)dx \\
&= -2\log\left(\frac{\left(1+\sqrt{\frac{x}{r}}\right)^{1/\sqrt{r}+1/\sqrt{x}}}{\left(1-\sqrt{\frac{x}{r}}\right)^{1/\sqrt{r}-1/\sqrt{x}}}\right)\Bigg|_{0}^{r} \\
&= -2\log\left(2^{2/\sqrt{r}}\right).
\end{aligned} \tag{104}$$

The first inequality follows because $r \geq \frac{1}{\kappa}$. The second term can be evaluated as:

$$\begin{aligned}
\int_{r}^{1} \frac{1}{x^{1.5}} \log(x/r-1)dx &= -2\log\left(\frac{\left(1+\sqrt{\frac{x}{r}}\right)^{1/\sqrt{r}+1/\sqrt{x}}}{\left(\sqrt{\frac{x}{r}}-1\right)^{1/\sqrt{r}-1/\sqrt{x}}}\right)\Bigg|_{r}^{1} \\
&= -2\log\left(\frac{\left(1+\sqrt{\frac{1}{r}}\right)^{1/\sqrt{r}+1}}{\left(\sqrt{\frac{1}{r}}-1\right)^{1/\sqrt{r}-1}}\right) + 2\log\left(2^{2/\sqrt{r}}\right) \\
&\geq -2\log\left(8/r\right) + 2\log\left(2^{2/\sqrt{r}}\right)
\end{aligned} \tag{105}$$

The inequality follows from noting that $\frac{1}{\sqrt{r}} \geq 1$ and

$$\text{For all } x \geq 1, \qquad \frac{(1+x)^{x+1}}{(x-1)^{x-1}} \leq 8x^2.$$

Plugging (104) and (105) in (103) we have:

$$N \geq -2\log(8/r) = -O(\log\kappa).$$

Returning to (100) and (101), we can combine this with (102) (and an assumption that $\kappa \geq 2$) to conclude that:

$$\max_{x\in[1/\kappa,1]} g(x) \geq -ck\log(\kappa)/\sqrt{\kappa}$$

for some fixed constant $c > 0$. Thus, since $g(x) \stackrel{\text{def}}{=} \log(|p(x)|)$,

$$\max_{x\in[1/\kappa,1]} |p(x)| \geq e^{-ck\log(\kappa)/\sqrt{\kappa}}$$

and it follows that for $|p(x)|$ to be small (e.g. $\leq 1/3$) for all $x \in [1/\kappa, 1]$ we need $k \geq \delta\sqrt{\kappa}/\log(\kappa)$ for some fixed constant $\delta > 0$. $\square$

E Other omitted proofs

Claim 5 (Lanczos Output Guarantee)**.** *Run for $k \leq n$ iterations using exact arithmetic operations, the Lanczos algorithm (Algorithm 1) computes $\mathbf{Q} \in \mathbb{R}^{n \times k}$, an additional column vector $\mathbf{q}_{k+1} \in \mathbb{R}^n$, a scalar β_{k+1}, and a symmetric tridiagonal matrix $\mathbf{T} \in \mathbb{R}^{k \times k}$ such that:*

$$\mathbf{AQ} = \mathbf{QT} + \beta_{k+1}\mathbf{q}_{k+1}\mathbf{e}_k^T, \tag{5}$$

and

$$[\mathbf{Q}, \mathbf{q}_{k+1}]^T [\mathbf{Q}, \mathbf{q}_{k+1}] = \mathbf{I}. \tag{6}$$

Together (5) and (6) also imply that:

$$\lambda_{\min}(\mathbf{T}) \geq \lambda_{\min}(\mathbf{A}) \qquad \textit{and} \qquad \lambda_{\max}(\mathbf{T}) \leq \lambda_{\max}(\mathbf{A}). \tag{7}$$

When run for $k \geq n$ iterations, the algorithm terminates at the n^{th} iteration with $\beta_{n+1} = 0$.

Proof. (5) is not hard to check directly by examining Algorithm 1 (see [PS79] for a full exposition). For (6), note that by Step 10 in Algorithm 1, each $\|\mathbf{q}_i\| = 1$. So we just need to show that $[\mathbf{q}_1, \ldots, \mathbf{q}_k]$ are mutually orthogonal.

Assume by induction that $[\mathbf{q}_1, \ldots, \mathbf{q}_{k-1}]$ are mutually orthogonal. Now consider the value β_{k-1}. $\beta_{k-1} = \|\mathbf{q}_{k-1}\|$ before $\mathbf{q}_{k-1}$ is normalized in Step 10. Thus by the computation of $\mathbf{q}_{k-1}$ in Steps 3-5 we have: $\beta_{k-1} = \mathbf{q}_{k-1}^T (\mathbf{A}\mathbf{q}_{k-2} - \beta_{k-2}\mathbf{q}_{k-3} - \alpha_{k-2}\mathbf{q}_{k-2})$. By the induction hypothesis, this reduces to $\beta_{k-1} = \left(\mathbf{q}_{k-1}^T \mathbf{A}\right)\mathbf{q}_{k-2}$.

The above relation should make Steps 3-5 of Algorithm 1 more clear. We set $\mathbf{q}_k$ to equal $\mathbf{A}\mathbf{q}_{k-1}$, and explicitly orthogonalize against $\mathbf{q}_{k-2}$ (Step 3) and then $\mathbf{q}_{k-1}$ (Step 4-5). So we have $\mathbf{q}_k^T\mathbf{q}_{k-1} = 0$ and $\mathbf{q}_k^T\mathbf{q}_{k-2} = 0$. It remains to consider $[\mathbf{q}_1, \ldots, \mathbf{q}_{k-3}]$. For $j \geq 3$, $\mathbf{q}_{k-j}^T\mathbf{A}$ lies in the span of $[\mathbf{q}_1, \ldots, \mathbf{q}_{k-2}]$ and then applying the inductive hypothesis we see that $\mathbf{q}_{k-j}^T\mathbf{A}\mathbf{q}_{k-1} = 0$. So there is no need to explicitly orthogonalize $\mathbf{A}\mathbf{q}_{k-1}$ against these vectors when we generate $\mathbf{q}_k$. It follows that $\mathbf{q}_k$ is orthogonal to *all* vectors in $[\mathbf{q}_1, \ldots, \mathbf{q}_{k-1}]$, which proves (6).

If $k = n$, the same argument shows that $\mathbf{q}_{n+1}$ is orthogonal to $[\mathbf{q}_1, ..., \mathbf{q}_n]$, which implies that $\mathbf{q}_{n+1} = \mathbf{0}$ and so the loop terminates at step (7-8). We thus have $\beta_{n+1}\mathbf{q}_{n+1}\mathbf{e}_n^T = \mathbf{0}$ and $\mathbf{AQ} = \mathbf{QT}$.

Finally, (7) follows from the Courant-Fischer min-max principle. In particular, if we multiply (5) on the left by $\mathbf{Q}^T$ and note from (6) that $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$ and $\mathbf{Q}^T\mathbf{q}_{k+1} = \mathbf{0}$, then as desired,

$$\mathbf{T} = \mathbf{Q}^T\mathbf{AQ}.$$

Then since $\|\mathbf{Qy}\| = 1$ for any unit norm $\mathbf{y}$,

$$\begin{aligned}
\lambda_{\min}(\mathbf{A}) = \min_{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\| = 1} \mathbf{x}^T\mathbf{Ax} \quad &\leq \quad \min_{\mathbf{y} \in \mathbb{R}^k : \|\mathbf{y}\| = 1} \mathbf{y}^T\mathbf{Q}^T\mathbf{AQy} = \lambda_{\min}(\mathbf{T}) \text{ and} \\
\lambda_{\max}(\mathbf{A}) = \max_{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\| = 1} \mathbf{x}^T\mathbf{Ax} \quad &\geq \quad \max_{\mathbf{y} \in \mathbb{R}^k : \|\mathbf{y}\| = 1} \mathbf{y}^T\mathbf{Q}^T\mathbf{AQy} = \lambda_{\max}(\mathbf{T}).
\end{aligned}$$

□