Lanczos-based matrix function computation with applications to network analysis

Nicholas West
St Cross College
University of Oxford

A thesis submitted for the degree of
Master of Science by Research

Michaelmas 2025

For my daughter Louise

Acknowledgements

I thank my supervisors, Yuji and Renaud, for sharing ideas, recommending reading, giving feedback on writings, and suggesting interesting directions to research. Less tangible, but perhaps more important, was their enthusiasm about computation and complex systems.

I am grateful for the personal support I've received from my family, especially my wife, Moira, as well as my siblings and my parents. My daughter Louise was born during the writing of this dissertation. Writing alongside caring for an infant seemed impossible at first, but she gave me worthy motivation to finish this dissertation.

Finally, I thank John and Daria Barry and the Canterbury Institute for supporting me financially, helping me to understand the meaning and importance of the academic vocation, and providing me with a community of humble, serious-minded academics to emulate. Some of these scholars have become my dearest friends.

Abstract

Our aim in this dissertation is to study iterative methods for computing matrix functions on large, sparse, symmetric matrices and to apply these routines to a concrete problem that arises in the analysis of undirected networks: evaluating the importance of edges.

We first focus on the relationship between the Lanczos procedure in numerical linear algebra and scalar polynomial approximation. We show how connections to classical Gauss quadrature lead to efficient algorithms for computing matrix function quadratic forms, low-rank directional derivatives of the trace of a matrix function, and low-rank updates of the trace of a matrix function. We further consider the correspondence between Lanczos-based methods and polynomial interpolation for the computation of the action of a matrix function on a vector, low-rank directional derivatives, and low-rank updates. Our emphasis on connections to scalar approximation theory allows for a simple, unified presentation of these methods and their theoretical properties. Perhaps most notably, we are the first to provide characterization theorems equating the Lanczos approximants for dynamic problems—low-rank derivatives and low-rank updates—to Gauss quadrature and bivariate polynomial interpolation.

Second, we consider practical numerical experimentation and applications to network science. We conduct experiments on large matrices arising from structured networks to illustrate the low time-complexity of Lanczos-based methods for network science. As an application, we study the identification of important edges in a network. We show that matrix functions can interpolate between local degree-based and global eigenvector-based measures, and propose an algorithm for ranking all edges of the network in time nearly-linear in the number of edges of the network by combining Lanczos-based procedures with randomized algorithms for diagonal estimation.

Contents

List of Figures

Notation

Symbol	Meaning
$A, B, C, \cdots, Z$	Matrices
$a, b, c, \cdots, z$	Scalars
$\mathbf{a}, \mathbf{b}, \cdots, \mathbf{z}$	Vectors
$f(B)$	Matrix function f evaluated at B
$f\{A, B\}(C)$	Bivariate matrix function $f\{A, B\}$ evaluated at C
$L_f(B, X)$	Frechet derivative of f evaluated at B in direction X
$\lambda(B) = \{\lambda_1, \cdots, \lambda_n\}$	Eigenvalues of B (typically increasing order)
$B = V\Lambda V^T$	Spectral decomposition
A	Adjacency matrix
L	Graph Laplacian
B	Generic symmetric matrix
I	Identity matrix
$\mathbf{b}$	Starting vector for Lanczos procedure
n	Dimension of matrix / nodes in network
m	Degree of polynomial or number of Lanczos iterates
$\mathcal{K}_m(B, \mathbf{b})$	mth Krylov subspace generated by B and $\mathbf{b}$
U_m	Orthonormal basis for $K_m(B, \mathbf{b})$; output of Lanczos
T_m	Tridiagonal matrix from the Lanczos procedure
α_j, β_j	Lanczos recurrence coefficients
$\mathbf{e}_i$	ith standard basis vector
$\mathrm{Tr}(B)$	Trace of matrix B
$\|\cdot\|_2$	Vector or matrix 2-norm
$\|\cdot\|_F$	Frobenius norm
$\|\cdot\|_X$	Max-norm over compact X
$d\alpha(x)$	Positive measure supported on real interval $[a, b]$
$\|\cdot\|_{d\alpha}$	Norm induced by $d\alpha(x)$
$\langle x, y\rangle_{d\alpha}$	Inner product induced by $d\alpha(x)$
$\mathbb{1}$	Vector of all ones
δ_{ij}	Kronecker delta
$\mathbb{P}_m$	Space of univariate polynomials of degree at most m
$\mathbb{P}_{m\times m}$	Space of bivariate polynomials of maximal degree m
$l_X^f(x)$	Lagrange interpolating polynomial
$l_{X\times Y}^f(x, y)$	Bivariate Lagrange interpolating polynomial

Symbol	Meaning
$E^{\star}_{m}(f, X)$	Error in best degree m approximation to f over X
$R^{m}_{GQ}[f, d\alpha]$	m-point Gauss quadrature remainder
$\pi_0, \pi_1, \cdots$	Orthogonal polynomials
$\mathrm{Zer}(p)$	Zeroes of polynomial p

Chapter 1

Introduction

Chapter overview

This chapter provides an overview of the scope and particular aims of this dissertation. In Section 1.1, we provide a review of the existing literature on matrix function computation and applications of matrix functions in network science, culminating in a discussion of the central aims of this dissertation and an outline of the work. Section 1.2 provides a summary of our technical contributions. We conclude with some logistical notes about prior work (Section 1.3) and software (Section 1.4).

1.1 Introduction

Given a symmetric matrix B, the corresponding matrix function $f(B)$ has found numerous applications in scientific computing and data science [Hig08]. Most notable for this dissertation is that the evaluation of $f(B)$ when B is the adjacency matrix or Laplacian matrix of a network is of increasing interest for the analysis of networks [EH10, FSS16, BB20]. In the context of networks, B may be very large, but is often sparse, having few non-zero entries. Unfortunately, even when B is sparse, $f(B)$ is typically dense. This means that standard routines for matrix function computation (see [Hig08, GVL13]) relying on dense matrix methods and explicit formation of the full matrix $f(B)$ become infeasible, and it is desirable to develop iterative alternatives that rely primarily on matrix-vector products.

We are interested here in one especially powerful class of matrix function algorithms: Lanczos-based procedures [Lan50]. Lanczos-based algorithms for symmetric matrices have become one of the go-to classes of iterative algorithms for large-scale eigenvalue computation [Saa11], solving linear systems [Saa03], and more recently, computing

quantities related to matrix functions [Che22]. They are the specialization of incredibly powerful polynomial Krylov subspace methods to symmetric problems [LS12].

We focus here on matrix function computation. The most well-known Lanczos methods can approximate matrix-vector products and quadratic forms by implicitly forming a polynomial approximation $f(B)\mathbf{b} \approx p(B)\mathbf{b}$ or $\mathbf{b}^T f(B)\mathbf{b} \approx \mathbf{b}^T p(B)\mathbf{b}$, where the bulk of the computation is contained in a small number of matrix-vector products with B [Hig08, GM09, GVL13, Che22]. This provides access to the matrix function $f(B)$ without ever forming it explicitly (or even specifying the approximating polynomial), and the matrix B need only be accessed through matrix-vector products of the form $\mathbf{x} \mapsto B\mathbf{x}$, which is critical in many large-scale applications. Moreover, when compared with other *a priori* polynomial methods, where the approximating polynomial p is determined before the main computational routine is executed, Lanczos-based procedures can obtain drastically improved convergence properties. This accelerated convergence is due to the *spectral adaptivity* of Lanczos routines, whereby the approximating polynomial adapts to the spectral structure of the input matrix, sometimes reducing the number of matrix-vector products required greatly [DTT98, GKL20]. Another important aspect of these procedures is that since they make it possible to compute the matrix-vector products $\mathbf{x} \mapsto f(B)\mathbf{x}$ and quadratic forms $\mathbf{x} \mapsto \mathbf{x}^T f(B)\mathbf{x}$, they can be combined with recent advances in randomized numerical linear algebra [MT20] in order to efficiently solve an even broader class of matrix function problems. A notable example of this is the so-called *stochastic Lanczos quadrature* procedure for estimating the trace of a matrix function [UCS17]; but these methods can be used relatedly for diagonal estimation [BKS07, BN22] or in conjunction with randomized sketching for low-rank approximation [MT20].

More recently, Lanczos procedures have been successfully extended to allow for the solution of certain *dynamic* matrix function problems, such as the computation of the sensitivity of a matrix function to low-rank perturbations—formally, the Frechet derivative in a low-rank direction $L_f(B, \mathbf{b}\mathbf{b}^T)$ [Kre19, KKRS21]—and the computation of low-rank updates of matrix functions—that is, the computation of the difference $f(B + \mathbf{b}\mathbf{b}^T) - f(B)$ without forming $f(B)$ or $f(B + \mathbf{b}\mathbf{b}^T)$ explicitly [BKS18, BCKS21, CKM22]. Although their development is more recent, these dynamic algorithms may be of great importance for applications where, although the matrices may change frequently, the changes are expected to be of low-rank. A notable example of this is in the representations of dynamic or temporal networks [AS14, ML16]. When a network undergoes small modifications of its node and edge sets, the corresponding changes in its companion adjacency and Laplacian matrices may be low-rank, allowing for

adoption of dynamic Lanczos procedures to solve certain network problems; indeed, these dynamic methods have begun to see applications to updating communicability measures [BKS18] and the optimization of network structure through modification of its edge set [Sch23, MT23].

Unfortunately, the existing theoretical analysis of these dynamic procedures is limited. It is not exactly clear how they are related to their predecessors for the $f(B)\mathbf{b}$ and $\mathbf{b}^T f(B)\mathbf{b}$ problems. Although there are several convergence results for the dynamic procedures that extend well-known convergence results for these static problems, there is as of yet no characterization of spectral adaptivity—this is despite the empirical observation that it does indeed happen. Additionally, it has been shown that when applied only to the trace of a symmetric matrix, the algorithm for low-rank updates obtains a doubled convergence rate [CKM22]; this analysis, however, does not characterize a fundamental difference between approximation of the trace and approximation of the whole matrix, and as a byproduct of the analysis, the bound depends on the dimension of the matrix, which is undesirable since these methods may often be applied to very large matrices.

This brings us to our first main goal and contribution in this dissertation: we aim to provide a simple theoretical account of these dynamic Lanczos procedures—low-rank derivatives and updates, and the traces thereof—in continuity with their more well-known predecessors for the action of a matrix function on a vector and quadratic forms involving matrix functions. Importantly, Lanczos procedures for quadratic forms are *precisely* equivalent to a particular Gauss quadrature rule for a related scalar integration problem [GM09], and Lanczos procedures for the action on a vector *precisely* form an interpolating polynomial [Saa92, Hig08]. We will show in Chapter 3 that the dynamic procedures in the symmetric case, when applied to the trace of a matrix function, are also precisely equivalent to Gauss quadrature rules for related scalar integrals. This characterization explains the doubled convergence rate observed in [CKM22], improving upon the bound, and showing that the same accelerated convergence is present in the computation of the sensitivity of the trace of a matrix function to a low-rank perturbation, which is not discussed in [Kre19] or [KKRS21]. Beyond this, we show in Chapter 4 that the dynamic Lanczos procedures for computing the full Frechet derivative in a low-rank direction and the low-rank update of a matrix function (as presented first in [BKS18, Kre19]) can be tied precisely to *bivariate* polynomial interpolants just as the Lanczos procedure for the $f(B)\mathbf{b}$ problem can be tied to a *univariate* polynomial interpolant. Our theoretical results, when combined with simple error estimates from scalar polynomial approximation

theory [Che66, Tre19], provide a simple and unified presentation of the convergence properties of these methods. In Chapter 5, we show how these theoretical results can be combined with well-known results from scalar polynomial approximation theory to obtain effective *a priori* error bounds. This includes a slight modification of the standard bounds based on intervals to characterize the spectral adaptivity observed in computing low-rank updates. Notably, this same approach could be used to characterize the spectral adaptivity of any of the quadrature based results from Chapter 3.

Our second main contribution is to render the significance of these algorithms concrete by investigating their suitability for applications in network science. For this reason, we investigate in Chapter 6 numerically how Lanczos procedures interact with large-scale matrices stemming from network applications, where sparsity and spectral inhomogeneity are common. We conduct experiments with both synthetically generated networks and real large-scale networks with thousands to millions of nodes. We include a comparison between the actual number of iterates required by the Lanczos procedure to compute matrix functions accurately compared with the number of iterates that would be required by an *a priori* polynomial method on the same matrix. Finally, in Chapter 7, we propose a framework for assessing edge importances within networks using matrix functions. We show that the proposed metrics interpolate between a local degree-based measure and popular global eigenvector-based measures that were proposed for example in [ROH06, MSN10, TPER+12]. Moreover, we describe how the algorithms from Chapter 3 and Chapter 4 allow for evaluation of these measures at each individual edge in nearly linear time, and we also develop an efficient computational scheme for evaluating the importance of all of the edges at the same time using techniques of randomized diagonal estimation [BKS07, BN22], albeit with reduced accuracy.

Our hope is that this dissertation will appeal both to the algorithmist as well as the network scientist by laying out clearly a number of problems that can be solved efficiently by Lanczos-based routines, providing simple theoretical characterizations and error analysis of the approximations, and investigating a concrete example of how these methods can be assembled to solve a problem arising in practice—edge ranking. The scope of this dissertation is limited to symmetric matrices, smooth functions that can be approximated well by polynomials, and applications to the evaluation of edge importances. But we will include in our final chapter a discussion of further algorithmic considerations and extensions together with an outline of other network science applications that may benefit from the algorithms discussed in this work.

1.2 Summary of technical contributions

Although we have established our aims broadly within the narrative of this work, we also wish to itemize here for reference some of the most notable technical contributions that we make throughout this dissertation.

1. Proposition 2.2. We show an equivalence between the matrix function difference $f(Y) - f(X)$ and bivariate matrix functions as described in [Kre10, Kre19]. Notably, this makes explicit a theoretical connection between the algorithms of [BKS18] and [Kre19].

2. Proposition 2.4. We provide a simple modification of the standard error analysis for polynomial approximation on an interval which can be used to study approximation on an interval with a small number of outlying points. Importantly, this provides a non-asymptotic error bound and avoids complicated potential theoretic analysis based on unions of intervals.

3. Theorem 3.6 and Theorem 3.9. These results make explicit the connections between scalar Gauss quadrature and Lanczos-based algorithms for computing dynamics of the trace of a matrix function under rank-one perturbations. They extend the famous correspondence studied in [GM09] between matrix function quadratic forms and scalar quadrature to the dynamic case.

4. Theorem 4.6 and Theorem 4.9. In direct analogy to the previous point, these results provide precise characterization of the Lanczos-based approximants to the whole rank-one update and rank-one derivative matrix in terms of bivariate polynomial interpolation. This is a generalization of the famous result from [Saa92] showing that the Lanczos approximant for the $f(B)\mathbf{b}$ problem is a polynomial interpolant, which is also discussed in [Hig08], for example.

5. Proposition 5.3. This result (and the accompanying example) provides an explicit *a priori* bound explaining the spectral adaptivity observed when computing low-rank updates via the Lanczos procedure. We are not aware of another theoretical characterization of spectral adaptivity of this procedure, although it is noted empirically in the original work [BKS18].

6. Proposition 7.1. We show that the proposed measures of edge centrality in Chapter 7 interpolate between a local degree-based centrality and a global eigenvector-based centrality, the latter having been introduced in [ROH06].

7. Algorithm 11. We describe an algorithm for approximating the spectral sensitivity edge centrality of all of the edges in the network simultaneously using randomized diagonal estimation.

1.3 Statement of previous work

In a previous MSc thesis [Wes23], the author worked on Krylov methods for low-rank updates of matrix functions and considered applications to networks with the same supervisors [Wes23]. Briefly, the dissertation considered algorithms from [BKS18] for low-rank updates of matrix functions and provided analysis of the convergence of these algorithms for the matrix exponential and the matrix pseudoinverse using error estimates for analytic functions based on the Chebyshev polynomials [Tre19]. We considered an application to network robustness optimization with comparison to [WPKVM14] and [AB16]. There is some overlap of material in order to keep the present work self-contained, but our presentation is new and the present dissertation is much wider in scope.

1.4 Software

All experiments were run in MATLAB R2022b on an Apple M2 MacBook using only CPU. Source code of MATLAB implementations of the algorithms discussed is available upon request or at the author's GitHub `westnickp`. We provide our own implementations of most linear algebraic problems; however, scalar approximation is carried out using the Chebfun package [Tre19]. Network data was obtained in the form of adjacency matrices from the SuiteSparse collection [KAB+19].

Chapter 2

Preliminaries

Chapter overview

This chapter introduces the technical foundation necessary for the remainder of the dissertation. We discuss matrix functions (Section 2.1), polynomial methods for matrix function computation (Section 2.2), some basic scalar polynomial approximation theory (Section 2.3), and the use of matrix functions in network analysis (Section 2.4). Much of the material is standard; however, there are a few results that will be of special importance for this dissertation. In particular, *bivariate* matrix functions provide a theoretical framework for understanding matrix function dynamics as described in Proposition 2.1 and Proposition 2.2, results which will be used extensively in Chapter 4. Additionally, the modification of uniform approximation in Proposition 2.4 to account for outliers is significant for the error analysis of algorithms presented in later chapters, see especially Section 5.3. Finally, the description of edge modification on a network as representable by low-rank matrix dynamics in Proposition 2.7 will be important when we consider measuring edge centralities in Chapter 7, and it justifies the use of dynamic Lanczos procedures in network science more broadly.

2.1 Matrix functions of symmetric matrices

Our goal in this section is to provide a brief crash course on the mathematics of matrix functions as will be necessary for the remainder of the dissertation; notably, we will place a major emphasis on connections between generic matrix functions and scalar polynomials, as this connection is what is exploited by the algorithms discussed in this dissertation. It is sufficient for the purpose of this dissertation to assume that the matrix B is symmetric with real entries and the relevant scalar function f is smooth on an interval containing the eigenvalues of B. This drastically simplifies the presentation

in many cases. A more complete treatment of matrix functions is available in [Hig08], which includes many of the same results in more general language handling the case of non-symmetric matrices. We will also discuss a generalization to so-called bivariate matrix functions as presented in [Kre10]. In the footnotes we provide proofs or discuss notational convention where the presentation departs significantly from [Hig08, Kre10]. Although this section primarily reviews material, the emphasis on connections between matrix function *dynamics* and bivariate matrix functions is of note. In particular, Proposition 2.2 is a new result, which clarifies a remark found in the introduction of [Kre19]. See the remark after the statement of the proposition for further details.

We begin with three complimentary but ultimately equivalent views of matrix functions. The *matrix function* $f(B)$ can be defined first via the spectral decomposition of a symmetric matrix B according to

$$f(B) \equiv V f(\Lambda) V^T, \quad \text{with } B = V \Lambda V^T. \tag{2.1}$$

Here, V is an orthogonal matrix satisfying $V^T V = I$, which we may also write by expanding its columns $V = [\mathbf{v}_1, \cdots, \mathbf{v}_n]$ with $\mathbf{v}_i^T \mathbf{v}_j = \delta_{ij}$, where δ_{ij} is the Kronecker delta with $\delta_{ij} = 0$ if $i \neq j$ and $\delta_{ii} = 1$. The matrix Λ is a diagonal matrix whose ith diagonal entry $\lambda_i \equiv \Lambda_{ii}$ is an eigenvalue of B with corresponding normalized eigenvector $\mathbf{v}_i$, and $f(\Lambda)$ is the diagonal matrix formed by evaluating $f(\Lambda)_{ii} = f(\lambda_i)$, applying f to the diagonal elements of Λ. This definition emphasizes that the matrix function $f(B)$ can be viewed as the application of the scalar function $f(x)$ to the *eigenvalues* of B. Two more definitions fall naturally out of ordinary matrix algebra. For a polynomial $p(x) = c_0 + c_1 x + \cdots + c_k x^k$, we may define

$$p(B) = c_0 I + c_1 B + \cdots + c_n B^k, \tag{2.2}$$

where exponentiation of the scalar x is replaced with exponentiation of the matrix B. This leads more generally to our second definition for functions with convergent Taylor expansions, for which we write

$$f(B) = \sum_{i=0}^{\infty} f_i B^i, \quad \text{with } f(x) = \sum_{i=0}^{\infty} f_i x^i, \tag{2.3}$$

where the f_i are the coefficients of the Taylor expansion of f about 0. It is simple to show that where (2.3) exists, it is equivalent to (2.1).[1] Finally, for our third definition, we note that a matrix function may be defined by an interpolating polynomial. We will discuss polynomial interpolation more in Chapter 4, but recall that any k

[1] For example, observe $B^k = V \Lambda^k V^T$, and apply this term by term to the Taylor expansion of f.

distinct real numbers $X = \{x_1, x_2, \cdots, x_k\}$ with associated function values $f(X) = \{f(x_1), \cdots, f(x_k)\}$ can be interpolated uniquely by a polynomial of degree at most $k-1$, called the *Lagrange interpolating polynomial*, which we will denote $l_X^f(x)$. That is, $l_X^f(x)$ is at most degree $k-1$ with $l_X^f(x_i) = f(x_i)$ for each $i = 1, 2, \cdots, k$. Then we may define

$$f(B) = l_{\lambda(B)}^f(B), \tag{2.4}$$

where $l_{\lambda(B)}^f(B)$ is a polynomial depending on f and B, and is therefore well-defined for example as in (2.2). It is again simple to show equivalence of (2.4) with the preceding definitions; however, (2.4) resembles the Cayley-Hamilton theorem, thereby emphasizing the fact that every matrix function is equivalent to a finite degree polynomial of that matrix, although an important caveat is that the polynomial in question depends on the matrix itself. We will see later in Chapter 4 that the interpolatory property of matrix functions is important in characterizing the approximants formed by Lanczos procedures for matrix functions.

The matrix function $f(B)$ inherits a clean differential calculus whenever $f(x)$ is smooth on (and near) the eigenvalues of B. The appropriate notion of derivative for a matrix function $f(B)$ is the *Frechet derivative at* B, denoted $X \mapsto L_f(B, X)$ for a matrix input $X \in \mathbb{R}^{n\times n}$, where $L_f(B, X)$ is a linear operator on $\mathbb{R}^{n\times n}$ satisfying

$$L_f(B, X) = f(B + X) - f(B) + O(\|X\|^2) \tag{2.5}$$

for all symmetric X and some matrix norm $\|\cdot\|$. That is, $L_f(B, X)$ agrees with $f(B+X) - f(B)$ to first-order in the size of X in a neighborhood of B. The existence and uniqueness of the solution to (2.5) on the space of symmetric matrices is guaranteed by differentiability of f in a neighborhood of $\lambda(B)$. When X is fixed, if $L_f(B, \cdot)$ exists, then

$$L_f(B, X) = \lim_{t\to 0} \frac{f(B + tX) - f(B)}{t}, \tag{2.6}$$

so that the Frechet derivative coincides with the *directional* (or *Gateaux*) derivative, although it should be noted that (2.5) is a stronger requirement than the existence of (2.6) for a fixed X. For $B = V\Lambda V^T$, it can be shown [Hig08, Theorem 3.11] that the Frechet derivative has the following equivalent form:

$$L_f(B, X) = V\left(F \circ V^T X V\right) V^T, \tag{2.7}$$

where $\circ$ indicates the Hadamard product (entry-wise multiplication) and F is a matrix given element-wise by

$$F_{ij} = \begin{cases} \dfrac{f(\lambda_i) - f(\lambda_j)}{\lambda_i - \lambda_j} & i \neq j, \\ F_{ii} = f'(\lambda_i) & i = j. \end{cases} \tag{2.8}$$

We may view (2.7) as an analogy to (2.1), although it highlights the increased complexity of the Frechet derivative relative to the underlying matrix function. If f can be expanded as in (2.3), then we may also write [Hig08, Problem 3.6]

$$L_f(B, X) = \sum_{i=1}^{\infty} f_i \sum_{k=1}^{i} B^{k-1} X B^{i-k}, \tag{2.9}$$

which offers a definition for the Frechet derivative analogous to (2.3).

Interestingly, (2.7) hints at a *bivariate* scalar problem underlying the Frechet derivative, since the entries of F are given by a bivariate function. Indeed, one way to understand the relationship between the Frechet derivative $L_f(B, \cdot)$ and the matrix function $f(B)$ is through the concept of *bivariate matrix functions* [Kre10]. We introduce these now.[2] In [Kre10] it is shown that, given a bivariate scalar function $g(x, y)$, together with three symmetric n-by-n matrices A, B, C, one can define a *bivariate matrix function evaluated at* C, denoted $g\{A, B\}(C)$, with

$$g\{A, B\}(C) = U \left(G \circ U^T C V \right) V^T, \tag{2.10}$$

where $A = U \Theta U^T$ and $B = V \Lambda V^T$ are respective spectral decompositions and G is defined by $G_{ij} = g(\Theta_{ii}, \Lambda_{jj})$, i.e. evaluating g on the tensor-product grid of the eigenvalues of A and B, the two-dimensional domain $\lambda(A) \times \lambda(B) = \{(\lambda, \mu) : \lambda \in \lambda(A), \mu \in \lambda(B)\}$. Alternatively, if g is a bivariate polynomial of maximal degree n, i.e.

$$g(x, y) = \sum_{i=0}^{n} \sum_{j=0}^{n} g_{ij} x^i y^j, \tag{2.11}$$

then we define

$$g\{A, B\}(C) \equiv \sum_{j=0}^{n} g_{ij} A^i C B^j, \tag{2.12}$$

in analogy with (2.2). Scalar polynomial interpolation on tensor-product grids is well-established (see Chapter 4, or [IK66, Chapter 6]), so that if $l^g_{\lambda(A)\times\lambda(B)}(x, y)$ is the bivariate Lagrange interpolating polynomial to $g(x, y)$ on the tensor-product grid $\lambda(A) \times \lambda(B)$, and we observe that (2.10) depends only on the value of g on $\lambda(A) \times \lambda(B)$, we may state

$$g\{A, B\}(C) = l^g_{\lambda(A)\times\lambda(B)}\{A, B\}(C), \tag{2.13}$$

[2] Notably, we will assume that A, B, C are all square symmetric matrices, which simplifies the notation and some of the results from [Kre10] greatly for our purposes. We refer the reader to [Kre10] for a more detailed treatment of the more general cases.

which shows the equivalence of (2.10) and (2.12). Of course, if g has an infinite rather than finite expansion akin to (2.12), then $g\{A, B\}(C)$ is well-defined so long as the expansion converges on the tensor-product grid $\lambda(A) \times \lambda(B)$.

Let's summarize what we have said. We introduced $g\{A, B\}$ as a linear operator on $\mathbb{R}^{n\times n}$, just as $f(B)$ is a linear operator on $\mathbb{R}^n$. Moreover, just as every matrix function $f(B)$ can be viewed as extending the univariate function f to operate on the eigenvalues of B, the bivariate matrix function $g\{A, B\}$ extends a bivariate scalar function g to operate on the tensor-product grid of the eigenvalues of A and B. Additionally, both univariate and bivariate matrix functions are equivalent to an appropriately defined interpolating polynomial according to (2.4) and (2.13), respectively. Bivariate matrix functions extend the notion of the classical (univariate) matrix function and provide a unifying framework for various problems in numerical linear algebra, such as Lyapunov and Sylvester equations (see [Kre10, Kre19] for further discussion). In this work, we have introduced the notion of a bivariate matrix function because it will allow us to characterize the algorithms described in Chapter 5 in terms of certain bivariate scalar interpolation problems. This characterization will be possible because of the following fact: *the Frechet derivative of a matrix function and matrix function differences may be viewed as bivariate matrix functions.* We prove this in the following discussion.

Given a univariate function f, we may introduce the first order divided difference function

$$\Delta f(x, y) = \begin{cases} \dfrac{f(y) - f(x)}{y - x} & y \neq x, \\ f'(x) & x = y. \end{cases} \tag{2.14}$$

The following result, proved more generally in [Kre10], follows immediately under our assumptions and notation from (2.7) and (2.10).

Proposition 2.1 (The Frechet derivative is a bivariate matrix function)**.** *Let f be differentiable on the eigenvalues of B and $\Delta f(x, y)$ be as in (2.14). Then*

$$L_f(B, X) = \Delta f\{B, B\}(X) \tag{2.15}$$

for all X in $\mathbb{R}^{n\times n}$.

A simple argument allows us to extend this result to apply to generic changes in matrix functions $f(B + X) - f(B)$.

Proposition 2.2 (Matrix function differences are bivariate matrix functions)**.** *For matrices X, Y, we may write*

$$f(Y) - f(X) = \Delta f\{Y, X\}(Y - X). \tag{2.16}$$

In particular, we may write

$$f(B+X) - f(B) = \Delta f\{B+X, B\}(X). \tag{2.17}$$

Proof. We can assume that f is a polynomial of degree at most $2n-1$ since according to (2.4) any polynomial p that interpolates f on the union of the spectra of Y and X has $p(Y) = f(Y)$ and $p(X) = f(X)$.

Then if

$$f(x) = \sum_{i=0}^{2n-1} f_i x^i; \quad f(X) = \sum_{i=0}^{2n-1} f_i X^i, \tag{2.18}$$

we can write

$$f(y) - f(x) = \sum_{i=1}^{2n-1} f_i (y^i - x^i); \quad f(Y) - f(X) = \sum_{i=1}^{2n-1} f_i (Y^i - X^i). \tag{2.19}$$

The $i = 0$ index can be dropped since the constant or constant-times-identity (e.g. $X^0 = I = Y^0$) terms cancel. For scalar polynomials, the difference of two powers yields

$$y^k - x^k = \sum_{j=0}^{k-1} y^{k-1-j}(y-x)x^j, \tag{2.20}$$

which allows for the simplification

$$\Delta f(x,y) = \frac{f(y) - f(x)}{y - x} = \sum_{i=1}^{2n-1} f_i \sum_{j=0}^{i-1} y^{i-1-j} x^j. \tag{2.21}$$

Thus, since this is a bivariate polynomial, we may apply (2.12) to observe

$$\Delta f\{Y, X\}(Y - X) = \sum_{i=1}^{2n-1} f_i \sum_{j=0}^{i-1} Y^{i-1-j}(Y-X)X^j. \tag{2.22}$$

Now we consider $f(Y) - f(X)$. Crucially, the difference of two powers of matrices has the same form[3]

$$Y^k - X^k = \sum_{i=0}^{k-1} Y^{k-1-i}(Y-X)X^i. \tag{2.23}$$

Substituting this result into the right-hand equation of (2.19) yields

$$f(Y) - f(X) = \sum_{i=1}^{2n-1} f_i \sum_{j=0}^{i-1} Y^{i-1-j}(Y-X)X^j. \tag{2.24}$$

[3]This can be proven by a simple induction argument. Such a proof is offered in [BKS18, Proposition 3.1], for example.

Thus, equating (2.24) and (2.19) yields

$$f(Y) - f(X) = \Delta f\{Y, X\}(Y - X)$$

as desired.

□

Remark 2.1. It is shown in [BKS18, Lemma 2.2] that the following identity holds:[4]

$$f\left(\begin{bmatrix} X & Y - X \\ & Y \end{bmatrix}\right) = \begin{bmatrix} f(X) & f(Y) - f(X) \\ & f(Y) \end{bmatrix}.$$

Thus, an immediate corollary of a suitable generalization of our result to potentially non-symmetric matrices is that f evaluated on the following $2n \times 2n$ matrix yields

$$f\left(\begin{bmatrix} X & Y - X \\ & Y \end{bmatrix}\right) = \begin{bmatrix} f(X) & \Delta f\{Y, X\}(Y - X) \\ & f(Y) \end{bmatrix}.$$

Thus, the connection between matrix function differences and the divided difference function provided in Proposition 2.2 can be used to make explicit that the Krylov subspace method for bivariate matrix functions proposed in [Kre19] is equivalent to the Krylov subspace method for low-rank updates in [BKS18]. This resolves the following comment in the introduction of [Kre19]: "We note in passing that the algorithm proposed in this paper shares similarities with a recently proposed Krylov subspace method for performing low-rank updates of matrix functions [BKS18]." That is, the algorithms are not merely "similar" but are in fact equivalent according to Proposition 2.2.

Proposition 2.1 and Proposition 2.2 show that matrix function dynamics—differences and derivatives—may be viewed as bivariate matrix functions. Although it may seem that we have substituted a more complicated theoretical object (bivariate matrix functions) for a less complicated object (univariate matrix functions), there are two important features of these results. First, even though B or $f(B)$ may not have nice matrix structure, if f can be represented (or approximated) by a low-degree polynomial, then when X is of low-rank it will imply that $L_f(B, X)$ and $f(B+X)-f(B)$ are of low-rank, so that these and related quantities can be drastically easier to compute. Second, we will see later in Chapter 4 that this characterization allows us to precisely characterize the approximants formed by Lanczos-based methods for dynamic matrix function problems.

[4]The proof is only for $Y - X$ of low-rank in [BKS18], but the appropriate generalization to arbitrary rank is given in [Wes23].

We conclude this section with some comments on computation. When B is small enough that its eigenvalue decomposition can be computed efficiently, then (2.1) and (2.7) provide reliable mechanisms for computing matrix functions and Frechet derivatives (when the input matrix is symmetric).[5] We summarize the resulting procedures in Algorithm 1 and Algorithm 2. Typically, eigenvalue computations for symmetric matrices without further structural assumptions can be executed efficiently and stably in $O(n^3)$ flops [GVL13]. Nevertheless, in many applications, n may be tens-of-thousands, millions, or even billions, in which case computing an eigenvalue decomposition is too costly, or even impossible. Since matrix-vector products are at most $O(n^2)$, or faster for structured B such as sparse matrices, it is often favorable to develop iterative procedures that use a small number of matrix-vector products to approximate important quantities without forming $f(B)$ explicitly. We discuss examples of such procedures in the next section.

Algorithm 1 EigFunm(f, B)
Matrix function evaluation of a symmetric matrix.

1: **Input:** Scalar function f, symmetric matrix $B \in \mathbb{R}^{n \times n}$
2: **Output:** Matrix function $f(B)$
3: $(V, \Lambda) \leftarrow \text{Eig}(B)$ ▷ Symmetric eigenvalue decomposition
4: $f(B) \leftarrow V f(\Lambda) V^T$ ▷ Evaluate $f(\Lambda)_{ii} = f(\Lambda_{ii})$ along diagonal
5: **Complexity summary:** $O(n^3)$ flops, $O(n^2)$ memory

Algorithm 2 EigFrech$(\Delta f, B, X)$
Frechet derivative evaluation for a symmetric matrix and a fixed direction.

1: **Input:** Divided difference function Δf, symmetric matrices $B, X \in \mathbb{R}^{n \times n}$
2: **Output:** Frechet derivative $L_f(B, X)$
3: $(V, \Lambda) \leftarrow \text{Eig}(B)$ ▷ Symmetric eigenvalue decomposition
4: Form F via $F_{ij} \leftarrow \Delta f(\Lambda_{ii}, \Lambda_{jj})$
5: $L_f(B, X) \leftarrow V \left(F \circ V^T X V \right) V^T$
6: **Complexity summary:** $O(n^3)$ flops, $O(n^2)$ memory

[5] Notably, the Frechet derivative in Algorithm 2 requires the evaluation of the divided difference function $\Delta f(x, y)$, for which some care must be taken to ensure numerical stability if there are closely spaced eigenvalues, rather than merely using (2.14) directly. Since this is not a central subject of this thesis, we use (2.14) unless $|x - y| < \sqrt{\varepsilon}$, where ε is floating point machine precision, in which case we apply a low-order quadrature rule to the integral formulation of $\Delta f(x, y)$ given in [Hig08, Appendix B].

2.2 Polynomial methods and the Lanczos procedure

When the computation or storage of the full eigenvalue decomposition $B = V\Lambda V^T$ is too costly, it becomes desirable to develop alternative computational routines. Fortunately, in many applications, one does not require the full matrix function $f(B)$, and it is sometimes the case that B has special structure such that the matrix-vector products $\mathbf{x} \mapsto B\mathbf{x}$ are rapidly computed. In this section, we consider approximating the action of a matrix function on a vector—the computation of $f(B)\mathbf{b}$—and the related quadratic form $\mathbf{b}^T f(B)\mathbf{b}$. We discuss here two popular iterative approaches based on polynomial approximation—a Chebyshev-based approach and a Lanczos-based approach—both of which can be drastically more efficient than the dense methods described in the previous section. For further details, we recommend [Che22]. To conclude this section, we will compare these approaches in order to highlight why we are interested in studying *Lanczos*-based procedures in this dissertation, and our goal in future sections will be to interrogate the peculiar convergence properties of the Lanczos procedure and to extend the analysis from these well-known problems to the less well-known but analogous problems of low-rank dynamics.

We begin with a motivating theoretical result. It is easy to show[6] the following bound on the matrix 2-norm $\|\cdot\|_2$ of a matrix function:

$$\|f(B)\|_2 = \max_i |f(\lambda_i)| \tag{2.25}$$

where $\lambda_1, \lambda_2, \cdots, \lambda_n$ are the eigenvalues of B. Then in particular for any f, g defined on $\lambda(B)$, since $\|C\mathbf{b}\|_2 \leq \|C\|_2 \|\mathbf{b}\|_2$ for any matrix C, we have

$$\begin{aligned} \|f(B)\mathbf{b} - g(B)\mathbf{b}\|_2 &\leq \|\mathbf{b}\|_2 \cdot \|f(B) - g(B)\|_2 \\ &= \|\mathbf{b}\|_2 \max_i |f(\lambda_i) - g(\lambda_i)|. \end{aligned} \tag{2.26}$$

This means that we may approximate $f(B)\mathbf{b}$ by solving the scalar problem of approximating $f(x) \approx g(x)$ on an interval containing $\lambda(B)$ for some function g for which $g(B)\mathbf{b}$ is easy to evaluate.

We arrive now at a first method for this section based on forming an explicit polynomial approximation *a priori* using the *Chebyshev polynomials*. Approximating a scalar function f by polynomials on an interval $[a, b]$ is well-studied, with powerful approaches available via Chebyshev polynomials (see [Tre19]). For simplicity, we

[6] This follows from the fact that $B = V\Lambda V^T$ is Hermitian (and therefore unitarily diagonalizable) and $\|\cdot\|_2$ is unitarily invariant. Then $\|f(B)\| = \left\|Vf(\Lambda)V^T\right\| = \|f(\Lambda)\| = \max_i |f(\lambda_i)|$.

discuss here the case $a = -1, b = 1$, but a simple linear transformation handles the more general case. That is, on $[-1, 1]$ it is often possible to form a high accuracy approximation of the form

$$f(x) \approx p(x) = \sum_{i=0}^{m-1} c_i P_i(x), \tag{2.27}$$

where the P_i are the Chebyshev polynomials and the c_i are constant coefficients of this approximation. The Chebyshev polynomials on $[-1, 1]$ satisfy the three-term recurrence

$$P_{i+1}(x) = 2xP_i(x) - P_{i-1}(x), \tag{2.28}$$

with initial conditions

$$P_0(x) = 1, \quad P_1(x) = x, \tag{2.29}$$

which translates to a matrix recurrence

$$P_{i+1}(B)\mathbf{b} = 2BP_i(B)\mathbf{b} - P_{i-1}(B)\mathbf{b}. \tag{2.30}$$

Thus, one may consider

$$f(B)\mathbf{b} \approx p(B)\mathbf{b} = \sum_{i=0}^{m-1} c_i P_i(B)\mathbf{b}. \tag{2.31}$$

If we can approximate (2.31), then this of course immediately entails the approximation to the quadratic form

$$\mathbf{b}^T f(B)\mathbf{b} \approx \mathbf{b}^T p(B)\mathbf{b}. \tag{2.32}$$

Notably, we do not need to form B or the $P_i(B)$ explicitly, but can generate the basis vectors $P_i(B)\mathbf{b}$ one at a time using one matrix-vector product with B and the recurrence (2.30); moreover, we can implement this approximation whilst storing only three of the basis vectors $P_i(B)\mathbf{b}$ at a time. Importantly, m is determined by the scalar approximation of f independent of the size of the matrix, and for many smooth functions with $m \ll n$ we obtain a highly accurate approximation (see the discussion in the following section about error estimation). We state in Algorithm 3 the resulting procedure (see also [GKL20]). Assuming that inputs $a < b$ which bound the eigenvalues of B are known, Algorithm 3 requires $O(m)$ matrix-vector products with the matrix B and $O(n)$ storage (in addition to the storage of B). For a dense matrix, this means $O(mn^2)$ computational time, and if B is available as a sparse matrix, this means that the computational time will be $O(m \cdot \operatorname{nnz} B)$, where $\operatorname{nnz} B$ is the number of non-zero entries in B. Note that when $m \ll n$, and especially when

matrix-vector products are easy to compute, these storage and computational cost requirements may be drastically better than the $O(n^3)$ dense methods presented in the previous section.

Algorithm 3 ChebAction$(B, \mathbf{b}, m, a, b)$
An a priori Chebyshev approximation for $f(B)\mathbf{b}$.

Input: Symmetric $B \in \mathbb{R}^{n\times n}$, initial vector $\mathbf{b} \in \mathbb{R}^n$, approximation size $m > 2$, real numbers $a < b$ such that $\Lambda(B) \subset [a, b]$
Output: $\hat{\mathbf{f}}$ approximating $f(B)\mathbf{b}$
Form scalar approximation $f(x) \approx p(x) = \sum_{i=0}^{m-1} c_i P_i(x)$ on $[a, b]$
Form $\mathbf{T}_0 \leftarrow P_0(B)\mathbf{b}$, $\mathbf{T}_1 \leftarrow P_1(B)\mathbf{b}$
Initialize $\hat{\mathbf{f}} \leftarrow c_0\mathbf{T}_0 + c_1\mathbf{T}_1$.
for $j = 2, 3, \cdots, m$ **do**
    $\mathbf{T}_j \leftarrow 2(\alpha B - \beta I)\mathbf{T}_{j-1} - \mathbf{T}_{j-2}$ ▷ α, β scaling parameters for $[a, b] \neq [-1, 1]$
    $\hat{\mathbf{f}} \leftarrow \hat{\mathbf{f}} + c_j\mathbf{T}_j$
end for
Complexity summary: $O(m \cdot \operatorname{nnz} B)$ time and $O(n)$ space

Now we consider a second approach based on the *Lanczos* procedure [Lan50]. Given a matrix B and the vector $\mathbf{b}$, for some number of iterates $m < n$, the Lanczos procedure constructs a rectangular matrix $U_m \in \mathbb{R}^{n\times m}$ and a small companion symmetric tridiagonal matrix $T_m \in \mathbb{R}^{m\times m}$ as follows. We generate the columns of U_m one-by-one by computing $\mathbf{u}_{j+1} = B\mathbf{u}_j$, orthogonalizing it against its two predecessors $\mathbf{u}_j$ and $\mathbf{u}_{j-1}$, and then normalizing it. This may be written as a three-term recurrence

$$\mathbf{u}_{j+1} = \frac{1}{\beta_j}\left(B\mathbf{u}_j - \alpha_j\mathbf{u}_j - \beta_{j-1}\mathbf{u}_{j-1}\right), \quad j = 0, 1, 2, \cdots \tag{2.33}$$

where

$$\alpha_j = \mathbf{u}_j^T B\mathbf{u}_j, \quad \beta_j = \|Bu_j - \alpha_j\mathbf{u}_j - \beta_{j-1}\mathbf{u}_{j-1}\|_2, \tag{2.34}$$

with the initial conditions $\beta_0\mathbf{u}_0 = 0$, $\mathbf{u}_1 = \mathbf{b}/\|\mathbf{b}\|$. Although in principle this procedure breaks down if any $\beta_{j+1} = 0$, it turns out that the approximation formed with m replaced by j will be exactly correct in that case; for this reason, this is often termed "lucky breakdown" for the Lanczos procedure, and we will always assume that $\beta_{j+1} \neq 0$ as is common in practice.[7] In matrix form, the recurrence relation is summarized by

$$BU_m = U_m T_m + \beta_m \mathbf{u}_{m+1}\mathbf{e}_m^T. \tag{2.35}$$

[7]When $\beta_j = 0$, it means that we have found an *invariant subspace* of dimension j.

where

$$U_m = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_m], \quad T_m = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \beta_2 & \alpha_3 & \ddots & \\ & & \ddots & \ddots & \beta_{m-1} \\ & & & \beta_{m-1} & \alpha_m \end{bmatrix}. \tag{2.36}$$

We summarize the algorithm for generating U_m and T_m in Algorithm 4.

Algorithm 4 LANCZOS$(B, \mathbf{b}, m)$
The Lanczos procedure.

Input: Symmetric $B \in \mathbb{R}^{n\times n}$, initial vector $\mathbf{b} \in \mathbb{R}^n$, iteration count m.
Output: $(U_m, T_m, \beta_m, \mathbf{u}_{m+1})$ satisfying the Lanczos relationship.
$\mathbf{u}_0 \leftarrow \mathbf{0}$; $\mathbf{u}_1 \leftarrow \mathbf{b}/\|\mathbf{b}\|_2$; $\beta_0 \leftarrow 0$
for $j = 1, 2, \cdots, m$ **do**
    $\mathbf{r}_j \leftarrow B\mathbf{u}_j$
    $\alpha_j \leftarrow \mathbf{u}_j^T \mathbf{r}_j$
    $\mathbf{w}_j \leftarrow \mathbf{r}_j - \alpha_j \mathbf{u}_j - \beta_{j-1}\mathbf{u}_{j-1}$
    $\beta_j \leftarrow \|\mathbf{w}_j\|$
    $\mathbf{u}_{j+1} = \mathbf{w}_j/\beta_j$ ▷ If $\beta_j = 0$, exit ("lucky breakdown")
    $\mathbf{u}_{j+1} \leftarrow$ REORTHO$(\mathbf{u}_{j+1}, U_j)$ ▷ Necessary for numerical stability
end for
Complexity summary: $O(m \cdot \operatorname{nnz} B) + O(m^2 n)$ time and $O(mn)$ space

Further discussion of the properties of Algorithm 4 is given in Chapter 3 and Chapter 4; for now, we simply note some important well-known properties (see for example [Par98, Hig08, GM09, GVL13, Che22]). The extremal eigenvalues of the symmetric tridiagonal T_m are excellent estimates for the extremal eigenvalues of B [GVL13, Chapter 10], and the well-known Conjugate Gradient method [LS12] for solving the linear system $B\mathbf{x} = \mathbf{b}$ iteratively when B is positive definite (with an initial guess $\mathbf{x} \approx 0$) is theoretically equivalent to estimating

$$B^{-1}\mathbf{b} \approx \|\mathbf{b}\|_2 U_m T_m^{-1} \mathbf{e}_1. \tag{2.37}$$

The columns of the matrix U_m form an *orthonormal basis* for the *mth polynomial Krylov subspace of* B *applied to* $\mathbf{b}$:

$$\mathcal{K}_m(B, \mathbf{b}) \equiv \operatorname{span}\{\mathbf{b}, B\mathbf{b}, B^2\mathbf{b}, \cdots, B^{m-1}\mathbf{b}\} = \{\mathbf{x} \mid \mathbf{x} = p(B)\mathbf{b},\ p \in \mathbb{P}_{m-1}\}, \tag{2.38}$$

which contains any vector $\mathbf{x}$ that can be expressed as a polynomial of B of degree at most $m-1$ applied to $\mathbf{b}$, i.e. $\mathbf{x} = p(B)\mathbf{b}$. That the columns of U_m are contained in $\mathcal{K}_m(B, \mathbf{b})$ is obvious, but orthonormality follows from the equivalence of the recurrence

(2.33) to the performance of full Gram-Schmidt orthonormalization of the basis [GM09, Chapter 4]. In finite precision arithmetic, the basis U_m can suffer from a loss of orthogonality when this three-term recurrence is used; however, the method becomes practically stable if one implements *double reorthogonalization*, where one orthogonalizes each new $\mathbf{u}_j$ against all of its predecessors twice [GLR05, GLRE05]. This means that Lanczos procedures generically require between $O(m)$ and $O(m^2)$ vector-vector orthogonalizations, depending on the level of numerical care that is taken. Throughout the remainder of the work, our theoretical results apply to methods based on the Lanczos procedure run in *exact arithmetic*—that is, we assume that the basis U_m is truly orthonormal. If double reorthogonalization is carried out then these theoretical results can be expected to hold in practice, and we will perform all experiments in this dissertation with this full reorthogonalization.[8] The bulk of memory requirements are from storing U_m, which requires $O(mn)$ space, but low-memory implementations are possible (although reorthogonalization is not typically possible with low-memory implementations).

We return now to the problem at hand, computing $f(B)\mathbf{b}$ and $\mathbf{b}^T f(B)\mathbf{b}$. The fact that the columns of U_m span $\mathcal{K}_m(B, \mathbf{b})$ ensures that there exists some $\mathbf{x} \in \mathbb{R}^m$ such that

$$\|f(B)\mathbf{b} - U_m\mathbf{x}\|_2 = \min_{p \in \mathbb{P}_{m-1}} \|f(B)\mathbf{b} - p(B)\mathbf{b}\|_2. \tag{2.39}$$

Obtaining such an $\mathbf{x}$ is not easy in general. However, inspired by the estimate (2.37), we may consider generalizing from $f(x) = x^{-1}$ to arbitrary f, and consider the analogous

$$f(B)\mathbf{b} \approx \|\mathbf{b}\|_2 U_m f(T_m)\mathbf{e}_1. \tag{2.40}$$

The resulting algorithm obtained from this choice is given in Algorithm 5. Further details on its derivation are provdied in Chapter 4. Notably, the evaluation of $f(T_m)$ can be done via dense methods such as Algorithm 1, which require only $O(m^3)$ time,[9] so that when $m \ll n$ we have that Algorithm 5 is dominated by the $O(m^2 n) + O(m \cdot \operatorname{nnz} B)$ cost of the Lanczos procedure.

[8] The behavior of Lanczos procedures in finite precision arithmetic is still an open area of research [Che22]. It should be noted however that the Lanczos procedure without reorthogonalization may be much more efficient since it does not require the storage of all of U_m and does not require the additional vector-vector orthogonalizations which may become expensive. Incredibly, although the behavior in finite precision is drastically different from what can be expected in exact arithmetic, the resulting procedure is well-behaved.

[9] In fact, faster symmetric tridiagonal eigenvalue decomposition routines exist [Cup80, CR13].

Algorithm 5 LanAction$(B, \mathbf{b}, m)$
Lanczos-based approximation to $f(B)\mathbf{b}$.

1: **Input:** Symmetric $B \in \mathbb{R}^{n\times n}$, initial vector $\mathbf{b} \in \mathbb{R}^n$, iteration count m.
2: **Output:** Vector $\mathbf{f}$ approximating $f(B)\mathbf{b}$.
3: Form $(U_m, T_m, \beta_m, \mathbf{u}_{m+1}) \leftarrow$ Lanczos$(B, \mathbf{b}, m)$
4: $\mathbf{f} \leftarrow \|\mathbf{b}\| U_m f(T_m)\mathbf{e}_1$
5: **Complexity summary:** $O(m \cdot \operatorname{nnz} B) + O(m^2 n)$ time and $O(mn)$ space

We postpone a careful error analysis for now, but we note that the following bound is standard (see e.g. [CTU21]):

$$\|f(B)\mathbf{b} - \|\mathbf{b}\|_2 U_m f(T_m)\mathbf{e}_1\|_2 \leq 2\|\mathbf{b}\|_2 \min_{p\in\mathbb{P}_{m-1}} \max_{x\in S} |f(x) - p(x)|\,, \tag{2.41}$$

where S is any set containing the spectra of B and T_m. That is, this approximation is nearly-optimal with respect to the *a priori* observation (2.26). In particular, (2.41) is true for $S = [\lambda_1, \lambda_n]$ where λ_1, λ_n are the extremal eigenvalues of B since the field of values of T_m is contained in the field of values of B.

So the Lanczos procedure does *at least* as well as any *a priori* approximation of the kind described early in this section; however, there are two remarkable facts about this Lanczos-based procedure that we should note: its *spectral adaptivity* and its *accelerated convergence for quadratic forms*. First, we note that (2.41) holds for any S that contains the eigenvalues of B and T_m, which means that for matrices with heterogeneous eigenvalue distributions, for example with a small number of outlying eigenvalues, Lanczos may significantly outperform *a priori* methods that attempt to approximate f on the whole interval $[\lambda_1, \lambda_n]$. Second, when we approximate $\mathbf{b}^T f(B)\mathbf{b}$ rather than $f(B)\mathbf{b}$, then the convergence rate is effectively *doubled*. Noting $\mathbf{b}^T U_m = \|\mathbf{b}\|\,\mathbf{e}_1^T$, one can show that

$$\left|\mathbf{b}^T f(B)\mathbf{b} - \|\mathbf{b}\|_2^2 \mathbf{e}_1^T f(T_m)\mathbf{e}_1\right| \leq \|\mathbf{b}\|_2^2 \min_{p\in\mathbb{P}_{2m-1}} \max_{x\in S} |f(x) - p(x)|\,, \tag{2.42}$$

where S is as before. Note that the convergence rate here has doubled. We will offer a proof and further discussion in Chapter 3, but we merely mention this for now in order to offer motivation for the chapters to come. Indeed, as a central contribution of this dissertation, we will show that these properties—spectral adaptivity and accelerated convergence for quadrature rules—obtain when extending to dynamic matrix function computations discussed in Chapter 3 and Chapter 4.

We conclude this section with an example to illustrate these points. Suppose we form a 1000-by-1000 matrix B_1 with eigenvalues linearly spaced in $[-1, 1]$ and a

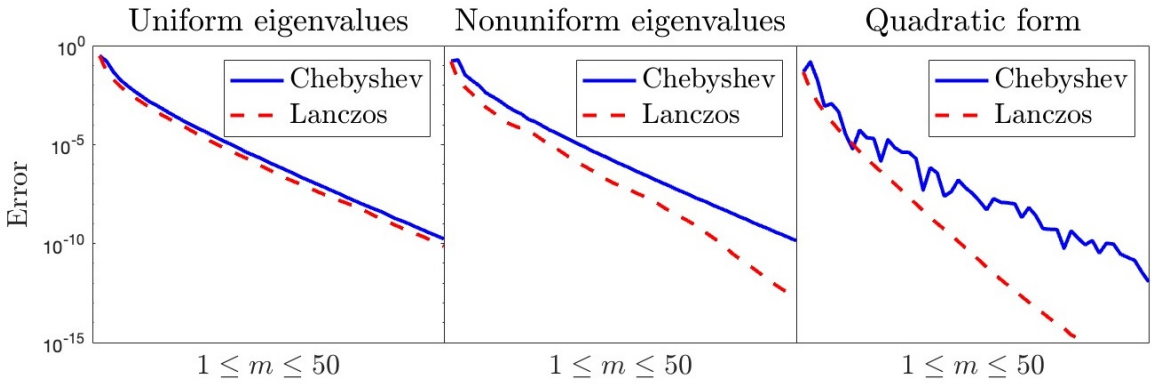

Figure 2.1: Depiction of accelerated convergence for Lanczos-based matrix function computation.

second matrix B_2 with 99% of its eigenvalues in $[-1, -1/3] \cup [1/3, 1]$ and the remaining 1% in $[-1/3, 1/3]$. For $f(x) = \sqrt{1.05 - x}$, we approximate $f(B_1)\mathbf{b}$ and $f(B_2)\mathbf{b}$ via Algorithm 3 and algorithm Algorithm 5. The resulting convergence curves are depicted in the first two plots of Figure 2.1. Notably, the convergence rates are nearly identical in the first plot, but in the second plot the Lanczos procedure exhibits the accelerating convergence that sets it apart. In the third plot, we also compare the approximation of $\mathbf{b}^T f(B_1)\mathbf{b}$ via the two methods, for which the Chebyshev method converges at a similar rate as before, but the Lanczos procedure obtains a doubled convergence rate.

2.3 Error estimation for polynomials

A key component of the success of the methods introduced in the previous section and to be studied in later chapters is that polynomials are good at approximating smooth functions, and that the convergence properties of polynomial approximations have been studied extensively [Che66, Riv81, Tre19]. In this section, we summarize some important properties of *uniform polynomial approximation*.

Figure 2.2 illustrates the phenomenon of interest. In the top row, we plot functions of decreasing smoothness (from left to right). Along the bottom, we plot the log of the maximal pointwise error in the function for the best polynomial approximants of increasing degree (from degree 0 up to degree 150). The error curves on the bottom exhibit drastically different convergence behaviors depending on the smoothness of the function. We can explain this phenomenon with some results about uniform approximation (sometimes called "best," "Chebyshev," or "minimax" approximation). Let us introduce the uniform error norm $\|f\|_X = \max_{x \in X} |f(x)|$ for a function f on

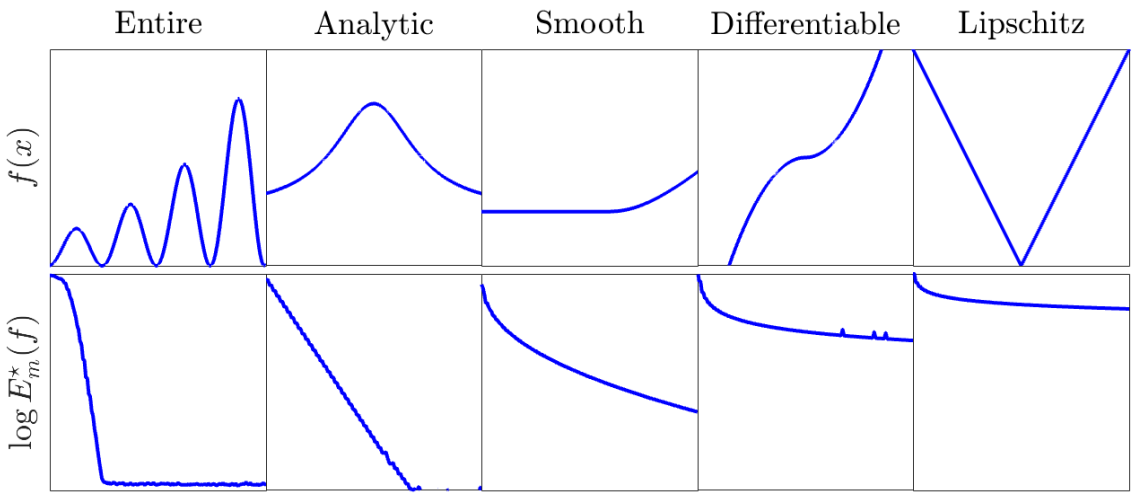

Figure 2.2: Illustration of Theorem 2.3. From left to right on the top we depict the functions $e^x \sin(x)$ (entire), $(1+x^2)^{-1}$ (analytic in a neighborhood of $[-1,1]$), $e^{-1/x}$ (infinitely differentiable), $|x|^3$ (differentiable), $|x|$ (continuous). The x-axis and y-axis are $[-1,1]$. Along the bottom row, we plot the error $\log E_m^\star(f,[-1,1])$ for $m = 0, 1, 2, \cdots, 150$.

a compact set X. We begin with approximation on a real interval $[a,b]$ through the quantity

$$E_m^*(f,[a,b]) \equiv \|f - p_m^\star\|_{[a,b]}, \tag{2.43}$$

which is the maximal point-wise error in the best degree m polynomial approximation $p_m^\star$ to f defined by

$$p_m^\star \equiv \operatorname*{argmin}_{p_m \in \mathbb{P}_m} \|f - p_m\|_{[a,b]}. \tag{2.44}$$

It is standard to show that such a polynomial exists, is unique, and can be computed via iterative Remez-type algorithms [Che66, Tre19]. Of course, for any continuous function f, the famous Weierstrass approximation theorem implies that $E_m^\star(f,[a,b]) \to 0$ as $m \to \infty$. But the following theorem summarizes a number of important results beyond Weierstrass about the behavior of $E_m^\star$ with m finite for different classes of functions f.

Theorem 2.3 (Uniform approximation)**.** *The following smoothness assumptions allow for the following bounds on uniform approximation:*

- *Suppose f is Lipschitz continuous on $[a,b]$ with Lipschitz coefficient K. Then*

$$E_m^\star(f,[a,b]) \leq \frac{K\pi(b-a)}{4(m+1)} \tag{2.45}$$

- *Suppose f is $d-1 \geq 1$ times continuously differentiable, with $f^{(d-1)}$ absolutely continuous and $f^{(d)}$ of total variation V. Then for $m > d$,*

$$E_m^\star(f, [a,b]) \leq \left(\frac{b-a}{2}\right)^d \left(\frac{2V}{\pi d(m-d)^d}\right) \tag{2.46}$$

- *Suppose f is analytic on the ellipse in the complex plane with foci at real numbers a, b and with full major and minor axes summing to $\rho(b-a)$ for some $\rho > 1$. We denote this ellipse $\mathbb{B}_{[a,b]}(\rho)$. Then*

$$E_m^\star(f, [a,b]) \leq \frac{2\rho^{-m}\|f\|_{\mathbb{B}_{[a,b]}(\rho)}}{\rho - 1} \tag{2.47}$$

Proof. The bound in (2.45) is a variant of *Jackson's theorem* and the quoted result is from [Che66, Chapter 4.6]. The bounds in (2.46) and (2.47) are obtained from [Tre19, Theorem 7.2] and [Tre19, Theorem 8.2].[10] The *Chebyshev projections* [Tre19] obtain the stated bounds in (2.46) and (2.47), although we have mapped the result from $[-1, 1]$ to $[a, b]$, causing the factor of $(b-a)^d/2^d$ to appear in (2.46), and the slightly more general definition of a Bernstein ellipse $\mathbb{B}_{[a,b]}(\rho)$. □

This theorem explains the behavior we see in Figure 2.2. Qualitatively, the smoother the function, the better the convergence rate—quantitatively, our convergence is of order $O(m^{-d})$ for functions that are $d-1$ times differentiable, and of order $O(\rho^{-m})$ for functions that are complex analytic, where $\rho > 1$ is a parameter measuring how analytic the function is.

One phenomenon that Theorem 2.3 does not capture, however, is what happens when the domain of approximation is not merely an interval. It is possible to use potential theoretic tools to determine rates of approximation for more complicated sets in the complex plane such as unions of intervals [DTT98, Kui06]; however, the theory is rather involved, the results are in essence asymptotic, and in particular sets of isolated points within the domain of approximation cannot be accounted for since these tools deal with sets of non-zero measure. We, however, will be interested in precisely the non-asymptotic, discrete regime, and in particular in the effect that a small number of outlying points has on the rate of approximation, as it is common in matrix applications to have a region in which the eigenvalues are effectively densely distributed together with a region where the eigenvalues are sparsely distributed. It

[10] A better constant on the error in (2.46) can be obtained (see *Favard's constants*), but (2.46) is simpler to state, a short proof is offered in [Tre19], and the constant is basically tight.

turns out that it is possible to offer a slight modification of Theorem 2.3 that can account for a small number of outliers effectively, which we consider now.

For points $X = \{x_1, x_2, \cdots, x_k\}$, together with the interval $[a, b]$, we wish also to study the error

$$E_m(f, [a, b] \cup X) = \|f - p_m^\star\|_{[a,b]\cup X} \tag{2.48}$$

where

$$p_m^\star \equiv \operatorname*{argmin}_{p\in\mathbb{P}_m} \|f - p\|_{[a,b]\cup X}. \tag{2.49}$$

We assume without loss of generality that the points in X are distinct and not contained in $[a, b]$. By the same standard theory (see e.g. [Che66]) the solution exists, is unique, and can be computed via variants of the Remez algorithm. In order to offer a simple characterization of error, let us introduce the divided difference function $\Delta_X f$. For a smooth function f, we wrote the first-order divided difference of f at x and y with $\Delta f(x, y)$ in (2.14). We generalize this to higher-order quotients (with a slightly modified notation) here. When $x \neq x_1$, we define the first-order divided difference

$$\Delta_{\{x_1\}} f(x) \equiv \frac{f(x) - f(x_1)}{x - x_1}, \tag{2.50}$$

and when $x = x_1$, $\Delta_{\{x_1\}} f(x) \equiv f'(x_1)$. Higher-order differences can be defined recursively via

$$\Delta_{\{x_1,\cdots,x_k\}} f(x) = \frac{\Delta_{\{x_1,\cdots,x_{k-1}\}} f(x) - \Delta_{\{x_1,\cdots,x_{k-1}\}} f(x_k)}{x - x_k}, \tag{2.51}$$

with the same exception at $x = x_k$ where $\Delta_{\{x_1,\cdots,x_k\}} f(x) = \Delta_{\{x_1,\cdots,x_{k-1}\}} f'(x)$. The ordering of the x_i does not mater, and there are many alternative formulations [dB05]. A nice short summary of their properties is also given in [Hig08, Appendix B]. Our notation here is non-standard, but we wish to emphasize that we may view $x \mapsto \Delta_X f(x)$ as a univariate function in x when the elements of X are fixed. Then we can state the following result:

Proposition 2.4 (Uniform approximation on an interval plus outliers)**.** *Let $m \geq k$. Then the following bound holds:*

$$E_m^\star(f, [a, b] \cup X) \leq E_{m-k}^\star(\Delta_X f, [a, b]) \cdot \left\| \prod_{i=1}^{k} (x - x_i) \right\|_{[a,b]}. \tag{2.52}$$

Proof. Let $l_X^f(x)$ be the polynomial of degree at most $k - 1$ interpolating f at the points in X. Note further the following characterization of the divided difference

function (see [dB05, Section 9]):

$$\Delta_X f(x) = \frac{f(x) - l_X^f(x)}{\prod_{i=1}^k (x - x_i)}.$$

Now consider polynomials of the form

$$\hat{p}(x) = l_X^f(x) + q(x) \prod_{i=1}^k (x - x_i),$$

where $q(x)$ is a polynomial of degree at most $m - k$, so that $\hat{p}$ is of degree at most m. Clearly $f(x_i) - \hat{p}(x_i) = 0$, so that

$$\|f - \hat{p}\|_{[a,b] \cup X} = \|f - \hat{p}\|_{[a,b]}.$$

Moreover, within $[a, b]$ we may factor out the non-zero monomial $\prod_{i=1}^k (x - x_i)$ to obtain

$$\begin{aligned} f(x) - \hat{p}(x) &= f(x) - l_X^f(x) - q(x) \prod_{i=1}^k (x - x_i) \\ &= (\Delta_X f(x) - q(x)) \prod_{i=1}^k (x - x_i). \end{aligned}$$

Thus, using polynomials of this form, we obtain

$$\begin{aligned} E_m^\star(f, [a, b] \cup X) &= \min_{p \in \mathbb{P}_m} \|f - p\|_{[a,b] \cup X} \\ &\leq \min_{q \in \mathbb{P}_{m-k}} \left\| (\Delta_X f(x) - q(x)) \prod_{i=1}^k (x - x_i) \right\|_{[a,b]} \\ &\leq \min_{q \in \mathbb{P}_{m-k}} \|\Delta_X f(x) - q(x)\|_{[a,b]} \cdot \left\| \prod_{i=1}^k (x - x_i) \right\|_{[a,b]} \\ &= E_{m-k}^\star(\Delta_X f, [a, b]) \cdot \left\| \prod_{i=1}^k (x - x_i) \right\|_{[a,b]} \end{aligned}$$

□

Remark 2.2. Although the proof of Proposition 2.4 is a straightforward modification of standard results from approximation theory, we have not seen a similar treatment elsewhere in the literature. We believe the result to be new.

To be clear, for a fixed k, once $m > k$ Proposition 2.4 shows that the rate of approximation for a function f by polynomials on an interval-plus outliers $[a,b] \cup X$ is governed by the difficulty of approximating $\Delta_X f$ on $[a,b]$ rather than f on a larger interval containing all of $[a,b] \cup X$. The important fact about divided differences for this characterization is that they inherit the same smoothness properties within $[a,b]$ as f. If f has d derivatives on $[a,b]$, then so does $\Delta_X f$, since the points in X are outside of $[a,b]$. And if f is analytic on a closed, convex domain $\Omega \supset X$, then so is $\Delta_X f$. In fact, it is possible to provide the following bounds on the size of $\Delta_X f$.

Proposition 2.5. *If f is analytic on a connected domain Ω in the complex plane with $X \subset \Omega$, where $X = \{x_1, x_2, \cdots, x_k\}$ consists of k distinct complex points, then $\Delta_X f$ is analytic on Ω as well. The following estimate holds for closed, convex Ω:*

$$\|\Delta_X f\|_\Omega \leq \frac{\|f^{(k)}\|_\Omega}{k!}. \tag{2.53}$$

Now let $[c,d] \supset ([a,b] \cup X)$. If f is $k \geq 0$ times differentiable on $[c,d]$ then $\Delta_X f$ is also k times differentiable on $[a,b]$. Moreover, for each x in $[a,b]$ there exists some ξ in $[c,d]$ depending on x such that

$$\Delta_X f(x) = \frac{f^{(k)}(\xi)}{k!}. \tag{2.54}$$

Proof. These results follow from the Genocchi-Hermite formula and the generalized mean value theorem [Hig08, Appendix B.16]. □

Note that Proposition 2.5 together with Proposition 2.4 means first that the *rate* of approximation of f on $[a,b] \cup X$ is identical to the rate of approximation to f on $[a,b]$, as we would expect given asymptotic results from potential theory. But beyond this, Proposition 2.4 holds for finite m and is easy to study, allowing for some elementary error estimates that we will see in Chapter 5 can be effective in some practical situations

2.4 Network analysis via matrix functions

We conclude this chapter by returning to the ultimate application of interest: the analysis of networks. Informally, a *network* (or mathematically equivalent, a *graph*) is a collection of nodes connected by edges, often used to model pairwise relationships in the social sciences, transportation studies, and biological or chemical systems [Est12b, New18]. Networks are also important in data science and machine learning

contexts since they are a near-ubiquitous model for relational data [FSS16, Ham20]. Standard tasks in network science involve the ranking of nodes or edges by importance or centrality within the network, providing generalized measures of similarity and dissimilarity or proximity and distance on the network, revealing community structure wherein certain subsets of the nodes are more strongly connected within the community than they are outside of the community, or finding low-dimensional representations of the network's elements.

Our goal in this section is to describe how matrix functions offer a convenient theoretical framework for many of these tasks [EH10, BK13]. Below, we introduce the main matrix representations of networks used in this dissertation and discuss how matrix functions allow for measurements of network structure.

First, we introduce the *adjacency matrix*. Suppose the n nodes of a network are labeled $1, 2, \cdots, n$ and that the edges of the network are labeled as ordered pairs, for example $e_k = (i, j)$ when node i is connected to j by the kth edge. Then the adjacency matrix A is a square matrix with $A_{ij} = 1$ if nodes i and j are connected, and $A_{ij} = 0$ otherwise. Notably, A is symmetric with non-negative entries. The number of edges connected to a node is a standard measure of local importance called the *degree* of the node. In this context, the row-sums, which can be computed by the matrix-vector product $A\mathbb{1}$ where $\mathbb{1}$ is a vector of all ones, yield the degrees of all nodes. A *walk of length* k on the network is a sequence of k edges $(i_1, i_2), (i_2, i_3), \cdots, (i_k, i_{k+1})$ such that each (i_l, i_{l+1}) is an edge in the network incident to nodes i_l and i_{l+1}. Our first result is a standard identity in spectral graph theory: powers of the adjacency matrix count the number of walks between nodes according to the edge structure.

Theorem 2.6 (Adjacency matrix powers count paths)**.** *The number of walks of length $k \geq 1$ between nodes i and j is given by $(A^k)_{ij}$.*

Proof. This result follows directly from the definition of matrix-matrix multiplication and induction. □

In light of Theorem 2.6, any matrix function f with a power series representation, when applied to the adjacency matrix, has a clear interpretation: it may be viewed as defining a (potentially infinite) weighted sum of the number of walks between two nodes. That is, if

$$f(A) = \sum_{k=0}^{\infty} f_k A^k, \tag{2.55}$$

then $f(A)_{ij}$ gives a generalized measure of the number of walks between two nodes, called the *f-communicability between nodes i and j*. Two node centrality measures are

immediate. The matrix-vector product $f(A)\mathbb{1}$, which sums the rows of $f(A)$, can be viewed as a generalization of node degree, measuring how well each node communicates with the rest of the graph, so that we call $(f(A)\mathbb{1})_i$ the *total f-communicability of node i*. Relatedly, the diagonal entry $f(A)_{ii}$ measures the number of walks that leave and return to node i, which we call the *f-centrality of node i*. Correspondingly, the sum of all entries $\mathbb{1}^T f(A)\mathbb{1}$ will be called the *total f-communicability of the network*, and the sum of diagonal entries—the trace $\mathrm{Tr}(f(A))$—will be called the *total f-centrality of the network*. These latter two measures can be used to describe the robustness of a network according to the abundance of communicating paths. This terminology was introduced in [EH10].

The matrix function $f(A)$ has an important geometric interpretation too. If $f(A)$ is a positive semidefinite matrix, then the quantity

$$C_{ij} = f(A)_{ii} + f(A)_{jj} - 2f(A)_{ij}, \tag{2.56}$$

which we call the *communicability distance* after [Est12a], forms a (squared) Euclidean distance metric on the network, and $f(A)$ may correspondingly be interpreted as a positive semidefinite *kernel* measuring the similarity between nodes on the graph with $f(A)_{ij}$. One may also consider the related *communicability angle* between nodes i and j [EH16] defined by

$$\gamma_{ij} = \frac{f(A)_{ij}}{\sqrt{f(A)_{ii} f(A)_{jj}}}, \tag{2.57}$$

which results from normalizing the kernel matrix $f(A)$. For further details on network kernels, see [FSS16].

Two important concrete examples of matrix functions are the *matrix exponential* $e^{\alpha A}$ and the *matrix resolvent* $(I - \alpha A)^{-1}$, which have the Taylor expansions

$$e^{\alpha A} = I + \alpha A + \frac{\alpha^2 A^2}{2!} + \cdots, \quad (I - \alpha A)^{-1} = I + \alpha A + \alpha^2 A^2 + \cdots, \tag{2.58}$$

where $\alpha < \|A\|_2$ is required for the second series to converge. Notably, the matrix exponential gives rise to the *Estrada index* of a network $\mathrm{Tr}(e^A)$ [EH10], and the matrix resolvent gives rise to the famous Katz centralities $(I - \alpha A)^{-1}\mathbb{1}$ [Kat53].

A second important matrix representation of a network is the *graph Laplacian*, denoted L, and defined by $L_{ij} = -1$ if nodes i and j are connected, with the degrees of the nodes along the diagonal, $L_{ii} = (A\mathbb{1})_i$. If D is a diagonal matrix with D_{ii} the degree of node i, then we may write $L = D - A$. Notably, the Laplacian is always singular with $\mathbb{1}$ in its null-space; however, when a network is connected, this zero eigenvalue is unique and its null-space is consequently one-dimensional. The smallest

Quantity	**Interpretation**
$f(A)_{ii}$	f-centrality of node i (self-communicability)
$(f(A)\mathbb{1})_i$	Total f-communicability of node i
$f(A)_{ij}$	f-communicability between nodes i and j
$f(A)_{ii} + f(A)_{jj} - 2f(A)_{ij}$	Communicability distance between nodes i and j
$\gamma_{ij} = f(A)_{ij}/\sqrt{f(A)_{ii}f(A)_{jj}}$	Communicability angle between nodes i and j
$\mathrm{Tr}(f(A))$	Total f-centrality of the network
$\mathbb{1}^T f(A)\mathbb{1}$	Total f-communicability of the network

Table 2.1: Interpreting matrix functions of the adjacency matrix.

non-zero eigenvalues and eigenvectors often encode community structure, rates of diffusion, and more on the network [FSS16]. The matrix exponential has another valuable physical interpretation when applied to the Laplacian. Diffusion on the graph may be modeled by

$$\frac{d\mathbf{x}}{dt} = -L\mathbf{x}, \tag{2.59}$$

which has the analytical solution

$$\mathbf{x}(t) = e^{-tL}\mathbf{x}_0 \tag{2.60}$$

So that e^{-tL} measures amount of heat diffusing from i to j in t units of time. In machine learning communities, this is often referred to as the *heat kernel* of the graph [Chu07]. Another popular matrix function is the Moore-Penrose Pseudoinverse, which in this instance may be written via

$$L^\dagger = \sum_{i=2}^{n} \lambda_i^{-1}\mathbf{v}_i\mathbf{v}_i^T, \tag{2.61}$$

where L has eigenvalues $0 = \lambda_1 < \lambda_2 \leq \lambda_3 \leq \cdots \leq \lambda_n$ with corresponding normalized eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \cdots, \mathbf{v}_n$. This matrix function gives rise to effective resistances and effective resistance distances [KR93, FSS16].

We introduce two final matrices that emphasize an edge-level representation of the network. Suppose that we label the s edges of the network $e_1, e_2, \cdots, e_s$, and we say that the ith edge is *incident* on nodes j, k if it joins those two nodes, i.e. $e_i = (j, k)$. Furthermore, suppose that we assign an arbitrary orientation to each edge so that $e_i = (j, k)$ indicates that e_i initiates at node j and terminates at node k. Then we define $X \in \mathbb{R}^{s\times n}$ the *signed incidence matrix* according to $X_{ij} = 1$ if edge i initiates at node j and $X_{ik} = -1$ if edge i terminates at node k, and $X_{ij} = 0$ otherwise. We

relatedly define the *unsigned incidence matrix* Y with $Y_{ij} = |X_{ij}|$. We will use $\mathbf{x}_i$ to indicate the transpose of the ith row of the incidence matrix X, and $\mathbf{y}_i$ to indicate the transpose of the ith row of the unsigned incidence matrix Y. Note that $\mathbf{x}_i$ has exactly two non-zero entries of value -1 and $+1$ in the j, k indices if $e_i = (j, k)$, and $\mathbf{y}_i$ has those same entries filled both by $+1$. Now we can write the Laplacian L and the adjacency matrix A as a sum of their edges—namely,

$$L = X^T X = \sum_{i=1}^{s} \mathbf{x}_i \mathbf{x}_i^T, \tag{2.62}$$

and

$$A = \left(\frac{Y^T Y - X^T X}{2} \right) = \frac{1}{2} \sum_{i=1}^{s} \left(\mathbf{y}_i \mathbf{y}_i^T - \mathbf{x}_i \mathbf{x}_i^T \right). \tag{2.63}$$

This perspective emphasizes matricially that a network is the sum of its edges, and notably it implies that the removal (or addition) of an edge is a rank-1 modification of the Laplacian or a rank-2 modification of the adjacency matrix.

Proposition 2.7 (Low-rank dynamics of undirected networks)**.** *The addition or deletion of an edge is a symmetric rank-1 modification of the Laplacian and a symmetric rank-2 modification of the adjacency matrix, as presented in (2.62) and (2.63).*

Remark 2.3. It also follows immediately that the removal of node i, where we delete all edges incident on the node, is a rank d_i modification of the Laplacian, were d_i is the degree of node i. It can be shown that the removal of a node of the adjacency matrix is a rank 2 modification of the adjacency matrix *regardless of the degree of the node*.

We will return to networks in the final part of the dissertation, but we hope that this last comment on low-rank dynamics convinces the reader that networks provide a relevant application within which low-rank dynamics are natural to consider. We will see that there are further matricial structures that are natural in real-world networks in Chapter 6, such as sparsity and spectral inhomogeneity, providing more structure for the matrix methods introduced in Section 2.2 and further considered in Chapter 3 and Chapter 4.

Further references

See [Hig08] for an introduction to matrix function theory and computation. For introductions to network science and graph theory, together with discussions of

connections to matrix functions, we suggest [Est12b, FSS16], as well as [Spi19] for an overview of spectral graph theory with algorithmic considerations. For iterative methods for eigenvalue problems and solving linear systems, see [Saa03, Saa11, LS12], for example. Polynomial Krylov methods for matrix function problems are discussed in [Hig08, GVL13], and further excellent resources with extensions to rational Krylov methods are the PhD theses [Gü13, Sch16]. Important further references that consider symmetric problems and the Lanczos procedure in further detail are the book [GM09] and Chen's recent PhD thesis [Che22]. Polynomial approximation theory is discussed in numerous texts [Tim63, Che66, IK66, Riv81, Tre19].

Chapter 3

Lanczos quadrature

Chapter overview

In this chapter, we study the connection between Gauss quadrature and the Lanczos procedure applied to various matrix function problems. More concretely, we review results from [GM09] and [Che22], which show that the approximation

$$\mathbf{b}^T f(B)\mathbf{b} \approx \|\mathbf{b}\|_2^2 \, \mathbf{e}_1^T f(T_m)\mathbf{e}_1$$

where T_m is computed via the Lanczos procedure applied to B with starting vector $\mathbf{b}$ corresponds precisely to the application of Gauss quadrature to a scalar problem.

Our main technical contributions appear in Section 3.3 and Section 3.4, where we demonstrate that the same framework can be used to compute the derivative of the trace of a matrix function in a low-rank direction and to efficiently update the trace under low-rank perturbations. That is, we propose the approximation

$$\frac{\partial}{\partial t}\left[\mathrm{Tr}(f(B + t\mathbf{b}\mathbf{b}^T))\right]_{t=0} \approx \|\mathbf{b}\|_2^2 \, \mathbf{e}_1^T f'(T_m)\mathbf{e}_1$$

and show that it similarly has an interpretation within the Gauss quadrature framework. We review the approximation

$$\mathrm{Tr}(f(B + \mathbf{b}\mathbf{b}) - f(B)) \approx \mathrm{Tr}(f(T_m + \|\mathbf{b}\|_2^2 \, \mathbf{e}_1\mathbf{e}_1^T) - f(T_m))$$

from [CKM22] and show that it has a similar interpretation through the lens of Gauss quadrature. This perspective improves on the bound in [CKM22] by removing dependence on the matrix dimension, which may be very large in practice.

3.1 Gauss quadrature and orthogonal polynomials

We begin by reviewing *scalar* Gauss quadrature. Consider computing the scalar Riemann-Stieltjes integral

$$\int_a^b f(x)d\alpha(x) \equiv \lim_{N\to\infty} \sum_{i=1}^{N} f(x_i)(\alpha(x_i) - \alpha(x_{i-1})) \tag{3.1}$$

where the x_i form a suitable partition of $[a, b]$, f is a continuous function defined on $[a, b]$, and α is a bounded, non-decreasing (though not necessarily continuous) function on $[a, b]$. An m-point numerical quadrature rule approximates this integral by some combination of m *nodes* $x_1, x_2, \cdots, x_m$ and m corresponding weights $w_1, \cdots, w_m$ such that

$$\int_a^b f(x)d\alpha(x) \approx \sum_{i=1}^{m} f(x_i)w_i. \tag{3.2}$$

When f is smooth, a tried and true approach is to use the Gauss nodes and weights associated with $d\alpha(x)$, which can be obtained from the orthogonal polynomials associated with $d\alpha$. Below, we introduce orthogonal polynomials and summarize how one may obtain the Gauss nodes together with some of their important theoretical properties. For brevity, we merely state these results, omitting many important (and beautiful!) details involved in the derivation, to which we refer the reader to [Sze39, Gau04, GM09]. We will extend the scalar problem to various matrix function problems in the following sections.

More concretely, the measure $d\alpha$, which we assume has $1 \leq N \leq \infty$ points of increase on the interval $[a, b]$, gives rise to an inner product and induced norm on the space of continuous functions

$$\langle f, g\rangle_{d\alpha} = \int_a^b f(x)g(x)d\alpha(x), \quad \|f\|_{d\alpha}^2 = \int_a^b f(x)^2 d\alpha(x). \tag{3.3}$$

We will use $\mathrm{Supp}(d\alpha)$ to indicate the *support* of $d\alpha$, the domain upon which α is increasing. In turn, this determines a family of *orthogonal polynomials*, which we denote for $k = 0, 1, 2, \cdots, N$ with $\pi_k(x; d\alpha)$, or simply π_k, for which each π_k is a polynomial of degree k and satisfies the orthonormality conditions

$$\langle \pi_i, \pi_j\rangle_{d\alpha} = \begin{cases} 0 & i \neq j, \\ 1 & i = j. \end{cases} \tag{3.4}$$

If α has an infinite number of points of increase, then the family of orthogonal polynomials is infinite, and if α has $N < \infty$ points of increase, then we consider the

finite family of polynomials through π_N. Notably, the polynomials $\{\pi_k\}_{k=0}^N$ can be generated recursively by a three-term recurrence

$$\pi_{i+1}(x) = \frac{1}{\beta_{i+1}} \left((x - \alpha_{i+1})\pi_i(x) + \beta_i \pi_{i-1}(x) \right), \quad i = 0, 1, 2, \cdots \tag{3.5}$$

where we define

$$\alpha_{i+1} = \langle x\pi_i, \pi_i \rangle_{d\alpha}, \quad \beta_{i+1} = \| (x - \alpha_{i+1})\pi_i - \beta_i \pi_{i-1} \|_{d\alpha} \tag{3.6}$$

together with the initial conditions $\beta_0 = 0$ and $\pi_0^{-1} = \sqrt{\int_a^b d\alpha(x)}$. Although it may seem that we have overloaded notation from Section 2.2, we will see in the next section that the Lanczos procedure introduced previously as Algorithm 4 computes the recurrence coefficients of a particular family of orthogonal polynomials, so the reuse of notation is fitting.

We are now ready to state the m point Gauss quadrature rule for (3.2). Suppose that we assemble the α_i and β_i into the symmetric tridiagonal matrix

$$T_m = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \alpha_{m-1} & \beta_{m-1} \\ & & \beta_{m-1} & \alpha_m \end{bmatrix}, \tag{3.7}$$

which is often called the *Jacobi matrix* associated with the orthogonal polynomials. Since T_m is symmetric and irreducible, it has m simple, real eigenvalues, which coincide with the m distinct roots of π_m, which are known to lie within $[a, b]$. See [GM09] for further details. We label the eigenvalues $t_1, t_2, \cdots, t_m$ with associated normalized eigenvectors $\mathbf{w}_1, \cdots, \mathbf{w}_m$. Define

$$\mu_0 \equiv \int_a^b d\alpha(x).$$

Then the choice

$$x_i \equiv t_i, \quad w_i \equiv \mu_0 (\mathbf{e}_1^T \mathbf{w}_i)^2 \tag{3.8}$$

in the approximation (3.2) yields *the m-point Gauss quadrature rule associated with the measure $d\alpha$*. That is, the nodes of Gauss quadrature are the eigenvalues of T_m and the weights are the squares of the first entries of the associated eigenvectors (rescaled by the size of the measure μ_0) [GM09, Theorem 6.2].[1] We then write

$$\int_a^b f(x) d\alpha(x) = \sum_{i=1}^m f(t_i) w_i + R_{GQ}^m[f, d\alpha], \tag{3.9}$$

[1] Often, this result is derived via interpolatory considerations [Tre19, Chapter 19].

where $R^m_{GQ}[f, d\alpha]$ is the *remainder term*. The following theorem summarizes why this is a successful approximation.

Theorem 3.1 (Gauss quadrature)**.** *Let $m \geq 1$. If f is a polynomial of degree at most $2m-1$,*

$$R^m_{GQ}[f, d\alpha] = 0. \tag{3.10}$$

If f is $2m$ times differentiable on $[a,b] \supset \mathrm{Supp}(d\alpha)$, there exists some $\xi \in [a,b]$ such that

$$R^m_{GQ}[f, d\alpha] = \mu_0 \frac{f^{(2m)}(\xi)}{(2m)!} \left(\beta_1 \beta_2 \cdots \beta_{m-1}\right)^2. \tag{3.11}$$

For any function f defined on $[a,b]$, the remainder satisfies the inequality

$$|R^m_{GQ}[f, d\alpha]| \leq 2\mu_0 E^\star_{2m-1}(f, S), \tag{3.12}$$

where $S \equiv \mathrm{Supp}(\mathrm{d}\alpha) \cup \mathrm{Zer}(\pi_m) \subset [a,b]$, and $E^\star_{2m-1}(f,S)$ is the error in the best uniform approximation to f on S (see section Section 2.3). Here, $\mathrm{Zer}(\pi_m)$ is the set of zeros of the polynomial π_m.

Proof. See [GM09, Chapter 6] for (3.10) and (3.11). The final result (3.12), which uses error estimates from uniform approximation, is more-or-less well-known (see e.g. [Gau81, Section 4.1.3]), and can be proven using the exactness result: for any $p \in \Pi_{2m-1}$, we have

$$\int_a^b p(x) d\alpha(x) = \sum_{i=1}^m p(x_i) w_i,$$

which immediately implies

$$\begin{aligned} R^m_{GQ}[f, d\alpha] &= \int_a^b f(x) d\alpha(x) - \sum_{i=1}^m f(x_i) w_i \\ &= \int_a^b f(x) d\alpha(x) - \int_a^b p(x) d\alpha(x) + \sum_{i=1}^m p(x_i) w_i - \sum_{i=1}^m f(x_i) w_i \\ &= \int_a^b (f-p)(x) d\alpha(x) + \sum_{i=1}^m (p-f)(x_i) w_i \end{aligned}$$

Notably, $\sum w_i = \mu_0$. Thus, taking absolute values and applying the triangle inequality yields

$$\begin{aligned} \left|R^m_{GQ}[f, d\alpha]\right| &\leq \left| \int_a^b (f-p)(x) d\alpha(x) \right| + \left| \sum_{i=1}^m (p-f)(x_i) w_i \right| \\ &\leq \int_a^b \left|(f-p)(x)\right| d\alpha(x) + \sum_{i=1}^m \left|(p-f)(x_i)\right| w_i \\ &\leq 2\mu_0 \max_{x \in \mathrm{Supp}(d\alpha) \cup \mathrm{Zer}(\pi_m)} \left|(f-p)(x)\right|. \end{aligned}$$

This result is true for arbitrary $p \in \mathbb{P}_{2m-1}$. Minimizing over $p \in \mathbb{P}_{2m-1}$ yields (3.12). □

This means that Gauss quadrature integrates polynomials of degree $2m-1$ exactly, and as a result of this property, approximately integrates functions that can be represented accurately by polynomials on the set S. Equation (3.11) gives an "exact" form of the remainder term, although it can be impractical for bounding the error since the $\xi \in [a, b]$ is unknown, and the bound involves high order derivatives of f. But we see in (3.12) that the convergence rate for the m-point Gauss quadrature rule is approximately twice that of the convergence rate of uniform approximation of f by polynomials, which can be incredibly fast for smooth f, as discussed in Section 2.3.

3.2 Quadratic forms

Now we turn to matrix functions: we wish to compute the quadratic form $\mathbf{b}^T f(B)\mathbf{b}$. Our motivation is two results: a connection between matrix function quadratic forms of the form $\mathbf{b}^T f(B)\mathbf{b}$ and Riemann-Stieltjes integrals, and a connection between the Lanczos procedure and orthogonal polynomials. That is, we will review the approximation

$$\mathbf{b}^T f(B)\mathbf{b} \approx \|\mathbf{b}\|_2^2 \, \mathbf{e}_1^T f(T_m)\mathbf{e}_1, \tag{3.13}$$

where T_m is generated by the Lanczos procedure (Algorithm 4). It turns out that the convergence properties of (3.13) are determined by the theory of Gauss quadrature discussed in the previous section. The material in this section is primarily drawn from [GM09]. See also [Che22].

Consider the following result:

Proposition 3.2. *Let B be a symmetric matrix with eigenvalues contained in an interval $[a, b]$, f a function defined on the eigenvalues of B, and $\mathbf{b}$ a vector. Then*

$$\mathbf{b}^T f(B)\mathbf{b} = \int_a^b f(x)\, d\alpha(x) \tag{3.14}$$

where $d\alpha$ is the measure determined by the piecewise constant function

$$\alpha(x) = \begin{cases} 0 & x \le \lambda_1 \\ \sum_{i=1}^{j} (\mathbf{e}_i^T V^T \mathbf{b})^2 & \lambda_i \le x < \lambda_{i+1} \\ \|\mathbf{b}\|_2^2 & \lambda_n < x, \end{cases} \tag{3.15}$$

with $\lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$ the eigenvalues of B with corresponding normalized eigenvectors $\{\mathbf{v}_i\}$ forming the columns of V.

Proof. Note that $B = V\Lambda V^T$, and then $f(B) = Vf(\Lambda)V^T$ by definition. So

$$\begin{aligned} \mathbf{b}^T f(B)\mathbf{b} &= \mathbf{b}^T V f(\Lambda) V^T \mathbf{b} \\ &= \sum_{i=1}^{n} f(\lambda_i)(\mathbf{e}_i^T V^T \mathbf{b})^2 \\ &= \int_a^b f(x) d\alpha(x) \end{aligned} \tag{3.16}$$

with α defined as in the statement of the proposition. □

Proposition 3.2 shows that the quadratic form $\mathbf{b}^T f(B)\mathbf{b}$ is exactly equivalent to a Riemann-Stieltjes integral whose weight function $d\alpha(x)$ is a discrete measure with support given by the eigenvalues of B and weights given by the expansion of $\mathbf{b}$ in the eigenvector basis V. Typically we do not know V since B may be a very large matrix and the computation (and storage) of all of its eigenvectors is likely too costly; consider instead the approximation of the integral by Gauss quadrature. This would require us to obtain the m nodes and weights of the quadrature rule. Fortunately, these are given precisely by the Lanczos iteration described in Algorithm 4.

Proposition 3.3. *The Jacobi matrix described in (3.7) for the measure $d\alpha$ given in (3.15) is equal to the symmetric tridiagonal matrix output by Algorithm 4 applied to the matrix B and vector $\mathbf{b}$. Moreover, the following equivalence holds:*

$$\mathbf{u}_j = \pi_{j-1}(B)\mathbf{b},$$

where the $\mathbf{u}_j$ are the orthonormal vectors output by the Lanczos procedure and the π_j are the orthonormal polynomials associated with the measure $d\alpha$.

Proof sketch. See [GM09] or [Che22]. The key observation is that according to Proposition 3.2

$$\langle f, g \rangle_{d\alpha} = \mathbf{b}^T f(A) g(A) \mathbf{b}.$$

Using this fact, the equivalence of the Lanczos recurrence ((2.33) and (2.34)) to the orthogonal polynomial recurrence ((3.5) and (3.6)) can be verified directly. The iterative computation of the orthogonal polynomials is often called the *Stieltjes procedure* [Gau04].

□

That is, the Lanczos procedure applied to a symmetric matrix B and a vector $\mathbf{b}$ outputs the Jacobi matrix associated with the measure described in Proposition 3.2. Combining the previous two results with our description of Gauss quadrature in

Theorem 3.1 explains the success of the Lanczos procedure for computing matrix function quadratic forms:

Theorem 3.4. *Suppose T_m is computed via the Lanczos procedure (Algorithm 4) applied to B with starting vector $\mathbf{b}$ and α is defined as in (3.15). Then*

$$\mathbf{b}^T f(B)\mathbf{b} = \|\mathbf{b}\|_2^2 \, \mathbf{e}_1^T f(T_m)\mathbf{e}_1 + R_{GQ}^m[f, d\alpha]. \tag{3.17}$$

As a result, if f is a polynomial of degree $2m-1$,

$$\mathbf{b}^T f(B)\mathbf{b} = \|\mathbf{b}\|_2^2 \, \mathbf{e}_1^T f(T_m)\mathbf{e}_1. \tag{3.18}$$

If f is $2m$-times differentiable on $[a,b] \supset \lambda(B)$, then there exists $\xi \in [a,b]$ with

$$\mathbf{b}^T f(B)\mathbf{b} - \|\mathbf{b}\|_2^2 \, \mathbf{e}_1^T f(T_m)\mathbf{e}_1 = \|\mathbf{b}\|_2^2 \frac{f^{(2m)}(\xi)}{(2m)!}(\beta_1\beta_2\cdots\beta_{m-1})^2. \tag{3.19}$$

Moreover, the following error bound holds:

$$\left|\mathbf{b}^T f(B)\mathbf{b} - \|\mathbf{b}\|_2^2 \, \mathbf{e}_1^T f(T_m)\mathbf{e}_1\right| \leq 2\,\|\mathbf{b}\|_2^2 \, E_{2m-1}^{\star}(f, S), \tag{3.20}$$

where S is any set containing the spectrum of B and the spectrum of T_m, i.e. $S \supset \lambda(B) \cup \lambda(T_m)$. In particular, the bound holds with $S = [a,b]$.

Proof. Using Proposition 3.2 and (3.9), we have

$$\mathbf{b}^T f(B)\mathbf{b} = \int_a^b f(x)d\alpha(x) = \sum_{i=1}^m f(t_i)w_i + R_{GQ}^m[f, d\alpha]$$

for t_i and w_i determined by the Jacobi matrix associated with $d\alpha$. Moreover, Proposition 3.3 shows that T_m output by the Lanczos procedure is the correct Jacobi matrix for $d\alpha$. Finally, the result (3.17) is obtained from the observation

$$\sum_{i=1}^m f(t_i)w_i = \mu_0 \sum_{i=1}^m f(t_i)(\mathbf{e}_1^T\mathbf{w}_i)^2 = \|\mathbf{b}\|^2 \, \mathbf{e}_1^T f(T_m)\mathbf{e}_1,$$

where we have simplified $\mu_0 = \|\mathbf{b}\|^2$ and $\sum_{i=1}^m f(t_i)(\mathbf{e}_1^T\mathbf{w}_i)^2 = \mathbf{e}_1^T f(T_m)\mathbf{e}_1$. The remainder term (3.19) and the bound (3.20) are restatements of Theorem 3.1 □

Remark 3.1. Combining (3.20) with Theorem 2.3 or Proposition 2.4 provides more concrete error bounds. A similar result is presented in [UCS17] for analytic functions, however the final bound depends on the spread of the eigenvalues $\lambda_n - \lambda_1$ which appears to be unnecessary, and the use of the discrete set S in (3.20) explains in part why the procedure exhibits spectral adaptivity.

This suggests the following algorithm for computing $\mathbf{b}^T f(B)\mathbf{b}$: compute T_m using Algorithm 4, then estimate $\mathbf{b}^T f(B)\mathbf{b} \approx \|\mathbf{b}\|^2 \mathbf{e}_1^T f(T_m)\mathbf{e}_1$. We state such a procedure in Algorithm 6. Importantly, Algorithm 6 accesses B only implicitly through matrix-vector products in the Lanczos implementation of Algorithm 4. Moreover, the call to the Lanczos procedure is the costliest component of the algorithm since the symmetric eigenvalue problem can be reliably solved in $O(m^3)$ time (or better if a tridiagonal eigensolver is available), which has no dependence on $n \gg m$.

Algorithm 6 LanczosQuad$(f, B, \mathbf{b}, m)$
Lanczos quadrature approximation for $\mathbf{b}^T f(B)\mathbf{b}$

1: **Input:** Scalar function f, Symmetric matrix $B \in \mathbb{R}^{n\times n}$, initial vector $\mathbf{b} \in \mathbb{R}^n$, iteration count m.
2: **Output:** $t \approx \mathbf{b}^T f(B)\mathbf{b}$
3: $(U_m, T_m, \beta_{m+1}, \mathbf{u}_{m+1}) \leftarrow$ Lanczos$(B, \mathbf{b}, m)$ ▷ Algorithm 4
4: $t \leftarrow \|\mathbf{b}\|^2 \cdot \mathbf{e}_1^T f(T_m)\mathbf{e}_1$
5: **Complexity summary:** $O(m^2 n) + O(m \cdot \operatorname{nnz} B)$

3.3 Rank-one derivatives of the trace

Now suppose we wish to compute the derivative of the trace of a matrix function in a rank-one direction, which is equivalent to the trace of the Frechet derivative, so we may write the quantity of interest

$$\frac{\partial}{\partial t}\left[\operatorname{Tr}(f(B + t\mathbf{b}\mathbf{b}^T))\right]_{t=0} = \operatorname{Tr}(L_f(B, \mathbf{b}\mathbf{b}^T)). \tag{3.21}$$

In this section, we introduce the approximation

$$\operatorname{Tr}(L_f(B, \mathbf{b}\mathbf{b}^T)) \approx \|\mathbf{b}\|^2 \mathbf{e}_1^T f'(T_m)\mathbf{e}_1. \tag{3.22}$$

To our knowledge, this has not been considered in the literature previously. The resulting Algorithm 7 may be viewed as a modification of the more general Algorithm 9 by taking the trace of the output of Algorithm 9. Originally, Algorithm 9 has been studied in [Kre19] and [KKRS21]; however, no discussion of the computation of the trace or its improved convergence properties are discussed. Our results connect the problem to Gauss quadrature and show that there is a concrete difference in convergence for the trace versus the full matrix according to this connection. In the next section, we will further use this result to analyze rank-one updates of the trace of a matrix function.

Our main observation in this section is the following:

Proposition 3.5. *Let f be differentiable on the eigenvalues of B. Then*

$$\mathrm{Tr}(L_f(B, \mathbf{b}\mathbf{b}^T)) = \mathbf{b}^T f'(B)\mathbf{b}. \tag{3.23}$$

Proof. We may assume that f is a polynomial of degree $2n-1$ since $f(B) = p(B)$ and $L_f(B, \mathbf{b}\mathbf{b}^T) = L_p(B, \mathbf{b}\mathbf{b}^T)$ if $f(\lambda_i) = p(\lambda_i)$ and $f'(\lambda_i) = p'(\lambda_i)$ for each eigenvalue λ_i of B. We may write

$$f(x) = \sum_{i=0}^{2n-1} f_i x^i, \quad f'(x) = \sum_{i=1}^{2n-1} i f_i x^{i-1}.$$

Recall from (2.9) that

$$L_f(B, \mathbf{b}\mathbf{b}^T) = \sum_{i=1}^{2n-1} f_i \sum_{k=1}^{i} B^{k-1}\mathbf{b}\mathbf{b}^{\mathbf{T}} B^{i-k},$$

so that taking the trace of both sides and applying elementary properties the trace[2] we obtain

$$\begin{aligned}
\mathrm{Tr}(L_f(B, \mathbf{b}\mathbf{b}^T)) &= \mathrm{Tr}\left(\sum_{i=1}^{2n-1} f_i \sum_{k=1}^{i} B^{k-1}\mathbf{b}\mathbf{b}^{\mathbf{T}} B^{i-k}\right) \\
&= \sum_{i=1}^{2n-1} f_i \sum_{k=1}^{i} \mathrm{Tr}\left(B^{k-1}\mathbf{b}\mathbf{b}^{\mathbf{T}} B^{i-k}\right) \\
&= \sum_{i=1}^{2n-1} f_i \sum_{k=1}^{i} \mathrm{Tr}\left(\mathbf{b}^{\mathbf{T}} B^{i-1}\mathbf{b}\right) \\
&= \sum_{i=1}^{2n-1} i f_i \mathbf{b}^{\mathbf{T}} B^{i-1}\mathbf{b} \\
&= \mathbf{b}^T \left(\sum_{i=1}^{2n-1} i f_i B^{i-1}\right)\mathbf{b} \\
&= \mathbf{b}^T f'(B)\mathbf{b}.
\end{aligned}$$

This completes the proof. □

Remark 3.2. Similar results appear in [Sch23], though the significance of the connection for the analysis of computation of the trace (i.e. convergence rate, connections to quadrature) is not discussed.

[2] In particular, we use linearity of the trace, the cyclic property of the trace, and that the trace of a scalar is itself.

This will allow us to compute the sensitivity of the trace using quadratic forms from before. We can state an algorithm for computing directional derivatives of the trace via Gauss quadrature, supposing we have access to the ordinary derivative f'. We state such an algorithm in Algorithm 7. It is equivalent to $\textsc{LanczosQuad}(f', B, \mathbf{b}, m)$ computed via Algorithm 6.

Algorithm 7 $\textsc{DynamicLanczosQuad}(f, B, \mathbf{b}, m)$
Dynamic Lanczos quadrature approximation

1: **Input:** Scalar function f, Symmetric matrix $B \in \mathbb{R}^{n\times n}$, initial vector $\mathbf{b} \in \mathbb{R}^n$, iteration count m.
2: **Output:** $t \approx \operatorname{Tr}(L_f(B, \mathbf{b}\mathbf{b}^T))$
3: Compute f'.
4: $t \leftarrow \textsc{LanczosQuad}(f', B, \mathbf{b}, m)$ ▷ Algorithm 6.
5: **Complexity summary:** $O(m^2 n) + O(m \cdot \operatorname{nnz} B)$

The important observation here is that all of the results from Theorem 3.4 apply here. That is, we may write the following characterization theorem:

Theorem 3.6. *For α given as in (3.15),*

$$\operatorname{Tr}(L_f(B, \mathbf{b}\mathbf{b}^T)) = \|\mathbf{b}\|^2 \mathbf{e}_1^T f'(T_m)\mathbf{e}_1 + R_{GQ}^m[f', d\alpha]. \tag{3.24}$$

If f is a polynomial of degree at most $2m$, then

$$\operatorname{Tr}(L_f(B, \mathbf{b}\mathbf{b}^T)) = \|\mathbf{b}\|^2 \mathbf{e}_1^T f'(T_m)\mathbf{e}_1. \tag{3.25}$$

Moreover, if f is $2m+1$-times differentiable on an interval $[a, b]$ containing the spectrum of B, then there exists $\xi \in [a, b]$ such that

$$\operatorname{Tr}(L_f(B, \mathbf{b}\mathbf{b}^T)) = \|\mathbf{b}\|^2 \mathbf{e}_1^T f'(T_m)\mathbf{e}_1 + \|\mathbf{b}\|^2 \frac{f^{(2m+1)}(\xi)}{(2m+1)!}(\beta_1 \cdots \beta_{m-1})^2. \tag{3.26}$$

In general, the remainder satisfies the bound

$$\left|\operatorname{Tr}(L_f(B, \mathbf{b}\mathbf{b}^T)) - \|\mathbf{b}\|^2 \mathbf{e}_1^T f'(T_m)\mathbf{e}_1\right| \leq 2\|\mathbf{b}\|^2 E_{2m}^\star(f', S), \tag{3.27}$$

where $S \supset \lambda(B) \cup \lambda(T_m)$, and $E_{2m}^\star(f', S)$ is the error in the best uniform approximation to f' on S.

This result shows that directional derivatives of the trace of a matrix function can be computed by the exact same quadrature methods that have become popular for the quadratic form $\mathbf{b}^T f(B)\mathbf{b}$. Consequently, the convergence rate of Algorithm 7 is proportional to $E_{2m}(f', S)$, rather than $E_m(f', S)$ as will be the case for the more general method Algorithm 9 discussed in the next chapter and in [Kre19, KKRS21]. Moreover, we will see in Chapter 7 that Algorithm 7 has a direct application in network science for computing measurements of edge importance.

3.4 Rank-one updates of the trace

The last problem that we introduce in this chapter is the computation of low-rank updates of the trace. We are interested in the approximation

$$\mathrm{Tr}(f(B + \mathbf{b}\mathbf{b}^T) - f(B)) \approx \mathrm{Tr}(f(T_m + \|\mathbf{b}\|^2 \mathbf{e}_1 \mathbf{e}_1^T) - f(T_m)). \tag{3.28}$$

This approximation stems from the more general algorithm presented in [BKS18, BCKS21] for computing the rank-one update $f(B + \mathbf{b}\mathbf{b}^T) - f(B)$; however, it is observed in the later publication [CKM22] that the trace approximation (3.28) obtains twice the convergence rate of the more general method. Our goal here is to elucidate this doubled convergence rate by connecting the approximation (3.28) directly to the approximation (3.22), and applying Gauss quadrature theory to obtain improved convergence theory. We will compare our result to the bound in [CKM22] at the end of this section.

Within our main result, we will use the following property of matrix perturbation theory [Ste90, Theorem 2.3]:

Theorem 3.7 (Perturbation of eigenvalues)**.** *Let λ be an isolated eigenvalue of B with associated left and right eigenvectors $\mathbf{x}$ and $\mathbf{y}$. Then for a perturbation matrix E with $\|E\|$ sufficiently small, there exists a unique eigenvalue $\tilde{\lambda}$ of $B + E$ such that*

$$\tilde{\lambda} = \lambda + \frac{\mathbf{x}^T E \mathbf{y}}{\mathbf{x}^T \mathbf{y}} + O(\|E\|^2). \tag{3.29}$$

Remark 3.3. We also remark that the eigenvectors $\mathbf{x}, \mathbf{y}$ also have perturbative expansions $\tilde{\mathbf{x}}, \tilde{\mathbf{y}}$, although we will not need explicit formulae. Further details are given in [Wil65], for example.

As a consequence, for a symmetric matrix B with distinct eigenvalues, $\lambda_1, \cdots, \lambda_n$ and corresponding normalized eigenvectors $\mathbf{v}_1, \cdots, \mathbf{v}_n$, we can also define $B(t) = B + t\mathbf{b}\mathbf{b}^T$ with parameter dependent eigenvalues $\lambda_i(t)$ and eigenvectors $\mathbf{v}_i(t)$ that are continuously differentiable in t, and we may quantify the derivative of the eigenvalues with respect to the perturbation $\mathbf{b}\mathbf{b}^T$

$$\lambda_i'(t) = (\mathbf{v}_i(t)^T \mathbf{b})^2. \tag{3.30}$$

This characterization holds due to Theorem 3.7 as long as $B(t)$ has distinct eigenvalues. We also make the following observation about the Lanczos procedure applied to $B(t)$:

Lemma 3.8. *Let T_m be the output of m-steps of the Lanczos procedure applied to B and $\mathbf{b}$. Define $T_m(t) = T_m + t\|\mathbf{b}\|^2 \mathbf{e}_1\mathbf{e}_1^T$ for $t \in \mathbb{R}$. Then $T_m(t)$ results from m-steps of the Lanczos procedure applied to the matrix $B(t) = B + t\mathbf{b}\mathbf{b}^T$ and starting vector $\mathbf{b}$.*

Proof. Let U_m, T_m be the output of the Lanczos procedure appleid to $B, \mathbf{b}$, and $\tilde{U}_m, \tilde{T}_m$ the result of Lanczos applied to $B(t), \mathbf{b}$. Note that it is sufficient to show $\tilde{U}_m = U_m$, since in that case

$$\tilde{T}_m = \tilde{U}_m^* B(t)\tilde{U}_m = U_m^* B U_m + t U_m^* \mathbf{b}\mathbf{b}^T U_m = T_m + t\|\mathbf{b}\|^2 \mathbf{e}_1\mathbf{e}_1^T.$$

We prove $\tilde{U}_m = U_m$ inductively. Note that $\mathbf{u}_1 = \mathbf{b}/\|\mathbf{b}\| = \tilde{\mathbf{u}}_1$. Now suppose $U_{m-1} = \tilde{U}_{m-1}$ with $m > 1$. Recall that $(I - U_{m-1}U_{m-1}^*)$ is an orthogonal projection onto the compliment of U_{m-1}. Since the three-term recurrence of the Lanczos procedure is equivalent to the Gram-Schmidt procedure and $U_{m-1} = \tilde{U}_{m-1}$ we may write

$$\mathbf{u}_m = \frac{(I - U_{m-1}U_{m-1}^*)B\mathbf{u}_{m-1}}{\left\|(I - U_{m-1}U_{m-1}^*)B\mathbf{u}_{m-1}\right\|}, \quad \tilde{\mathbf{u}}_m = \frac{(I - U_{m-1}U_{m-1}^*)B(t)\mathbf{u}_{m-1}}{\left\|(I - U_{m-1}U_{m-1}^*)B(t)\mathbf{u}_{m-1}\right\|}.$$

But since $\mathbf{b}^T\mathbf{u}_{m-1} = \mathbf{u}_1^T\mathbf{u}_{m-1} = 0$,

$$\begin{aligned}(I - U_{m-1}U_{m-1}^*)B(t)\mathbf{u}_{m-1} &= (I - U_{m-1}U_{m-1}^*)B\mathbf{u}_{m-1} + t(I - U_{m-1}U_{m-1}^*)\mathbf{b}\mathbf{b}^T\mathbf{u}_{m-1} \\ &= (I - U_{m-1}U_{m-1}^*)B\mathbf{u}_{m-1},\end{aligned}$$

which implies $\mathbf{u}_m = \tilde{\mathbf{u}}_m$. □

Now we may state the main result of this section:

Theorem 3.9. *Define $B(t) = B + t\mathbf{b}\mathbf{b}^T$ and correspondingly the output of the Lanczos procedure applied to $B(t)$ and $\mathbf{b}$ with $T_m(t) = T_m + t\|\mathbf{b}\|^2\mathbf{e}_1\mathbf{e}_1^T$. Suppose f is differentiable on an interval containing $\Lambda(B(t))$ for all $0 \le t \le 1$. Let $d\alpha_t$ be the measure associated with each $B(t)$ as defined in (3.15). Then*

$$\mathrm{Tr}(f(B+\mathbf{b}\mathbf{b}^T) - f(B)) = \mathrm{Tr}(f(T_m + \|\mathbf{b}\|^2\mathbf{e}_1\mathbf{e}_1^T) - f(T_m)) + \int_0^1 R_{GQ}^m[f', d\alpha_t]dt. \quad (3.31)$$

Consequently, if f is a polynomial of degree $2m$, then

$$\mathrm{Tr}(f(B + \mathbf{b}\mathbf{b}^T) - f(B)) = \mathrm{Tr}(f(T_m + \|\mathbf{b}\|^2\mathbf{e}_1\mathbf{e}_1^T) - f(T_m)). \quad (3.32)$$

If $f^{(2m+1)}$ exists on an interval $[a, b]$ containing $\lambda(B(t))$ for each $0 \le t \le 1$, then there exists $\xi(t) \in [a, b]$ for each $t \in [0, 1]$ such that

$$\begin{aligned}\mathrm{Tr}(f(B + \mathbf{b}\mathbf{b}^T) - f(B)) &= \mathrm{Tr}(f(T_m + \|\mathbf{b}\|^2\mathbf{e}_1\mathbf{e}_1^T) - f(T_m)) \quad (3.33)\\ &+ \frac{\|\mathbf{b}\|^2(\beta_1\beta_2\cdots\beta_{m-1})^2}{(2m+1)!}\int_0^1 f^{(2m+1)}(\xi(t))dt. \quad (3.34)\end{aligned}$$

Finally, defining the remainder

$$r_m = \mathrm{Tr}(f(B + \mathbf{b}\mathbf{b}^T) - f(B)) - \mathrm{Tr}(f(T_m + \|\mathbf{b}\|^2 \mathbf{e}_1\mathbf{e}_1^T) - f(T_m)), \tag{3.35}$$

we obtain the bounds

$$|r_m| \leq 2\|\mathbf{b}\|^2 \max_{t\in[0,1]} E_{2m}^{\star}(f', S_t) \tag{3.36}$$

for $S_t \supset \lambda(B(t)) \cup \lambda(T_m(t))$.

Proof. We prove (3.31). Note that the subsequent remainder characterizations and error bounds are straightforward modifications of the results in Theorem 3.1 and Theorem 3.6.

The fundamental theorem of calculus for Frechet spaces allows us to write

$$f(B + \mathbf{b}\mathbf{b}^T) - f(B) = \int_0^1 L_f(B(t), \mathbf{b}\mathbf{b}^T)dt. \tag{3.37}$$

Taking the trace of each side, we obtain

$$\begin{aligned} \mathrm{Tr}(f(B + \mathbf{b}\mathbf{b}^T) - f(B)) &= \mathrm{Tr}\left(\int_0^1 L_f(B(t), \mathbf{b}\mathbf{b}^T)dt\right) \\ &= \int_0^1 \mathrm{Tr}(L_f(B(t), \mathbf{b}\mathbf{b}^T))dt \\ &= \int_0^1 \mathbf{b}^T f'(B(t))\mathbf{b}dt, \end{aligned} \tag{3.38}$$

where in the final line we have applied Proposition 3.5. Applying Gauss quadrature to the quadratic form within the integral, we see

$$\int_0^1 \mathbf{b}^T f'(B(t))\mathbf{b}dt = \int_0^1 \|\mathbf{b}\|^2 \sum_{i=1}^m f'(s_i(t))(\mathbf{e}_1^T\mathbf{w}_i(t))^2 dt + \int_0^1 R_{GQ}[f', d\alpha_t]dt. \tag{3.39}$$

Here, $s_i(t)$ is the ith eigenvalue of $T_m(t)$ and $\mathbf{w}_i(t)$ is the corresponding eigenvector. It remains to show that

$$\int_0^1 \|\mathbf{b}\|^2 \sum_{i=1}^m f'(s_i(t))(\mathbf{e}_1^T\mathbf{w}_i(t))^2 dt = \mathrm{Tr}(f(T_m + \|\mathbf{b}\|^2 \mathbf{e}_1\mathbf{e}_1^T) - f(T_m)). \tag{3.40}$$

Denote by $s_i(t)$ the ith eigenvalue of $T_m(t)$ with an associated normalized eigenvector $\mathbf{w}_i(t)$. Importantly, for all t, $T_m(t)$ has distinct eigenvalues as it is an irreducible, symmetric, tridiagonal matrix. Then $s_i(t)$ and $\mathbf{w}_i(t)$ are differentiable according to Theorem 3.7. Now note

$$\begin{aligned} \mathrm{Tr}(f(T_m + \|\mathbf{b}\|^2 \mathbf{e}_1\mathbf{e}_1^T) - f(T_m)) &= \sum_{i=1}^m f(s_i(1)) - f(s_i(0)) \\ &= \sum_{i=1}^m \int_0^1 f'(s_i(t))s_i'(t)dt. \end{aligned} \tag{3.41}$$

But $s_i'(t)$ is the rate of change of the ith eigenvalue of $T_m(t)$ under the perturbation $t\|\mathbf{b}\|^2 \mathbf{e}_1\mathbf{e}_1^T$, and is according to Theorem 3.7 given by $s_i'(t) = \|\mathbf{b}\|^2 (\mathbf{w}_i(t)^T\mathbf{e}_1)^2$. Substituting this value for $s_i'(t)$ and interchanging the integral with the sum gives the desired equivalence. □

To be clear, we have just shown that the approximation

$$\operatorname{Tr}(f(B+\mathbf{b}\mathbf{b}^T) - f(B)) \approx \operatorname{Tr}(f(T_m + \|\mathbf{b}\|^2\mathbf{e}_1\mathbf{e}_1^T) - f(T_m))$$

is equivalent to integrating the Gauss quadrature approximation from the previous section exactly. Crucially, this characterization of the approximation (3.28) is different from that given in [CKM22], where the analysis is exclusively based on exactness properties of the algorithm. In fact, the bound presented in [CKM22] restated in our notation is as follows:

$$|r_m| \leq 4nE_{2m}^{\star}(f,[a,b]), \tag{3.42}$$

where n is the dimension of the matrix (i.e. B is a square n-by-n matrix) and $[a,b]$ is an interval containing the eigenvalues of B. We may compare (3.42) to either (3.36). Both bounds depend essentially on the rate of uniform polynomial approximation of f (or f') on a bounded domain of the real line; however, there are two important distinctions to be made. First, the bounds presented here do not depend on n, the dimension of the input matrix, which is significant since these methods are intended for very large matrices with n in the thousands, millions, or even billions. Second, in our final bound, we do not relax the domain of approximation to the interval $[a,b]$, as we may use the discussion surrounding Proposition 2.4 in order to characterize the effect that outliers have on the approximation. We remark, however, that a simple modification of the proof in [CKM22] could also make this accommodation (but without removing dependence on n). We include in Algorithm 8 a variant of our Lanczos-based quadrature algorithms for computing rank-one updates to the trace.

Algorithm 8 IntLanczosQuad$(f, B, \mathbf{b}, m)$
Integrated dynamic Lanczos quadrature for updating the trace of a matrix function.

1: **Input:** Scalar function f, Symmetric matrix $B \in \mathbb{R}^{n\times n}$, initial vector $\mathbf{b} \in \mathbb{R}^n$, iteration count m.
2: **Output:** $t \approx \operatorname{Tr}(f(B+\mathbf{b}\mathbf{b}^T) - f(B))$
3: $(U_m, T_m, \beta_{m+1}, \mathbf{u}_{m+1}) \leftarrow$ Lanczos$(B, \mathbf{b}, m)$ ▷ Algorithm 4
4: $t \leftarrow \operatorname{Tr}(f(T_m + \|\mathbf{b}\|^2\mathbf{e}_1\mathbf{e}_1^T) - f(T_m))$
5: **Complexity summary:** $O(m^2 n) + O(m \cdot \operatorname{nnz} B)$

3.5 Example

In this chapter, we described three matrix function approximation problems with algorithms that are equivalent to Gauss quadrature. Let us conclude with some numerical examples illustrating that the convergence curves for each is effectively the same as that of Gauss quadrature for the related scalar problem. The results are shown in Figure 3.1. Below, we describe the experiment.

We consider the functions $f(x) = e^{-x^2}$, $g(x) = (1 + 25^2)^{-1}$, and $h(x) = |x|^3$ on the interval $[-1, 1]$.[3] Notably, these are entire, analytic, and twice differentiable, respectively. We approximate the integrals

$$I_1 = \int_{-1}^{1} f(x)dx, \quad I_2 = \int_{-1}^{1} g(x)dx, \quad I_3 = \int_{-1}^{1} h(x)dx, \tag{3.43}$$

together with

$$I_1' = \int_{-1}^{1} f'(x)dx, \quad I_2' = \int_{-1}^{1} g'(x)dx, \quad I_3' = \int_{-1}^{1} h'(x)dx, \tag{3.44}$$

by Gauss-Legendre quadrature.[4] We also form two random symmetric matrices $B_1 \in \mathbb{R}^{1000\times 1000}$ with eigenvalues densely distributed in $[-1, 1]$ and a normalized random vector $\mathbf{b}$. We form a second random matrix $B_2 \in \mathbb{R}^{1000\times 1000}$ with eigenvalues in $[-1/2, 1/2]$, and then we perturb B_2 by adding $0.5(\mathbf{v}_1\mathbf{v}_1^T - \mathbf{v}_n\mathbf{v}_n^T)$, where $\mathbf{v}_1$ and $\mathbf{v}_n$ are the eigenvectors associated with the largest and smallest eigenvalues of B_2, so that the resulting B_2 has an eigenvalue distribution that is approximately distributed as $[-1/2, 1/2] \cup \{-1, 1\}$. Now we form the matrix function approximations described in this chapter—(3.13), (3.22), and (3.28)—for each function f, g, and h, and we monitor the error in the approximations. Notably, quadratic forms exhibit similar convergence properties to the I_1, I_2, I_3, whereas the dynamic algorithms exhibit behavior similar to I_1', I_2', I_3'. Additionally, we can observe that convergence for the matrix problems associated with B_2 is drastically improved for analytic functions due to the difference between approximation on $[-1, 1]$ vs $[-1/2, -1/2]\cup\{-1, 1\}$ and the implicit adaptation of Lanczos to this difference.

[3] Actually, we introduce a small shift $\epsilon = 0.1$ so that the functions are not symmetric (which would cause odd versus even approximants to be very different). We work with $f(x - \epsilon)$ and so on.

[4] That is, the nodes and weights are given by the Legendre polynomials, the orthogonal polynomials associated with the ordinary measure $d\alpha(x) = dx$.

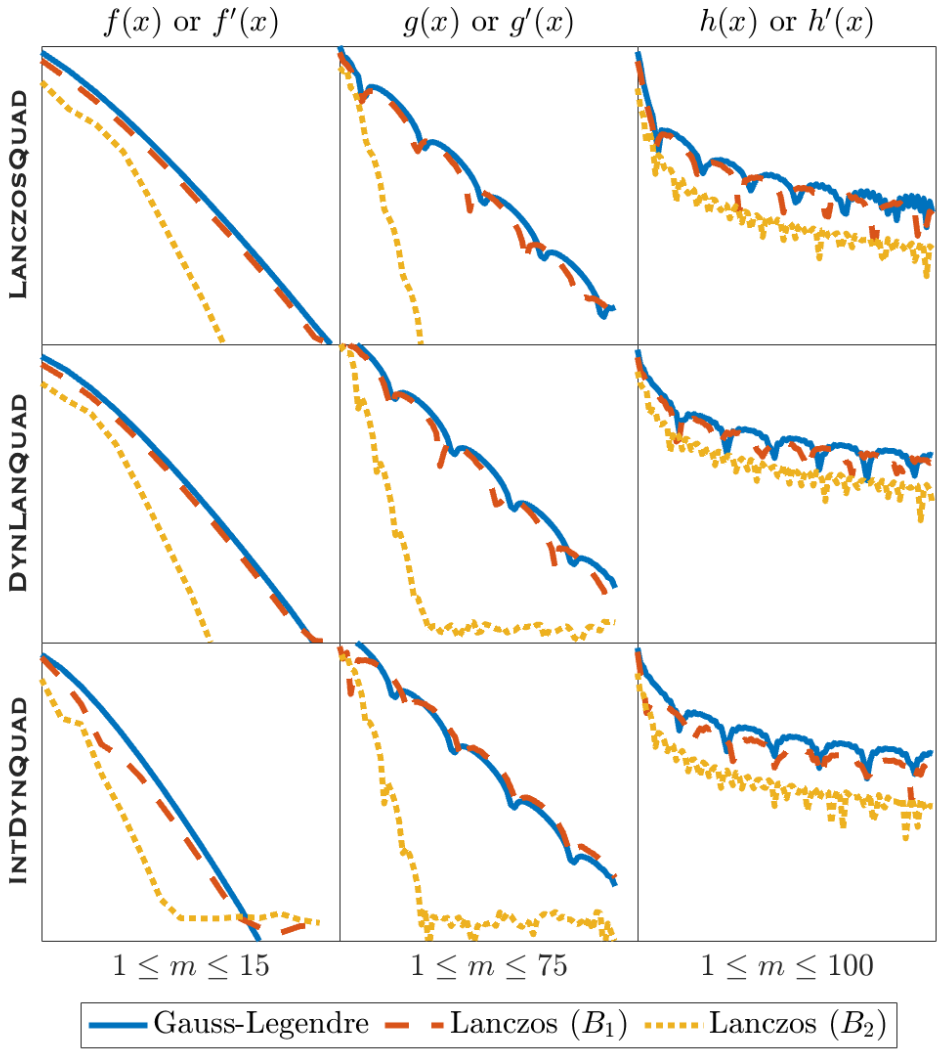

Figure 3.1: Error plots for Lanczos quadrature. This figure accompanies the discussion in Section 3.5. Each y axis is approximation error on a log-scale. Each x axis gives the number of iterates in the Lanczos procedure or the number of nodes in the Gauss-Legendre numerical quadrature approximation.

Chapter 4

Lanczos interpolation

Chapter overview

If the last chapter was about *quadrature*, then this chapter is about *interpolation*. We continue the study of matrix function approximants obtained from the Lanczos procedure and their theoretical connections to scalar polynomial approximation. We will review briefly how one can approximate a scalar function using polynomial interpolation, emphasizing that the zeros of orthogonal polynomials provide a good interpolating set. We also review the well-known characterization of

$$f(B)\mathbf{b} \approx \|\mathbf{b}\|_2 \, U_m f(T_m)\mathbf{e}_1 \tag{4.1}$$

as forming a particular interpolating polynomial [Saa92]. Our primary technical contributions come in the following sections within which we provide analogous characterizations for the approximation of the Frechet derivative in a rank-one direction (proposed in [Kre19, KKRS21])

$$L_f(B, \mathbf{b}\mathbf{b}^T) \approx \|\mathbf{b}\|_2^2 \, U_m L_f(T_m, \mathbf{e}_1\mathbf{e}_1^T)U_m^T \tag{4.2}$$

and approximation of rank-one updates of the matrix function (proposed in [BKS18]) via

$$f(B + \mathbf{b}\mathbf{b}^T) - f(B) \approx U_m \left(f(T_m + \|\mathbf{b}\|_2^2 \, \mathbf{e}_1\mathbf{e}_1^T) - f(T_m) \right) U_m^T. \tag{4.3}$$

We show these latter two approximations are precisely equivalent to certain bivariate matrix functions from interpolating bivariate polynomials of Δf.

4.1 Scalar polynomial interpolation

We begin with a review of polynomial interpolation. The key property that we wish to communicate is that interpolation on the zeros of the orthogonal polynomials is nearly optimal in a relevant norm.

Let $X = \{x_1, x_2, \cdots, x_m\} \subset [a, b]$ be a finite collection of distinct real numbers, and suppose that we have the function data $f(x_1), f(x_2), \cdots, f(x_m)$. It is a standard result in polynomial approximation theory (see e.g. [Tre19, Chapter 5]) that there exists a unique polynomial of degree at most $m-1$ interpolating the data. This polynomial is called the *Lagrange interpolating polynomial*, which we will denote l_X^f, and it is a polynomial uniquely determined by the interpolation conditions

$$l_X^f(x_i) = f(x_i), \quad i = 1, 2, \cdots, m \tag{4.4}$$

together with the restriction that l_X^f be of degree at most $m-1$.

Although l_X^f interpolates f at the x_i, it is not generally true that, for large m, $l_X^f \approx f$ on $[a, b]$ pointwise—even when f is a smooth function. Polynomial interpolation is a delicate affair, and the interpolatory nodes must be chosen carefully. At the same time, there are certain sets of nodes that are excellent.[1] Here, we make the following observation: *interpolation on the zero sets of orthogonal polynomials is nearly optimal in the $\|\cdot\|_{d\alpha}$ norm.* Our goal in this section is to formalize this observation, and in the subsequent sections we will provide analogous results for matrix function problems.

Often, it is convenient to write l_X^f in an alternate polynomial basis. For $i = 1, 2, \cdots, m$, let $l_i(x)$ be the polynomial of degree $m-1$ defined by $l_i(x_i) = 1$ and $l_i(x_j) = 0$ when $i \neq j$, which exists and is unique by the same reasoning as l_X^f. The $l_i(x)$ are often called the *fundamental polynomials of Lagrange interpolation.* Then we may write

$$l_X^f(x) = \sum_{i=1}^{m} l_i(x) f(x_i). \tag{4.5}$$

More generally, the polynomials $\{l_i(x)\}_{i=1}^m$ are linearly independent and form a basis for any polynomial of degree $m-1$. The following result shows that when the x_i are chosen to be the zeros of π_m, it turns out that we have obtained an alternate orthogonal basis for polynomials of degree $m-1$.

Theorem 4.1 (Orthogonality of fundamental Lagrange polynomials)**.** *Let $X = \mathrm{Zer}(\pi_m) = \{x_1, x_2, \cdots, x_m\}$ be the zeros of the degree m orthogonal polynomial associated with the measure $d\alpha$ supported on $[a, b]$, and let the $l_i(x)$ be the fundamental polynomials of Lagrange interpolation such that $l_i(x_j) = \delta_{ij}$. Then*

$$\langle l_i, l_j \rangle_{d\alpha} = w_i \delta_{ij}, \quad i, j = 1, 2, \cdots, m,$$

where the w_i are the Gauss quadrature weights associated with the x_i.

[1] For uniform approximation, one can use the *Chebyshev nodes* for interpolation [Tre19], which is the foundation of the Chebfun software package.

Proof. See [Sze39, Theorem 14.2.1]. □

With this result in hand, we consider the interpolating polynomial $l^f_{\mathrm{Zer}(\pi_m)}$ as an approximation to f. Recall that we may expand any sufficiently smooth function f on $\mathrm{Supp}(d\alpha)$ in terms of its Fourier coefficients:

$$f(x) = \sum_{i=0}^{\infty} f_i \pi_i(x), \quad f_i \equiv \langle f, \pi_i \rangle_{d\alpha}.$$

Of course, this immediately implies a simple characterization of the norm of f in terms of the squares of its Fourier coefficients,

$$\|f\|^2_{d\alpha} = \sum_{i=0}^{\infty} f_i^2.$$

Moreover, the optimal degree m approximation to f in the $\|\cdot\|_{d\alpha}$ norm is the projection of f onto the orthonormal basis $\{\pi_i\}$, obtained formally by truncating the series above. That is, the minimizer is given by

$$\operatorname*{argmin}_{p \in \mathbb{P}_m} \|f - p\|^2_{d\alpha} = \sum_{i=0}^{m} f_i \pi_i(x),$$

and the error associated with it is

$$\min_{p \in \mathbb{P}_m} \|f - p\|^2_{d\alpha} = \sum_{i=m+1}^{\infty} f_i^2.$$

Computation of the projection coefficients $f_i = \langle f, \pi_i \rangle_{d\alpha}$ is not always possible; however, the following result shows that interpolation on the zeros of π_m is equivalent to approximating the integrals involved in computing the f_i by Gauss quadrature.

Theorem 4.2. *Let $l^f_{\mathrm{Zer}(\pi_m)}$ be the Lagrange interpolating polynomial of f on $\mathrm{Zer}(\pi_m)$. We may equivalently write*

$$l^f_X(x) = \sum_{i=0}^{m-1} c_i \pi_i(x), \quad c_i \equiv \sum_{j=1}^{m} f(x_j) \pi_i(x_j) w_j,$$

where the x_j and w_j are the nodes and weights associated with the mth Gauss quadrature rule of $d\alpha$.

Proof. Note that we may write both l^f_X and π_i in the form of (4.5):

$$l^f_X(x) = \sum_{j=1}^{m} l_j(x) f(x_j), \quad \pi_i(x) = \sum_{j=1}^{m} l_j(x) \pi_i(x_j).$$

Then using Theorem 4.1, we can compute c_i directly:

$$c_i = \langle l_X^f, \pi_i \rangle_{d\alpha} = \int_a^b l_X^f(x)\pi_i(x)d\alpha(x) \tag{4.6}$$

$$= \int_a^b \left(\sum_{j=1}^m l_j(x) f(x_j) \right) \left(\sum_{k=1}^m l_k(x)\pi_i(x_k) \right) d\alpha(x) \tag{4.7}$$

$$= \sum_{j=1}^m f(x_j)\pi_i(x_j)w_j. \tag{4.8}$$

□

This shows that interpolation on the zeros of π_m is equivalent to approximation of the projection coefficients of the optimal approximant with Gauss quadrature. An immediate consequence of this result is that we may decompose the squared error $\left\| f - l_X^f \right\|_{d\alpha}^2$ into two components:

$$\left\| f - l_X^f \right\|_{d\alpha}^2 = \sum_{i=0}^{m-1} (f_i - c_i)^2 + \sum_{i=m}^{\infty} f_i^2,$$

or the error in $l_X^f \approx f$ decomposes into the error in approximation of coefficients (the first term above, depending on how well $c_i \approx f_i$) and an intrinsic projection term (the second term above, representing the size of f contained in the orthogonal complement of Π_{m-1}). In our notation, we may observe

$$\sum_{i=0}^{m-1} (f_i - c_i)^2 = \sum_{i=1}^{m} (R_{GQ}^m[f\pi_i, d\alpha])^2. \tag{4.9}$$

Since $c_i \neq f_i$ in general, l_X^f is not the optimal approximation to f in the $\|\cdot\|_{d\alpha}$ norm; however, the above result shows that the error in the approximation—in particular its deviation from the optimal approximation—is closely tied to the error in Gauss quadrature.

Although the characterization above is precise, it is not very concrete, as we know little about the exact behavior of the coefficient error for generic measures. For certain $d\alpha$, the first term error can be explicitly related to the projection error, and they will typically be of the same order, so that interpolation is *nearly* optimal.[2] But it is difficult to use the above result to analyze generic $d\alpha$. Nevertheless, we may observe that due to the exactness of Gauss quadrature, if f is a polynomial of degree

[2]For example, this argument explains the success of the Chebyshev interpolants in [Tre19]. See also, for example, [Xia12].

$m-1$, then $c_i = f_i$ for each $i = 0, 1, \cdots, m-1$, and there is no projection error. We can use this exactness observation together with our earlier estimates on uniform approximation to obtain a more concrete convergence result:

Proposition 4.3. *Let $X = \mathrm{Zer}(\pi_m)$. Suppose f is a polynomial of degree $m-1$. Then*

$$f(x) = l_X^f(x)$$

for all x. Otherwise, we may bound

$$\left\| f - l_X^f \right\|_{d\alpha} \leq 2\sqrt{\mu_0} E_{m-1}^\star(f, S) \tag{4.10}$$

where S is a set containing $\mathrm{Supp}(d\alpha) \cup \mathrm{Zer}(\pi_m)$ and $\mu_0 = \int_a^b d\alpha(x)$.

Proof. Exactness follows from the observations in the preceding paragraph, or from the uniqueness of Lagrange interpolation. We prove (4.10). For an arbitrary $p \in \Pi_{m-1}$,

$$\left\| f - l_X^f \right\|_{d\alpha} = \left\| f - p + p - l_X^f \right\|_{d\alpha} \tag{4.11}$$

$$\leq \|f - p\|_{d\alpha} + \left\| p - l_X^f \right\|_{d\alpha}. \tag{4.12}$$

Of course,

$$\|f - p\|_{d\alpha}^2 = \int_a^b (f(x) - p(x))^2 d\alpha(x) \leq \mu_0 \max_{x \in \mathrm{Supp}(d\alpha)} |f(x) - p(x)|^2.$$

Moreover, since $p - l_X^f$ is a polynomial of degree $m-1$, and its square is a polynomial of degree $2m-2 \leq 2m-1$, we have by exactness of Gauss quadrature and the interpolatory nature of l_X^f on these nodes that

$$\left\| p - l_X^f \right\|_{d\alpha}^2 = \int_a^b (p(x) - l_X^f(x))^2 d\alpha(x) = \sum_{i=1}^m (p(x_i) - f(x_i))^2 w_i,$$

for which we may apply similar bound

$$\leq \mu_0 \max_{x \in \mathrm{Zer}(\pi_m)} |f(x) - p(x)|^2.$$

Notably, we used the property $\sum_i w_i = \mu_0$. Thus, taking square roots and minimizing over $p \in \Pi_{m-1}$, we obtain

$$\left\| f - l_X^f \right\|_{d\alpha} \leq \sqrt{\mu_0} \min_{p \in \Pi_{m-1}} \left(\max_{x \in \mathrm{Supp}(d\alpha)} |f(x) - p(x)| + \max_{x \in \mathrm{Zer}(\pi_m)} |f(x) - p(x)| \right) \tag{4.13}$$

$$\leq 2\sqrt{\mu_0} \min_{p \in \Pi_{m-1}} \max_{x \in S} |f(x) - p(x)| \tag{4.14}$$

$$= 2\sqrt{\mu_0} E_{m-1}^\star(f, S). \tag{4.15}$$

□

We remark that in most scalar applications, the $\|\cdot\|_{d\alpha}$ is not of central interest—this is one of the reasons approximation theory has so emphasized the max-norm we described in Section 2.3. It turns out, however, that this is a very natural norm to consider when we turn to matrix functions, as we see in the next section.

To conclude, we note that the preceding characterization results extend naturally to bivariate interpolation by considering product grids $X \times Y = \{(x_i, y_j) : x_i \in X, y_j \in Y\}$. Here, we will assume that $X = \{x_i\}_{i=1}^m = \mathrm{Zer}(\pi_m)$ and $Y = \{y_i\}_{i=1}^m = \mathrm{Zer}(\tilde{\pi}_m)$ are zero sets of two (potentially identical) families of orthogonal polynomials, with corresponding Gauss weights $\{w_i\}$ and $\{\tilde{w}_i\}$, respectively. For a smooth function $f(x, y)$, we may expand

$$f(x,y) = \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} f_{ij}\pi_i(x)\tilde{\pi}_j(y), \quad f_{ij} \equiv \int\int f(x,y)\pi_i(x)\pi_j(y)d\alpha(x)d\tilde{\alpha}(y).$$

The Lagrange interpolant on this grid is

$$l^f_{X\times Y}(x,y) = \sum_{i=1}^{m}\sum_{j=1}^{m} l_i(x)\,\tilde{l}_j(y)\,f(x_i,y_j), \tag{4.16}$$

where $l_i(x)$ and $\tilde{l}_i(y)$ are fundamental Lagrange interpolating nodes for X and Y, respectively. Moreover, the bivariate interpolant admits the expansion

$$l^f_{X\times Y}(x,y) = \sum_{p=0}^{m-1}\sum_{q=0}^{m-1} c_{pq}\,\pi_p(x)\tilde{\pi}_q(y), \quad c_{pq} = \sum_{i=1}^{m}\sum_{j=1}^{m} f(x_i,y_j)\,\pi_p(x_i)\tilde{\pi}_q(y_j)\,w_i\tilde{w}_j,$$

where $\{\pi_p\}$ and $\{\tilde{\pi}_q\}$ are the respective orthonormal polynomial families for measures $d\alpha$ and $d\tilde{\alpha}$, and c_{pq} is obtained by applying iterated Gauss quadrature to the integrals

$$f_{pq} = \int\int f(x,y)\pi_p(x)\pi_q(y)d\alpha(x)d\tilde{\alpha}(y).$$

This formula shows that the bivariate interpolation coefficients are obtained from iterated Gauss quadrature (or Gauss cubature) of the projection coefficients $\langle f, \pi_p\pi_q\rangle_{d\alpha\times d\tilde{\alpha}}$. As in the univariate case, the error decomposes into a projection term and a term involving the error in the coefficients, and both vanish when f is a polynomial of degree at most $m-1$ in each variable. We do not prove an analog to Proposition 4.3 since uniform approximation by bivariate polynomials is beyond the scope of the present dissertation, and it seems that the proposition may not admit a direct extension to the bivariate case; however, we do note that in classical settings it is well-known that the bivariate problem inherits similar convergence properties to the univariate case [Tre17].

4.2 The action of a matrix function

Now we turn to matrix functions. In this section, we review the problem of computing $f(B)\mathbf{b}$ via the approximation

$$f(B)\mathbf{b} \approx \|\mathbf{b}\| \, U_m f(T_m)\mathbf{e}_1, \tag{4.17}$$

where U_m, T_m are output by the Lanczos procedure, as we previously introduced in Algorithm 5. This approach has been studied extensively.[3] However, we wish to emphasize the characterization of this approximation via polynomial interpolation, which we will extend in the following sections to the more recent algorithms developed for example in [BKS18, Kre19, KKRS21].

For motivation, we observe first that

$$\|f(B)\mathbf{b} - p(B)\mathbf{b}\|_2^2 = \mathbf{b}^T(f(B) - p(B))^2\mathbf{b} = \int_a^b (f(x) - p(x))^2 d\alpha(x) = \|f - p\|_{d\alpha}^2 .$$

That is, if we consider approximating $f(B)\mathbf{b} \approx p(B)\mathbf{b}$, then the standard 2-norm error in our vector approximation is exactly equivalent to the $\|\cdot\|_{d\alpha}$ studied in the previous section, where $d\alpha$ is once again the measure described in (3.15). Beyond this, the following characterization of (4.17) (restated in our notation here) has been known for a long time (for example see [Saa92]):

Theorem 4.4. *Let U_m, T_m be generated by m iterates of the Lanczos procedure applied to the matrix B and starting vector $\mathbf{b}$. Then*

$$\|\mathbf{b}\|_2 \, U_m f(T_m)\mathbf{e}_1 = l^f_{\mathrm{Zer}(\pi_m)}(B)\mathbf{b}, \tag{4.18}$$

where π_m is the degree m orthogonal polynomial obtained from $d\alpha$.

That is, the Lanczos approximation (4.17) is exactly forming the polynomial interpolant described in the previous section. There are two important corollaries to this observation. First, the procedure is *exact* when f is a polynomial of degree $m-1$.

Corollary 4.4.1. *If f is a polynomial of degree $(m-1)$, then (4.17) is exact:*

$$f(B)\mathbf{b} = \|\mathbf{b}\| \, U_m f(T_m)\mathbf{e}_1.$$

Proof. This follows immediately from Theorem 4.4. □

[3] See the standard references [Hig08, GVL13], or the recent PhD thesis [Che22] which emphasizes symmetric matrices.

Remark 4.1. Alternatively, exactness can be established from an inductive argument on the Krylov subspace $\mathcal{K}_m(B, \mathbf{b})$. Once exactness is established, Theorem 4.4 follows from the interpolatory definition of matrix functions (2.4) wherein $f(T_m) = l^f_{\lambda(T_m)}(T_m)$.

The second important corollary to Theorem 4.4 is that we immediately obtain an error bound:

Corollary 4.4.2. *We may bound the error of (4.17) via*

$$\|f(B)\mathbf{b} - \|\mathbf{b}\|_2 U_m f(T_m)\mathbf{e}_1\|_2 \leq 2\,\|\mathbf{b}\|_2\, E^\star_{m-1}(f, S), \tag{4.19}$$

where S is any set containing $\lambda(B) \cup \lambda(T_m)$.

Proof. The proof is almost identical to that of Proposition 4.3 due to the equivalence of the tridiagonal matrix obtained from the Lanczos procedure and the Jacobi matrix obtained from $d\alpha$ as defined in (3.15). That is, we observe

$$\|f(B)\mathbf{b} - \|\mathbf{b}\|_2 U_m f(T_m)\mathbf{e}_1\|_2 = \left\| f - l^f_{\mathrm{Zer}(\pi_m)} \right\|_{d\alpha}$$

and then follow the proof of Proposition 4.3. □

Remark 4.2. This error bound is often quoted in the literature with the choice $S = [a, b]$ for some a, b lower and upper bounds, respectively, of the eigenvalues of B. We note here, however, that making explicit the discrete sets involved may allow for sharper error estimation. We also note that the decomposition of the error into projection and aliasing terms as in the previous section may be of interest for further study.

4.3 Rank-one directional derivatives

In this section, we consider extending the discussion from the previous section to the computation of the Frechet derivative in a rank-one direction, i.e. the computation of

$$L_f(B, \mathbf{b}\mathbf{b}^T) = \frac{\partial}{\partial t}[f(B + t\mathbf{b}\mathbf{b}^T)]_{t=0}. \tag{4.20}$$

Notably, as described in section 2.1, we may also view the Frechet derivative as a bivariate matrix function, i.e.

$$L_f(B, \mathbf{b}\mathbf{b}^T) = \Delta f(B, B^T)\{\mathbf{b}\mathbf{b}^T\}, \tag{4.21}$$

where $\Delta f(x, y)$ is the first-order divided difference of f at x and y. In [Kre19] and [KKRS21], the following approximation is recommended:

$$L_f(B, \mathbf{b}\mathbf{b}^T) \approx \|\mathbf{b}\|^2 U_m L_f(T_m, \mathbf{e}_1\mathbf{e}_1^T)U_m^T. \tag{4.22}$$

The resulting algorithm is stated in Algorithm 9. The analysis in [KKRS21] is driven by the consideration that the Frechet derivative for analytic f can be represented by a Cauchy-integral formula. In [Kre19], the approximation is given as an example of a more general framework for Krylov methods for bivariate problems with a tensorized Krylov projection framework. Although in both cases it is noted that the algorithms simplify for Hermitian matrices, there are not yet precise characterizations of the approximation (4.22) in terms of interpolating polynomials. We also remark that the existing error bounds depend on the field of values of the input matrix B, which in the Hermitian case is an interval containing the eigenvalues of B, and thus there is not yet a characterization of the spectral adaptivity of the procedure, even though spectral adaptivity has been observed numerically.

Our goal in this section is to tie the approximation (4.22) precisely to a bivariate polynomial approximation problem. Before stating our main result, we observe that (4.22) has a polynomial exactness property.

Lemma 4.5. *If f is a polynomial of degree m, then (4.22) is exact:*

$$L_f(B, \mathbf{b}\mathbf{b}^T) = \|\mathbf{b}\|^2 U_m L_f(T_m, \mathbf{e}_1\mathbf{e}_1^T) U_m^T.$$

Alternatively, if g is a bivariate polynomial of degree at most $m-1$, then

$$g\{B, B\}(\mathbf{b}\mathbf{b}^T) = \|\mathbf{b}\|^2 U_m g\{T_m, T_m\}(\mathbf{e}_1\mathbf{e}_1^T) U_m^T,$$

which holds in particular for $g(x, y) = \Delta f(x, y)$.

Proof. This can be viewed as a corollary of Corollary 4.4.1. Recall that according to (2.9),

$$\begin{aligned}
L_f(B, \mathbf{b}\mathbf{b}^T) &= \sum_{i=1}^{m} f_i \sum_{j=0}^{i-1} B^j \mathbf{b}\mathbf{b}^T B^{i-1-j} \\
&= \sum_{i=1}^{m} f_i \sum_{j=0}^{i-1} (\|\mathbf{b}\| U_m T_m^j \mathbf{e}_1)(\mathbf{e}_1^T T_m^{i-1-j} U_m^T \|\mathbf{b}\|) \\
&= \|\mathbf{b}\|^2 U_m \left(\sum_{i=1}^{m} f_i \sum_{j=0}^{i-1} T_m^j \mathbf{e}_1 \mathbf{e}_1^T T_m^{i-1-j} \right) U_m^T \\
&= \|\mathbf{b}\|^2 U_m L_f(T_m, \mathbf{e}_1\mathbf{e}_1^T) U_m^T.
\end{aligned}$$

Note that we used the exactness property Corollary 4.4.1 with $B^j\mathbf{b} = \|\mathbf{b}\| U_m T_m^j \mathbf{e}_1$ when $j < m$. For exactness for bivariate polynomials, we use the same proof starting with (2.12). □

That is, approximation (4.22) inherits polynomial exactness from (4.17). Our main characterization result now follows:

Theorem 4.6. *Let U_m, T_m be computed by the Lanczos procedure applied to B with starting vector $\mathbf{b}$. Moreover, let $l^{\Delta f}_{\mathrm{Zer}(\pi_m)\times\mathrm{Zer}(\pi_m)}(x,y)$ be the bivariate Lagrange interpolant to the divided difference function $\Delta f(x,y)$ on the product grid $\mathrm{Zer}(\pi_m)\times\mathrm{Zer}(\pi_m)$. That is, according to (4.16),*

$$l^{\Delta f}_{\mathrm{Zer}(\pi_m)\times\mathrm{Zer}(\pi_m)}(x,y) = \sum_{i=1}^{m}\sum_{j=1}^{m} l_i(x) l_j(y) \Delta f(x_i, x_j).$$

Then

$$\|\mathbf{b}\|^2 U_m L_f(T_m, \mathbf{e}_1\mathbf{e}_1^T) U_m^T = l^{\Delta f}_{\mathrm{Zer}(\pi_m)\times\mathrm{Zer}(\pi_m)}\{B,B\}(\mathbf{b}\mathbf{b}^T).$$

Proof. We can prove this directly:

$$\begin{aligned}
\|\mathbf{b}\|^2 U_m L_f(T_m, \mathbf{e}_1\mathbf{e}_1^T) U_m^T &= \|\mathbf{b}\|^2 U_m \Delta f\{T_m, T_m\}(\mathbf{e}_1\mathbf{e}_1^T) U_m^T \\
&= \|\mathbf{b}\|^2 U_m l^{\Delta f}_{\mathrm{Zer}(\pi_m)\times\mathrm{Zer}(\pi_m)}\{T_m, T_m\}(\mathbf{e}_1\mathbf{e}_1^T) U_m^T \\
&= l^{\Delta f}_{\mathrm{Zer}(\pi_m)\times\mathrm{Zer}(\pi_m)}\{B,B\}(\mathbf{b}\mathbf{b}^T).
\end{aligned}$$

The first line used Proposition 2.1, the second line uses (2.13), and the third line applies the exactness of Lemma 4.5. □

That is, we may view the approximation of the Frechet derivative in a rank-one direction via the Lanczos procedure as *precisely* forming a bivariate Lagrange interpolating polynomial to the divided difference function $\Delta f(x,y)$. Notably, the Frobenius norm of the Frechet derivative has the following simple form:

$$\left\|L_f(B, \mathbf{b}\mathbf{b}^T)\right\|_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{n} (V^T\mathbf{b})_i^2 (V^T\mathbf{b})_j^2 (\Delta f(\lambda_i, \lambda_j))^2, \tag{4.23}$$

which may be viewed as the iterated Riemann-Stieltjes integral

$$= \int_a^b \int_a^b \Delta f(x,y) d\alpha(x) d\alpha(y)$$

with $d\alpha$ defined as in (3.15). Consequently, we observe that the error in the approximation (in the matrix Frobenius norm) is given precisely by

$$\begin{aligned}
&\left\|L_f(B, \mathbf{b}\mathbf{b}^T) - l^{\Delta f}_{\mathrm{Zer}(\pi_m)\times\mathrm{Zer}(\pi_m)}\{B,B\}(\mathbf{b}\mathbf{b}^T)\right\|_F^2 \\
= \int_a^b \int_a^b &(\Delta f(x,y) - l^{\Delta f}_{\mathrm{Zer}(\pi_m)\times\mathrm{Zer}(\pi_m)}(x,y))^2 d\alpha(x) d\alpha(y).
\end{aligned}$$

Ideally, this error term would be analyzed directly in order to capture the spectral adaptivity of the procedure, although we are not aware of readily available non-asymptotic tools from approximation theory for fairly generic $d\alpha$.

We provide instead the following concrete error estimate, which appears also in [Kre19].

Proposition 4.7. *Let f be continuously differentiable on an interval $[a, b]$ containing the eigenvalues of B. Define the remainder term*

$$r_m \equiv L_f(B, \mathbf{b}\mathbf{b}^T) - \|\mathbf{b}\|^2 U_m L_f(T_m, \mathbf{e}_1\mathbf{e}_1^T) U_m^T.$$

Then

$$\|r_m\|_F \leq 2\|\mathbf{b}\|^2 E^\star_{m-1}(f', [a, b]).$$

Proof. Let p be a polynomial of degree m. Note that since f is continuously differentiable and $\Delta f(x, y) - \Delta p(x, y) = \Delta(f - p)(x, y)$, we can apply the mean value theorem of divided differences to obtain

$$\Delta(f - p)(x, y) = (f' - p')(\xi) \tag{4.24}$$

for some $\xi \in [x, y] \subset [a, b]$. Notably, if p is a polynomial of degree m, then $\Delta p(x, y)$ is a bivariate polynomial of degree $m - 1$ in x and y. The exactness of Theorem 4.6 implies for p any polynomial of degree m,

$$L_p(B, \mathbf{b}\mathbf{b}^T) - \|\mathbf{b}\|^2 U_m L_p(T_m, \mathbf{e}_1\mathbf{e}_1^T) U_m^T = 0.$$

Substituting this into our expression for r_m and applying the triangle inequality bounds $\|r_m\|_F$ by two terms:

$$\begin{aligned}\|r_m\|_F \leq &\left\|L_f(B, \mathbf{b}\mathbf{b}^T) - L_p(B, \mathbf{b}\mathbf{b}^T)\right\|_F \\ &+ \|\mathbf{b}\|^2 \left\|L_p(T_m, \mathbf{e}_1\mathbf{e}_1^T) - L_f(T_m, \mathbf{e}_1\mathbf{e}_1^T)\right\|_F.\end{aligned}$$

An immediate consequence of Equation (4.23) is that

$$\left\|L_f(B, \mathbf{b}\mathbf{b}^T) - L_p(B, \mathbf{b}\mathbf{b}^T)\right\|_F \leq \|\mathbf{b}\|^2 \max_{x,y\in[a,b]} |\Delta(f - p)(x, y)|,$$

and similarly

$$\|\mathbf{b}\|^2 \left\|L_p(T_m, \mathbf{e}_1\mathbf{e}_1^T) - L_f(T_m, \mathbf{e}_1\mathbf{e}_1^T)\right\|_F \leq \|\mathbf{b}\|^2 \max_{x,y\in[a,b]} |\Delta(f - p)(x, y)|.$$

Thus, applying (4.24) and minimizing over polynomials of degree m (whose derivatives are polynomials of degree $m - 1$), we obtain the final result

$$\|r_m\|_F \leq 2\|\mathbf{b}\|^2 E_{m-1}(f', [a, b]).$$

□

This provides a means of estimating the error in (4.22) using Theorem 2.3, but note that it uses the whole spectral interval and does not apply to discrete sets as many of the results in this dissertation have. This means that we will not be able to account for the "spectral adaptivity" that we observe in practice. Still, we note that this bound seems to be effective up to a small constant factor when the eigenvalues of B are densely distributed in $[a, b]$. It would be of interest to obtain a result that accounts for the spectral adaptivity of the procedure.

We provide in Algorithm 9 the resulting algorithm, whose most expensive components are the m-steps of the Lanczos procedure together with the evaluation of a Frechet derivative on a small m-by-m problem. It is important to note that the output approximation is in low-rank form:

$$L_f(B, \mathbf{b}\mathbf{b}^T) \approx U_m X_m U_m^T$$

for $U_m \in \mathbb{R}^{n\times m}$, $X_m \in \mathbb{R}^{m\times m}$. That is, even though $L_f(B, \mathbf{b}\mathbf{b}^T) \in \mathbb{R}^{n\times n}$, which involves n^2 entries, we may represent our approximation with $O(mn) + O(m^2)$ storage. Moreover, if only an individual entry is required, this can be computed in $O(m^2)$ time since $(U_m X U_m^T)_{ij} = \mathbf{u}_i^T X \mathbf{u}_j$ where we have use $\mathbf{u}_k$ to represent the transpose of the kth row of U_m (not the kth column), which is a vector in $\mathbb{R}^m$. If only a small subset of the original matrix is of interest, then it makes sense to output this low-rank factorization and observe only the entries of interest, such as the diagonal or a small submatrix. If $K = O(n)$ entries are queried, this would only require $O(nm^2)$ time.

Algorithm 9 LanczosFrech$(f, B, \mathbf{b}, m)$
Lanczos approximation of Frechet derivative.

1: **Input:** Scalar function f, Symmetric matrix $B \in \mathbb{R}^{n\times n}$, initial vector $\mathbf{b} \in \mathbb{R}^n$, iteration count m.
2: **Output:** U_m, F such that $(U_m F U_m^T)_{ij} \approx (L_f(B, \mathbf{b}\mathbf{b}^T))_{ij}$
3: Compute $(U_m, T_m) \leftarrow$ Lanczos$(B, \mathbf{b}, m)$.
4: Compute $F \leftarrow \|\mathbf{b}\|^2 L_f(T_m, \mathbf{e}_1\mathbf{e}_1^T)$
5: **Complexity summary:** $O(m^2 n) + O(m \cdot \operatorname{nnz} B) + O(m^3)$

4.4 Rank-one updates

Let us now consider computing a rank-one update. In [BKS18], the approximation

$$f(B + \mathbf{b}\mathbf{b}^T) - f(B) \approx U_m(f(T_m + \|\mathbf{b}\|^2 \mathbf{e}_1\mathbf{e}_1^T) - f(T_m))U_m^T \tag{4.25}$$

is introduced, which results in Algorithm 10. The motivation in [BKS18] is in terms of tensor-product Krylov spaces. We provide here a different perspective that unites this approximation with the approximation of the Frechet derivative by emphasizing bivariate Lagrange interpolation. Moreover, we provide an error bound that may account for spectral adaptivity due to outlying eigenvalues.

Before stating our main result, we remark on a connection due to the fundamental theorem of calculus for Frechet spaces between the approximation of this section (4.25) and the approximation of the last (4.22) for computing the Frechet derivative. That is, using notation from Section 3.4 for $B(t) = B + t\mathbf{b}\mathbf{b}^T$ and $T_m(t) = T_m + t\left\|\mathbf{b}\right\|^2 \mathbf{e}_1\mathbf{e}_1^T$, the fundamental theorem of calculus for Frechet differentiable spaces yields

$$f(B + \mathbf{b}\mathbf{b}^T) - f(B) = \int_0^1 L_f(B(t), \mathbf{b}\mathbf{b}^T)dt, \tag{4.26}$$

and similarly

$$f(T_m + \left\|\mathbf{b}\right\|^2 \mathbf{e}_1\mathbf{e}_1^T) - f(T_m) = \left\|\mathbf{b}\right\|^2 \int_0^1 L_f(T_m(t), \mathbf{e}_1\mathbf{e}_1^T)dt. \tag{4.27}$$

This means that if f is differentiable on $\lambda(B(t))$ and $\lambda(T_m(t))$ for $t \in [0,1]$, then any error bound for (4.22) yields an error bound for (4.25) via

$$\begin{aligned}\left\| \int_0^1 L_f(B(t), \mathbf{b}\mathbf{b}^T) - \left\|\mathbf{b}\right\|^2 U_m L_f(T_m(t), \mathbf{e}_1\mathbf{e}_1^T)U_m^T dt \right\|_F \\ \leq \max_{t\in[0,1]} \left\| L_f(B(t), \mathbf{b}\mathbf{b}^T) - \left\|\mathbf{b}\right\|^2 U_m L_f(T_m(t), \mathbf{e}_1\mathbf{e}_1^T)U_m^T \right\|_F .\end{aligned} \tag{4.28}$$

In particular, we may conclude that (4.25) is *exact* whenever f is a polynomial of degree m.

Lemma 4.8. *Let f be a polynomial of degree m. Then equality holds in (4.25), or*

$$f(B + \mathbf{b}\mathbf{b}^T) - f(B) = U_m(f(T_m + \|\mathbf{b}\|^2\mathbf{e}_1\mathbf{e}_1^T) - f(T_m))U_m^T. \tag{4.29}$$

Moreover, due to Proposition 2.2, we may state this equivalently with

$$\Delta f\{B, B + \mathbf{b}\mathbf{b}^T\}(\mathbf{b}\mathbf{b}^T) = U_m \Delta f\{T_m, T_m + \left\|\mathbf{b}\right\|^2 \mathbf{e}_1\mathbf{e}_1^T\}(\mathbf{e}_1\mathbf{e}_1^T)U_m^T. \tag{4.30}$$

Proof. This follows immediately from (4.28) together with the exactness of the approximation of the Frechet derivative Lemma 4.5. It is also possible to provide an inductive proof [BKS18] based on the projection properties of Krylov subspaces. Finally, we note that Proposition 2.2 also allows one to use results from [Kre19] for Krylov subspace methods for bivariate matrix functions. □

Our main characterization result is the following.

Theorem 4.9. *Let U_m, T_m be computed by the Lanczos procedure applied to B with starting vector $\mathbf{b}$. Moreover, let $l^{\Delta f}_{X\times Y}$ be the bivariate Lagrange interpolating polynomial to the divided difference function f on the product grid $X \times Y$ where $X = \lambda(T_m)$ and $Y = \lambda(T_m + \|\mathbf{b}\|^2 \mathbf{e}_1\mathbf{e}_1^T)$. Then*

$$U_m \left(f(T_m + \|\mathbf{b}\|^2 \mathbf{e}_1\mathbf{e}_1^T) - f(T_m)\right) U_m^T = l^{\Delta f}_{X\times Y}\{B, B + \mathbf{b}\mathbf{b}^T\}(\mathbf{b}\mathbf{b}^T).$$

Proof. We may prove this directly by using Lemma 4.8. That is,

$$\begin{aligned} U_m \left(f(T_m + \|\mathbf{b}\|^2 \mathbf{e}_1\mathbf{e}_1^T) - f(T_m)\right) U_m^T &= U_m \Delta f\{T_m, T_m + \|\mathbf{b}\|^2 \mathbf{e}_1\mathbf{e}_1^T\}(\mathbf{e}_1\mathbf{e}_1^T) U_m^T \\ &= U_m l^{\Delta f}_{X\times Y}\{T_m, T_m + \|\mathbf{b}\|^2 \mathbf{e}_1\mathbf{e}_1^T\}(\mathbf{e}_1\mathbf{e}_1^T) U_m^T \\ &= l^{\Delta f}_{X\times Y}\{B, B + \mathbf{b}\mathbf{b}^T\}(\mathbf{b}\mathbf{b}^T). \end{aligned}$$

On the first line, we used the equivalence from Proposition 2.2; the second line uses Equation (2.13); the final line applies Lemma 4.8. □

Note that this means that characterization of the error in the Frobenius norm is possible through an iterated integral:

$$\begin{aligned} &\left\| \Delta f\{B, B + \mathbf{b}\mathbf{b}^T\}(\mathbf{b}\mathbf{b}^T) - l^{\Delta f}_{X\times Y}\{B, B + \mathbf{b}\mathbf{b}^T\}(\mathbf{b}\mathbf{b}^T) \right\|_F^2 \\ &\qquad = \int\int (\Delta f(x, y) - l^{\Delta f}_{X\times Y}(x, y))^2 d\alpha(x) d\tilde{\alpha}(y). \end{aligned}$$

At the same time, the exactness of the procedure immediately allows us to bound its 2-norm error using estimates from Section 2.3.

Proposition 4.10. *Let r_m^f denote the remainder of (4.25), i.e.*

$$r_m^f \equiv f(B + \mathbf{b}\mathbf{b}^T) - f(B) - U_m \left(f(T_m + \|\mathbf{b}\|^2 \mathbf{e}_1\mathbf{e}_1^T) - f(T_m)\right) U_m^T. \tag{4.31}$$

Then the two-norm satsifies the following bound:

$$\left\| r_m^f \right\|_2 \leq 4E_m(f, S) \tag{4.32}$$

where $S = \lambda(B) \cup \lambda(B + \mathbf{b}\mathbf{b}^T) \cup \lambda(T_m) \cup \lambda(T_m + \|\mathbf{b}\|^2 \mathbf{e}_1\mathbf{e}_1^T)$.

Proof. We observe that for any polynomial of degree m, by Lemma 4.8, we may subtract $p(B)$ and correspondingly add $U_m p(T_m) U_m^T$ without affecting the remainder r_m^f. We can do this also with $p(B + \mathbf{b}\mathbf{b}^T)$ and $U_m p(T_m + \|\mathbf{b}\|^2 \mathbf{e}_1\mathbf{e}_1^T) U_m^T$. Thus, we have

$$r_m^f = r_m^{f-p}$$

for any p a polynomial of degree at most m. Applying the triangle inequality to $\left\|r_m^{f-p}\right\|_2$ and choosing p to be the best approximation to f on the set S yields the desired bound. □

Remark 4.3. Note that this bound is similar to that presented in [BKS18], but it makes explicit a dependence on the discrete spectrum of B.

We should note that the bound (4.32) *does not depend on* $\|\mathbf{b}\|$. This means that we can expect it to be pessimistic as $\|\mathbf{b}\| \to 0$. Indeed, as discussed at the beginning of this section, we know that in this region the error in the Frobenius norm behaves like the error in (4.22), according to (4.28). Nevertheless, $\|\mathbf{b}\|$ may also be much larger than zero, and so it is interesting that the bound holds uniformly. Note also that the output of the algorithm is in low-rank form, so that the comments from the end of the preceding section pertaining to Algorithm 9 apply also here.

Algorithm 10 LanczosUpdate$(f, B, \mathbf{b}, m)$
Lanczos approximation of Frechet derivative.

1: **Input:** Scalar function f, Symmetric matrix $B \in \mathbb{R}^{n\times n}$, initial vector $\mathbf{b} \in \mathbb{R}^n$, iteration count m.
2: **Output:** U_m, F such that $(U_m F U_m^T)_{ij} \approx (f(B + \mathbf{b}\mathbf{b}^T) - f(B))_{ij}$
3: Compute $(U_m, T_m) \leftarrow$ Lanczos$(B, \mathbf{b}, m)$.
4: Compute $F \leftarrow f(T_m + \|\mathbf{b}\|^2 \mathbf{e}_1\mathbf{e}_1^T) - f(T_m)$
5: **Complexity summary:** $O(m^2 n) + O(m \cdot \operatorname{nnz} B) + O(m^3)$

4.5 Example

To conclude this chapter, we recreate the example from the end of Chapter 3. We approximate the same functions f, g, and h and their derivatives f', g', and h'; however, now we consider forming the Chebyshev interpolants of increasing degree (see [Tre19] for further details). That is, instead of Gauss-Legendre quadrature, we approximate $f \approx p_m$ where p_m interpolates f on $m+1$ Chebyshev points, and we measure $\max_{x\in[-1,1]} |f(x) - p_m(x)|$. We compare this error in approximation on $[-1, 1]$ to the error in the approximations (4.17), (4.22), and (4.25), where the first is the error in the vector 2-norm and the latter two errors are measured in terms of the matrix 2-norm, and we do this using the same matrices B_1 and B_2 and $\mathbf{b}$ chosen randomly as described in Chapter 3. The results are depicted in Figure 4.1.

Importantly, we note essentially the same behavior in error as was observed in Section 3.5; however, the convergence rate is now governed by polynomial approximation of degree m rather than $2m$ since we are now dealing with approximation on the

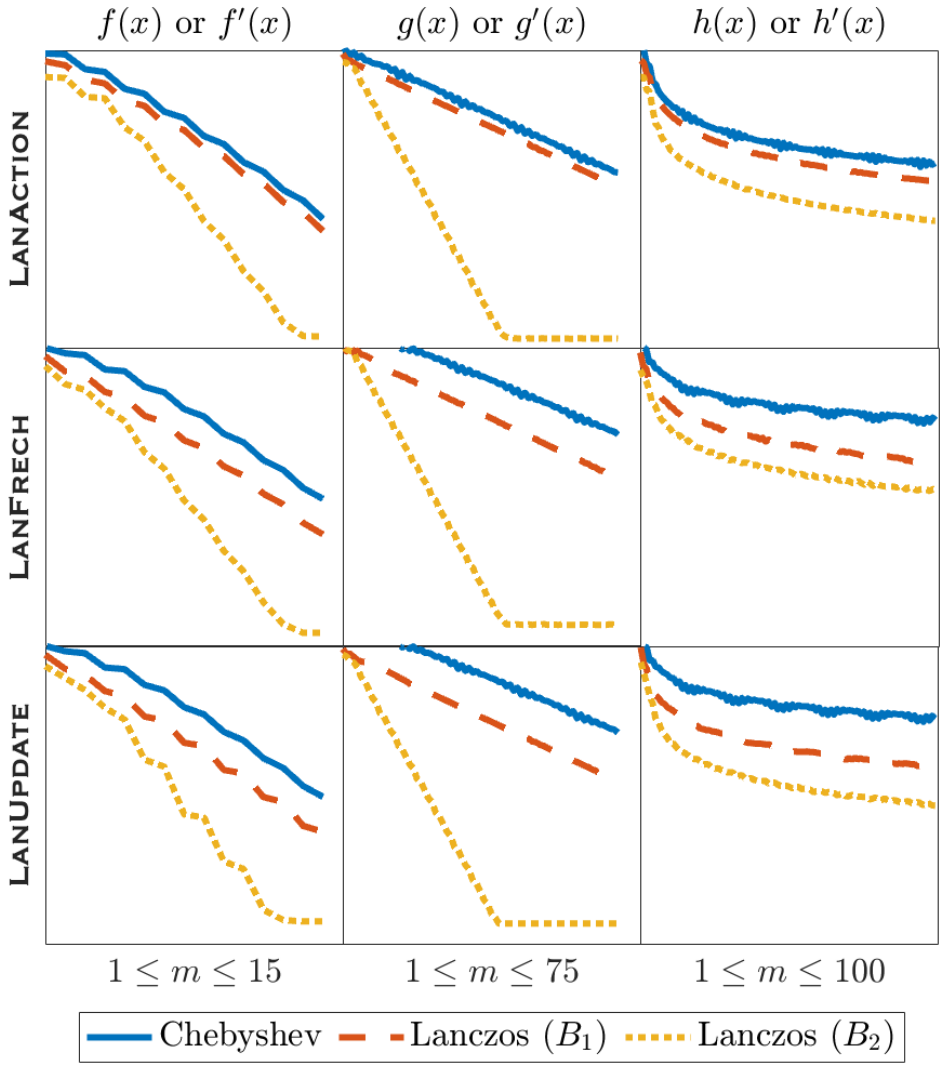

Figure 4.1: Error plots for Lanczos interpolation.

whole interval rather than a numerical integration problem. That is, we see that the three methods for matrix function approximation considered here really do behave like scalar polynomial interpolation, as we would expect from the characterization results of this chapter.

Chapter 5

Analysis and experiments for important functions

Chapter overview

Our goal in this chapter is to apply the analysis presented in the last few chapters to the computation of some matrix functions that are important in practice. That is, we will use the approximation theory discussed in Section 2.3 together with the characterization results and error bounds from Chapter 3 and Chapter 4 in order to provide a concrete analysis of the convergence properties of these Lanczos-based procedures for various important functions.[1]

It should be noted that bounds based on polynomial approximation are common, so similar analysis is available elsewhere; however, we believe that the use of Proposition 2.4 to characterize spectral adaptivity for Lanczos procedures is new and simplifies upon potential theoretic arguments (for example in [DTT98, Kui06]) whilst providing concrete, non-asymptotic bounds. It seems also to be the first attempt to quantify the spectral adaptivity of Lanczos procedures for matrix function updates in a rank-one direction. We hope this chapter will be a helpful reference for how polynomial approximation bounds apply concretely to the problem of matrix function computation.

[1]Notably, all of the bounds considered here are *a priori* bounds (they require knowledge of the spectrum of the matrix that is not available in practice); consequently, the point of these experiments is to illustrate the convergence properties of the algorithms in a controlled setting rather than to provide a stopping criterion for the use of the algorithm in practice. For this latter need, one requires *a posteriori* estimates which estimate the error using the output of the algorithm as it runs. The best available *a posteriori* estimates seem to be simply to let $d \geq 1$ be a small parameter for which we compare the approximation at step m to the approximation at step $m + d$ (see [GM09, BKS18] for more details). These are heuristic, though effective. In future work, it would be interesting to see if better *a posteriori* estimates can be developed from the perspective of polynomial approximation.

5.1 Background

Before beginning, we should discuss existing convergence analysis. The use of polynomial error estimates to quantify the behavior of Lanczos-based (or more generally Arnoldi-based) matrix function approximations is not new. For example, error bounds based on the Taylor expansion of the exponential function go back to [Saa92], which were subsequently tightened in [HL97] using more careful arguments based on Faber polynomials and conformal mapping. Due to the close relationship between the Lanczos procedure and orthogonal polynomials, it is perhaps the most natural framework for error estimation (at least in the symmetric case). As a result, error bounds based on minimax polynomial error on the field of values are common.

Improving upon these estimates is difficult. One approach is to replace intervals by more generic sets and to apply potential theoretic arguments to estimate rates of convergence [DTT98, Kui06]. Importantly, however, the results are necessarily asymptotic and do not yield explicit bounds. Another powerful idea is based on the use of integral formulations of matrix functions. For example, the Cauchy-integral formula in complex analysis has analogues for matrix functions, their Frechet derivatives, and low-rank updates (assuming the function f is analytic on a domain enclosing the eigenvalues of the operator, see e.g. [Hig08, KKRS21]). It is also possible to use other integral frameworks, such as for Markov functions [BKS18]. It turns out that we can then view Krylov subspace methods as an integral of shifted linear system solves, and we may obtain error bounds by "integrating the error" of shifted linear system solves on a well-chosen contour. This is the idea of the techniques used in [HL97] for analyzing the action of the matrix exponential $e^B\mathbf{b}$. Extensions of these ideas are discussed in [BKS18] for low-rank updates and [KKRS21] for Frechet derivatives. A related idea has also been applied to the Lanczos procedure recently in [CGMM22] for generic analytic functions very successfully. One can also consider rational functions [FS09] or Stieltjes functions [FS16]. We do not pursue these kinds of bounds further in this dissertation. One issue with these approaches is that they rely heavily on integral representations and analyticity of the function on the spectrum of the matrix. Although many important functions are complex analytic in the region we care about, this is not always the case, and the arguments can become rather involved.

In this chapter, we show how the bounds from Section 2.3 together with the characterizations provided in Chapter 4 allow for simple a priori error estimates that can in some cases be modified to incorporate spectral adaptivity due to outlying points. Since the $f(B)\mathbf{b}$ problem has received so much attention in the literature,

we will focus primarily on the computation of $L_f(B, \mathbf{b}\mathbf{b}^T)$ and the rank-one update $f(B+\mathbf{b}\mathbf{b}^T)-f(B)$. We do not seek to make the bounds as tight as possible, but rather to show how straightforward bounds based on polynomial approximation can explain convergence behavior. Notably, similar bounds could also be applied to non-analytic functions using Theorem 2.3.

5.2 The matrix exponential

We begin with perhaps the most well-known matrix function: the matrix exponential. We consider here the approximation of the Frechet derivative according to Section 4.3. That is, let $f(x) = e^x$ for and consider the approximation

$$L_f(B, \mathbf{b}\mathbf{b}^T) \approx \|\mathbf{b}\|^2 U_m L_f(T_m, \mathbf{e}_1\mathbf{e}_1^T)U_m^T.$$

Notably, e^x is entire, so we expect that it obtains superlinear convergence since it is analytic in every ellipse $\mathbb{B}_{[a,b]}(\rho)$ for all $\rho > 1$. We specialize Theorem 2.3 to the exponential in the following way: note that $\mathbb{B}_{[a,b]}(\rho)$ has a single point of largest real part at

$$z^\star(\rho) = \frac{b+a}{2} + \frac{(b-a)}{4}(\rho + \rho^{-1}),$$

so that

$$\max_{z \in \mathbb{B}_{[a,b]}(\rho)} e^z = e^{z^\star(\rho)}.$$

Then Theorem 2.3 yields

$$E_m^*(f, [a,b]) \leq \frac{2e^{z^\star(\rho)}}{\rho^m(\rho-1)}.$$

Notably, for any fixed m, we can numerically optimize the right-hand side for the minimizing $\rho > 1$, which yields the desired superlinear convergence. This produces the following error bound:

Proposition 5.1. *Let $f(x) = e^x$ and let*

$$r_m \equiv L_f(B, \mathbf{b}\mathbf{b}^T) - \|\mathbf{b}\|^2 U_m L_f(T_m, \mathbf{e}_1\mathbf{e}_1^T)U_m^T$$

be the remainder after the approximation (4.22). Then

$$\|r_m\|_F \leq \frac{4\|\mathbf{b}\|^2 e^{z^\star(\rho)}}{\rho^{m-1}(\rho-1)}.$$

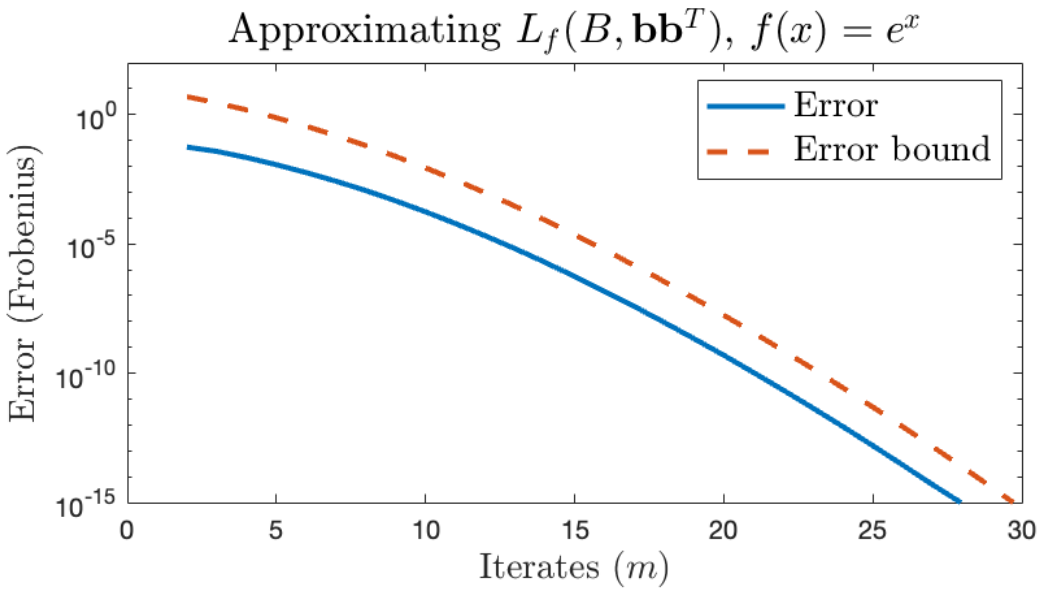

Figure 5.1: Approximating the Frechet derivative without outliers. The error is measured in the Frobenius norm, and the error bound is from Proposition 5.1.

As a first example, we consider a 2000-by-2000 matrix B with eigenvalues linearly spaced in $[-20, 0]$. The result is depicted in Figure 5.1. A similar bound is presented in [KKRS21] based on the Cauchy-integral representation, and it is compared there to bounds for the matrix exponential based on polynomial approximation.

Notably, however, this *does not* capture the spectral adaptivity that the procedure exhibits in practice, as is seen from the following second example. We form four matrices: B_1 has linearly spaced eigenvalues on $[-20, 0]$ as before, B_2 has linearly spaced eigenvalues on $[-20, 0]$ with ten outliers at $\{-100, -92, \cdots, -28\}$, B_3 has linearly spaced eigenvalues on $[-20, 0]$ with one outlier at $\{10\}$, and B_4 has linearly spaced eigenvalues on $[-20, 0]$ with one outlier at $\{-30\}$. For a random vector $\mathbf{b}$ of unit norm, we approximate the Frechet derivative $L_f(B_i, \mathbf{b}\mathbf{b}^T)$ via and record the error in the Frobenius norm. The results are depicted in Figure 5.2. The two distributions with only one outlier exhibit almost identical convergence rates to that with eigenvalues densely distributed in $[-20, 0]$; the fourth, which has 10 outlying eigenvalues, is correspondingly delayed in its convergence by about 10 iterates, but settles in to a convergence similar to that of the matrix with eigenvalues on $[-20, 0]$ and much faster than would be expected for a matrix with eigenvalues in $[-100, 0]$. To our knowledge, there is no characterization of this spectral adaptivity for the Frechet derivative, excepting the observation that it does occur in experiments. We discuss spectral adaptivity more in the next section when we turn to rank-one updates.

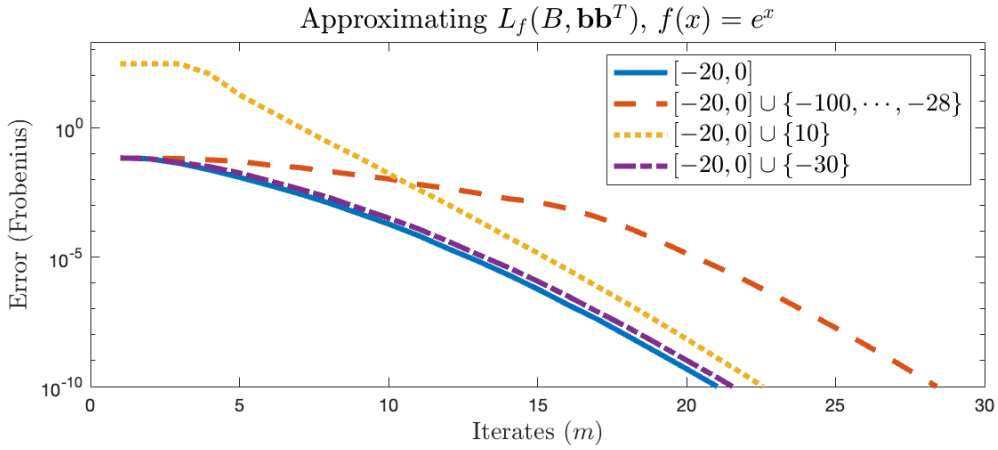

Figure 5.2: Approximating the Frechet derivative with outliers.

5.3 Fractional powers

In this section, we consider updating fractional powers of a matrix with (4.25). That is, we have $f(x) = x^{-\gamma}$ for some $\gamma > 0$. Notably, when $\gamma = 1$, we obtain the ordinary matrix inverse. Famously, a rank one perturbation of an invertible matrix is also a rank one perturbation of the inverse of the matrix, according to the Sherman-Morrison formula [GVL13]. It is not generally true, however, that a rank-one perturbation of a matrix function is rank one.[2] We consider as an example in this section the inverse square root function with $\gamma = -1/2$ and we compare to a bound for Markov functions offered in [BKS18].

Suppose our matrix has eigenvalues in $[a, b]$ with $0 < a < b$. Then the maximum of $x^{-\gamma}$ on $\mathbb{B}_{[a,b]}(\rho)$ occurs at the point of smallest real part:

$$z^\star(\rho) = \frac{a+b}{2} - \frac{(b-a)}{4}(\rho + \rho^{-1}). \tag{5.1}$$

Since the function has a branch point at $x = 0$, this means that $\rho > 1$ also has a corresponding upper bound $\rho < \rho_{max}$ which can be obtained by solving the quadratic equation $z^\star(\rho) = 0$. We may state explicitly

$$\rho_{max} = \frac{(\sqrt{a} + \sqrt{b})^2}{(b-a)}.$$

Remark 5.1. This result relates the convergence rate of (4.25) when $f(x) = x^{-\gamma}$ to the well-known convergence rate for the Conjugate Gradient (CG) algorithm for solving positive definite linear systems [Saa03, GVL13]. If we substitute $\kappa \equiv b/a$, which is

[2] Generically, if r is a rational function (or a polynomial) of degree d, then the perturbation is rank d [BVL00]. But this does not apply to fractional α, for example.

the condition number of a positive definite matrix with maximal eigenvalue b and minimal eigenvalue a, then we obtain the bound

$$\rho_{max} = \frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}.$$

This is due to the shared singularity at 0 for any $\gamma \neq 0$ (where $\gamma = 1$ corresponds to the solution of a linear system). Thus, the limiting value of ρ is the same for all $\gamma \neq 0$. For well-conditioned matrices, the algorithm will converge faster than $O((\rho_{max} - \epsilon)^m)$ for any small enough ϵ. Of course, this bound for CG is elegant and known to predict convergence in some cases, but in many cases where the eigenvalues of the matrix have clustering it can be extremely pessimistic.

Specializing Theorem 2.3, we obtain

$$E_m(f, [a, b]) \leq \frac{2}{\rho^m(\rho - 1)z^\star(\rho)^\gamma}.$$

This leads to the following proposition:

Proposition 5.2. *Let $f(x) = x^{-\gamma}$ and B and $B + \mathbf{b}\mathbf{b}^T$ both have all eigenvalues in an interval $[a, b]$ with $0 < a < b$. Then the matrix 2-norm error in Equation (4.25) is bounded by*

$$\|r_m\|_2 \leq \frac{8}{\rho^m(\rho - 1)z^\star(\rho)^\gamma},$$

where $1 < \rho < \rho_{max}$ and r_m is defined as in (4.31).

Clearly, the bound deteriorates as $\rho \to 1$ or $\rho \to \rho_{max}$, so we may again numerically minimize the right-hand side with respect to ρ for fixed m.

In [BKS18], the following bound (specialized to our notation) was obtained for rank-one updates of Markov functions, of which the inverse square root is an example:

$$\|r_m\|_2 \leq 8|f'(z^\star(\rho))| \, \|\mathbf{b}\|^2 \, \rho^{-m}. \tag{5.2}$$

Once again, $1 < \rho < \rho_{max}$. Note that this includes an explicit dependence on $\mathbf{b}$, but it relies on the derivative of f in the bound. Moreover, it predicts the same rate of convergence as Proposition 5.2. It is notable, however, that a more general form of the bound applies to the more general algorithm for non-symmetric matrices, whereas the assumption that the underlying matrix is Hermitian is essential to our analysis.

We illustrate the significance of Proposition 5.2 with the following example.[3] We form four symmetric matrices B_1, B_2, B_3, and B_4, all with $n = 500$ eigenvalues in

[3] This example is inspired by [BKS18, Example 5.10].

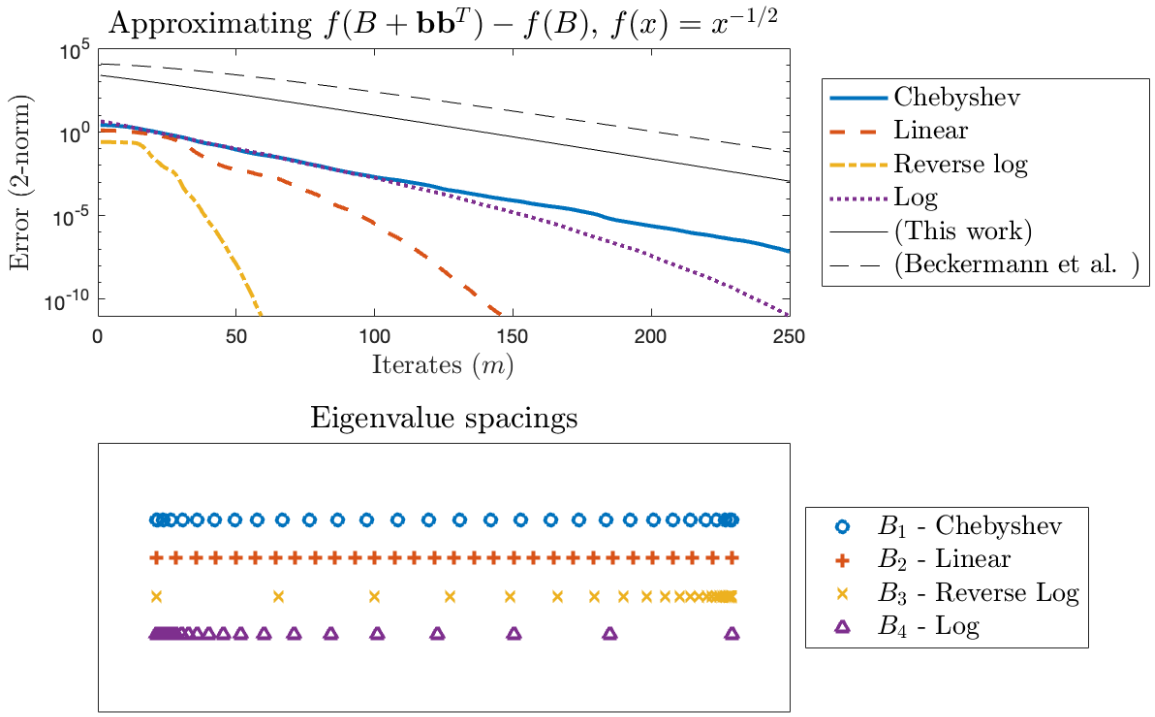

Figure 5.3: Approximating a rank-one update.

$[0.01, 10]$, but with different distributions (we depict the distributions for smaller $n = 30$ in the second panel of Figure 5.3). B_1 has eigenvalues throughout the whole spectrum with clustering near both ends given by the Chebyshev nodes after a linear transplantation to $[0.01, 10]$. B_2 has linearly spaced eigenvalues. B_3 has eigenvalues with a reversed logarithmic spacing so that there is clustering near 10. B_4 has ordinary logarithmic spacing so that there is clustering near 0.01. The vector $\mathbf{b}$ is a random normalized vector so that $B_j + \mathbf{b}\mathbf{b}^T$ has eigenvalues in $[0.01, 10.1]$ with basically the same spacing. The result of approximating the rank-one update for each matrix, along with the bound predicted by Proposition 5.2, is depicted in the first panel of Figure 5.3. We also show the bound Equation (5.2) from [BKS18] for reference.

We may observe that up to a constant the error bound of Proposition 5.2 seems to capture the behavior of the convergence curve well for the matrix with Chebyshev-spaced eigenvalues; nevertheless, each of the other eigenvalue distributions exhibits the spectral adaptivity of the Lanczos procedure that is so powerful and the bounds become rapidly pessimistic.

We explore this more by refining Proposition 5.2 with the extension of standard uniform approximation bounds provided in Proposition 2.4. We will focus on the reverse logarithmic spacing case where clustering occurs at the top of the interval and

there are a small number of small eigenvalues—this is the case in the example above where the spectral adaptivity is the most pronounced. That is, we begin by replacing the interval $[a,b]$ with the semidiscrete set $\{\lambda_1, \lambda_2, \cdots, \lambda_k\} \cup [a,b]$, where we assume $0 < \lambda_1 < \lambda_2 < \cdots < \lambda_k < a < b$. When $m > k$, then we may bound

$$E_m(f, [a,b] \cup \{\lambda_j\}) \leq \left\| \prod_{j=1}^{k} (x - \lambda_j) \right\|_{[a,b]} \cdot E_{m-k}(\Delta_{\{\lambda_j\}} f, [a,b]).$$

Without any further assumptions on the λ_j, we may apply the simple bound

$$\left\| \prod_{j=1}^{k} (x - \lambda_j) \right\|_{[a,b]} \leq C^k$$

where C is the length of an interval that contains $\{\lambda_j\}_{j=1}^k \cup [a,b]$. The second term in the product may be bounded by

$$E_{m-k}(\Delta_{\{\lambda_j\}} f, [a,b]) \leq \frac{2 \left\| \Delta_{\{\lambda_j\}_{j=1}^k} f \right\|_{\mathbb{B}_{[a,b]}(\rho)}}{\rho^{m-k}(\rho - 1)}.$$

We may choose $1 < \rho < \rho_{max}$ large enough so that each of the λ_j are contained within $\mathbb{B}_{[a,b]}(\rho)$ without intersecting with the origin, so that Proposition 2.5 applies and we write

$$\left\| \Delta_{\{\lambda_j\}} f \right\|_{\mathbb{B}_{[a,b]}(\rho)} \leq \frac{\left\| f^{(k)} \right\|_{\mathbb{B}_{[a,b]}(\rho)}}{k!}.$$

Notably, the dependence on the λ_j has entirely vanished from the bound now, so that we may write once $m > k$ that

$$E_m(f, \{\lambda_j\} \cup [a,b]) \leq \frac{2C^k \left\| f^{(k)} \right\|_{\mathbb{B}_{[a,b]}(\rho)}}{\rho^{m-k}(\rho - 1)k!}.$$

This means that the rate of convergence is unaffected by a few small outliers outside of $[a,b]$. Moreover, the constant modifier $\left\| f^{(k)} \right\|_{\mathbb{B}_{[a,b]}(\rho)}$ can be made explicit when $f^{(k)}$ is known—in this case, we obviously have $f^{(k)}(x) = (-1)^k \gamma(\gamma-1)\cdots(\gamma-k+1)x^{-\gamma-k}$. Finally, we combine these observations with Proposition 4.10 in order to obtain the following bound.

Proposition 5.3. *Let $f(x) = x^{-\gamma}$ and $\lambda(B) \subset \{\lambda_j\}_{j=1}^k \cup [a,b]$ with each $\lambda_j \leq a$. Moreover, assume $\lambda_{max}(B + \mathbf{b}\mathbf{b}^T) \leq b$. Then if $m > k$ we have*

$$\|r_m\|_2 \leq \frac{8 \cdot C^{4k} |\gamma(\gamma-1)\cdots(\gamma-4k+1)|}{\rho^{m-4k}(\rho-1)z^\star(\rho)^{\gamma+4k}(4k)!}. \tag{5.3}$$

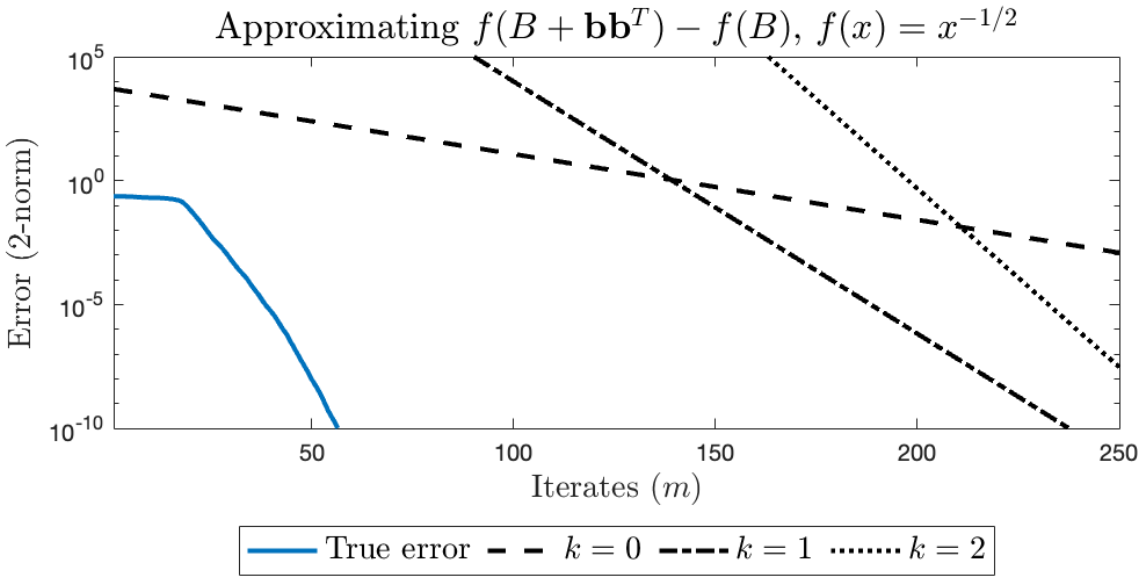

Figure 5.4: Approximating a rank-one update with outliers.

Although the constants in this result grow rapidly with k, we note that for small, fixed k, the improvement on Proposition 5.2 may be considerable, especially in the case that $\rho_{max} = 1 + \epsilon$ with $0 < \epsilon \ll 1$ when one uses $[a, b]$ as the interval containing all of the eigenvalues of B without considering outliers, or the bulk of the spectrum is bounded away from the small outliers.

We now revisit the previous example with this result in hand. We focus on the case when B has reversed logarithmically spaced eigenvalues on $[0.01, 10]$. Note that for $n = 500$, this means that $\lambda_1 = 0.01$, $\lambda_2 = 2.129\cdots$, and $\lambda_j \in [3.799\cdots, 10]$ for all remaining j. We plot bound Proposition 5.3 for $k = 0$ (which corresponds to the original bound Proposition 5.2) as well as $k = 1$ and $k = 2$, in which the interval $[a, b] = [0.01, 10]$ can be replaced by $[2.129, 10]$ and $[3.799, 10]$, so that ρ_{max} may become much larger. Clearly, the bound suffers drastically from the rapid growth of the constants with respect to k; nevertheless, in this example $k = 1$ shows a drastic improvement once $m \approx 175$ since the rate of convergence increases so drastically.

Chapter 6

Numerical experiments with networks

Chapter overview

This chapter marks a shift in the tone of the dissertation from analysis to practical experimentation. We are interested in this chapter and its successor in applying Lanczos-based procedures to matrix function problems arising in network science. More specifically, in this chapter, we review some common structural assumptions and discuss this structure in example datasets (both synthetically generated and drawn from a popular database). The central point of this chapter is that the number of matrix-vector products required for accurate computation of matrix function quantities via Lanczos procedures often remains remarkably small even for networks with a large spread of eigenvalues due to its spectral adaptivity. As a consequence, the run-time of Lanczos-based procedures is almost linear in the number of edges in the network. We compare the practical convergence rate with the expected degree required of *a priori* methods. We perform our experiments in MATLAB.

6.1 Matrix structure in network analysis

We begin this chapter with a quick overview of important structural properties that are common in networks encountered in applications together with the practical implications for linear algebraic algorithms. We've included a visualization in Figure 6.1 of a few synthetically generated networks, the sparsity pattern of their adjacency matrices, and histogram plots of the eigenvalue distributions of the adjacency and Laplacian matrices.

Low-average degree/Sparsity In many networks, the average node degree remains small independent of the number of nodes in the graph. These networks are often called *sparse*. This immediately implies that the adjacency and Laplacian matrices defined in Section 2.4 are sparse matrices, and consequently matrix-vector products can be computed very efficiently.

Low maximum degree/bandedness Beyond sparsity, in some applications it is reasonable to expect that the maximum degree of nodes in the network remains small independent of the number of nodes. These networks exhibit spatial localization, in which it is difficult to move from one region of the network to a distant region in few steps. An important example is road networks. This spatial localization translates to banded adjacency and Laplacian matrices. Various efficient sparse algorithms exist for banded matrices [Saa03]. Notable for the discussion in this work is the fact that low order polynomials inherit bandedness as well, and matrix functions that are approximated well by polynomials inherit nice off-diagonal decay properties [BR07, BBR13].[1]

Community structure/spectral outliers Many networks exhibit community structure wherein the network can be partitioned into families of nodes within which there is high connectivity but outside of which the network has low connectivity [FSS16]. The matrices associated with these networks may be nearly block-diagonal, and when this feature is strong, it may manifest a small number of "spectral outliers"—eigenvalues of the adjacency or Laplacian matrices that are well-separated from the bulk of the eigenvalues [vL07]. As discussed previously, Lanczos-based procedures are very powerful at adapting to spectral structure such as this.

Scale-free networks/spectral outliers Many networks have have a scale-free structure in which a few nodes have drastically more edges than the average [BB03, EG17]. This is especially common within social networks. Most nodes within the network are of low degree, but there are some very high degree nodes—often, the degrees of the nodes obey a power-law distribution. Remarkably, the eigenvalues of the associated matrices often have similar structure wherein the spread of

[1] We feel that it is very important to mention this separately from sparsity of the network. In fact, in the literature it seems often to be stated that sparse networks can be brought into banded form, or that polynomials of sparse networks are also sparse; however, there is a strong implicit assumption here that the network is not merely sparse but also exhibits this spatial localization. If there is just one high degree node, then the sparsity of low-degree polynomials of the matrix can be destroyed rapidly.

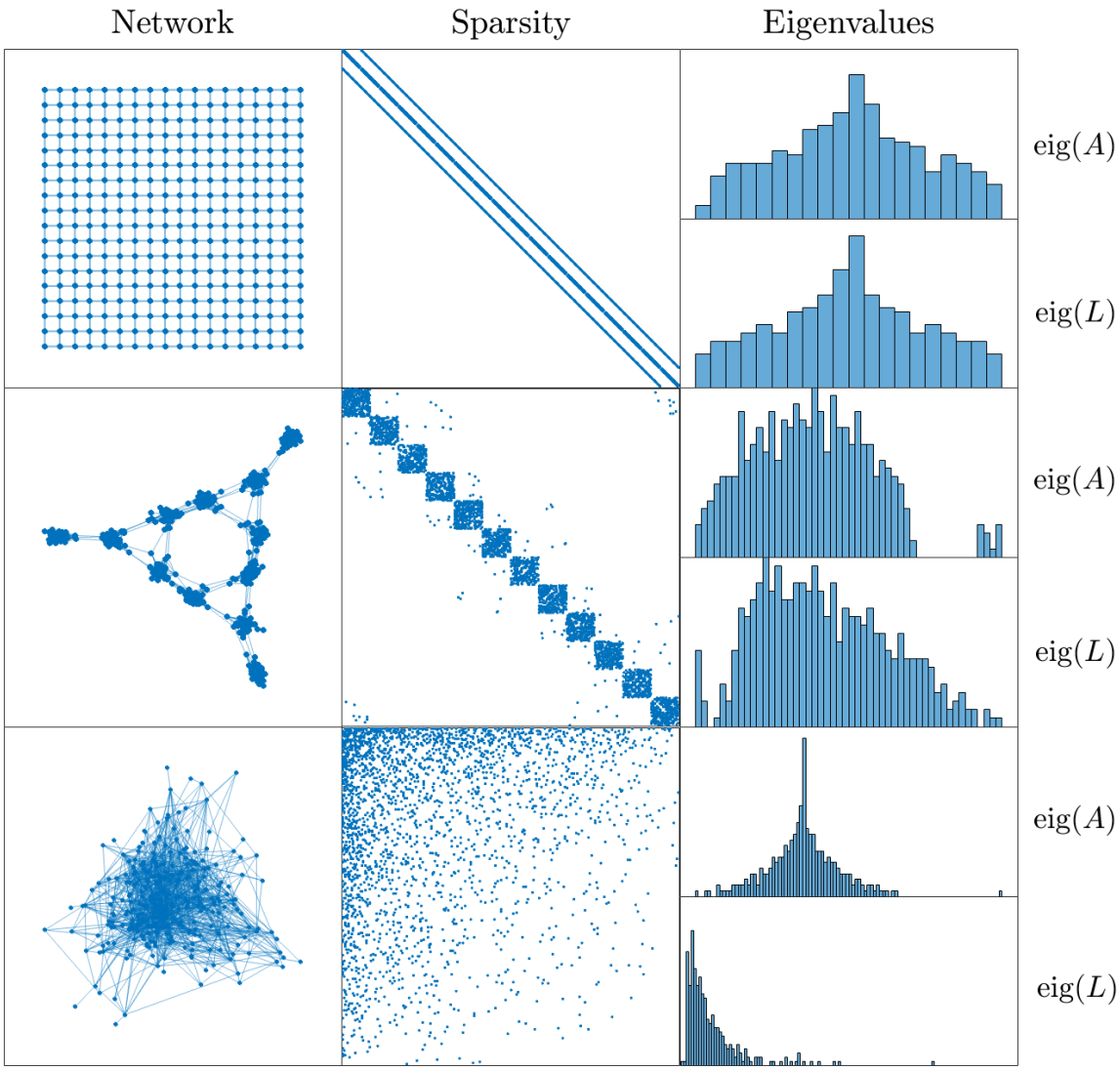

Figure 6.1: Example networks.

eigenvalues is large but the number of eigenvalues in the extremal end of the spectrum are rare.

6.2 Description of data

In this section, we give an overview of the network data that we use for our experiments. Some of the networks that we generate are synthetically generated; others are taken from the SuiteSparse matrix collection [KAB+19], which has a large number of standard large-scale test networks. We have provided a summary in table form for these real datasets in Table 6.1, but we include some more detailed comments below.

6.2.1 Synthetically generated networks

We briefly overview the models that we use to generate synthetic networks that will be used for experiments.

2D Square Grid. Fix $n = kr$. For each $1 \leq i \leq k, 1 \leq j \leq r$, we construct a node $v_{(i,j)}$ with neighbors $v_{(i\pm1,j)}$ and $v_{(i,j\pm1)}$, but we remove (or ignore) the boundary nodes with $i = 0, k+1$ or $j = 0, r+1$. It has similar spectral properties to road networks and other networks that are embedded in 2D space.

Erdos-Renyi network. An Erdos-Renyi network is a random network generated by including each edge independently with a fixed probability p [EKYY13]. Fix n and select $0 < p < 1$ as the probability of edge existence. Then $A_ij = 1$ with probability p or $A_{ij} = 0$ with probability $1 - p$ independent of all other edges. This is perhaps the most famous random network. It exhibits a transition where it is connected with high probability when $np > \log n$. We will work in the sparse regime where the network is fully connected, $np = k \log n$ for various k which is small but larger than 1. Notably, the network then has $O(kn \log n)$ edges. The eigenvalues of the adjacency matrix obey a semicircle law with support in $[-2\sqrt{np(1-p)}, 2\sqrt{np(1-p)}]$ together with an extreme eigenvalue of order np. Note that np is also the expected degree of a vertex in the network, so most eigenvalues are bounded in magnitude by $\sqrt{d_{average}}$.

Stochastic block model. The stochastic block model is popular for modeling community structure in networks [Abb18]. In this work, we use the following version. Let the network be partitioned into k communities of sizes $n_1, n_2, \cdots, n_k$. Partition the adjacency matrix into a corresponding block form. Define a symmetric k-by-k matrix P where P_{rs} represents the probability that a node from community r is connected to a node from community s. Letting $P_{rr} \gg P_{rs}$ if $r \neq s$ represents stronger connectivity within a community. For an entry A_{ij} of the adjacency matrix, if node i is in community r and node j is in community s then we sample $A_{ij} = 1$ with probability P_{rs} and $A_{ij} = 0$ with probability $1 - P_{rs}$. Each entry is sampled independently of the others.

Barabasi-Albert. Barabasi-Albert networks [BB03] are generated via preferential attachment, wherein a new node joining the graph is more likely to connect to nodes that already have high degree than those that have low degree. This simulates power-law degree distributions and models phenomena in social networks.

We grow the network one node at a time. Each new node forms k edges that attach to an existing node j with probability proportional to d_j where d_j is the degree of the node. Our particular implementation lets k be a random bounded integer to introduce some degree heterogeneity, and we typically use a small Erdos-Renyi network as the seed before engaging in the preferential attachment procedure.

6.2.2 Real-world networks

Here we describe the real networks that we use for our experiments. We have loaded these networks through SuiteSparse, excepting the Minnesota road network which is available in MATLAB. For each network, we keep only the largest connected component. Some summary statistics are available in Table 6.1.

Road networks. Road networks are sparse with low maximal degree and high diameter. We load four road networks: the Minnesota road network from MATLAB together with the California, Pennsylvania, and Texas road networks from the SuiteSparse collection. The Minnesota road network has $n \approx 2500$ nodes, whereas the others have $n \approx 10^6$ nodes.

Social networks. Social networks have low average degree but high maximal degree and low diameter. We load four social networks from the SuiteSparse collection: Youtube, Gowalla, Astrophysics Citation, and Condensed Matter Collaboration.

Other. We include a couple of miscellaneous networks. We include the US Power Grid and Amazon Product networks from the SuiteSparse collection. These both have community structure and therefore spectral inhomogeneity.

6.3 Experiment 1: Computing $\sin(A)\mathbf{b}$ and $\exp(-L)\mathbf{b}$

Now we consider a first experiment. Our goal here is to illustrate the favorable scalability of the Lanczos approximation due to its spectral adaptivity, especially in comparison with explicit polynomial approximations wherein the approximating polynomial is formed *a priori* using the extremal eigenvalues of the matrix.

We compare what m leads to accurate computation with the Lanczos procedure relative to the value of m required to obtain an approximating polynomial of the same accuracy using Chebfun's minimax algorithm on an interval that contains the

Network	n	nnz(A)	$\lambda_{\min}(A)$	$\lambda_{\max}(A)$	$\lambda_{\max}(L)$
Minnesota	2,640	6,600	-3.15	3.23	6.88
Pennsylvania	1,090,000	3,080,000	-3.93	4.42	10.4
Texas	1,350,000	3,760,000	-3.95	4.91	13.2
California	1,960,000	5,520,000	-3.93	4.64	13.2
Astrophysics	17,900	394,000	-28.3	94.4	505
Condensed Matter	21,400	183,000	-15.8	37.9	280
Gowalla	197,000	1,900,000	-122	171	14,700
YouTube	1,130,000	5,980,000	-177	210	28,800
US Power	4,940	13,200	-4.50	7.48	20.1
Amazon Products	335,000	1,850,000	-23.2	24.0	550

Table 6.1: Network summary. We list the number of nodes n, number of non-zeros in the adjacency matrix nnz(A) (which is 2 times the number of edges), largest and smallest eigenvalues of the adjacency matrix, and largest eigenvalue of the Laplacian. Numbers have been rounded to 3 significant digits for readability.

eigenvalues of the relevant matrices. We compute $\sin(A)\mathbf{b}$ and $\exp(-L)\mathbf{b}$ for each network in our dataset (both synthetic and real), where $\mathbf{b}$ is a random vector. We measure the value of m in the Lanczos procedure required to obtain an accuracy of 10^{-10}, which we label m_{act}, and we compare this with m_{pred}, which is the degree of the polynomial (optimal in the uniform error norm) that obtains the same error on the whole interval $[\lambda_{min}(A), \lambda_{max}(A)]$ or $[0, \lambda_{max}(L)]$. As a first plot, we show in Figure 6.2 a loglog plot of the raw values of m obtained versus the number of nodes in the network and make two observations. First, that the growth rate is very slow in all cases when viewed as a function of n. We have plotted as a dashed line the reference function $m = 20n^{1/5}$, which seems to capture the behavior of the worst cases and grossly overestimate the best cases. This is because the functions used $\sin(x), e^{-x}$ are entire functions and thus exhibit superlinear convergence. As a second observation, the blue crosses representing m_{act} tend to be much smaller than their counterpart red circles m_{pred}, which is because many of these networks have spectral outliers, so that the Lanczos procedure is able to adapt to the spectra of the underlying matrices and outperform a priori approximation on the whole interval. We illustrate this difference more clearly in a second figure, Figure 6.3 where we give a loglog plot of the ratio $m_{\text{pred}}/m_{\text{act}}$ against number of nodes in the network. We also include distinct markers for each type of network. Note that the plots are on a loglog scale due to the wide ranging network sizes and values of $m_{\text{act}}/m_{\text{pred}}$. Importantly, for road networks and

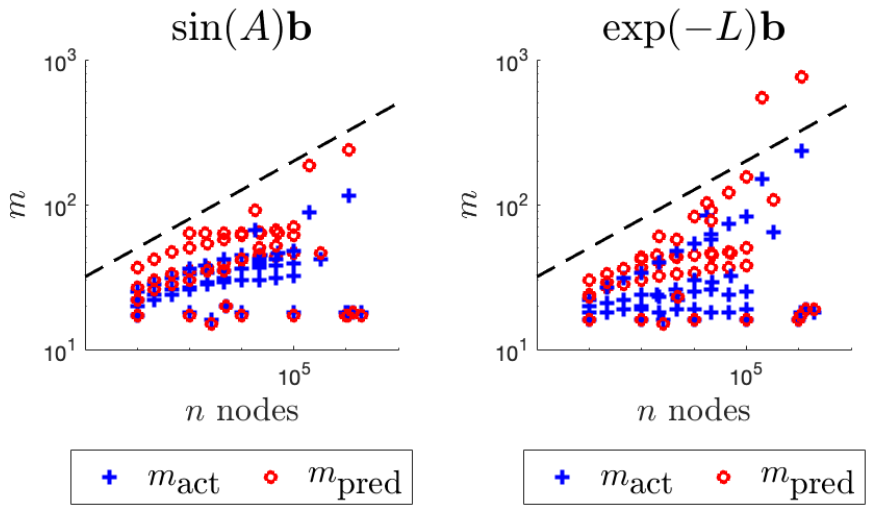

Figure 6.2: Comparison of Lanczos convergence with *a priori* expectation. Note that the axes are on a loglog scale and that the reference line depicted is $m = 20n^{1/5}$

the grid networks, whose eigenvalues densely fill the field of values of the matrix, there is little difference between the two approaches, but for networks that have spectral outliers, the difference can be massive, especially as the network size increases, as is most notable for the social networks and networks grown with preferential attachment (which have approximate power-law eigenvalue distributions). Note that the largest social networks see an improvement on the order of $m_{\text{act}}/m_{\text{pred}} \approx 1/3$ when computing $e^{-L}\mathbf{b}$.

6.4 Experiment 2: Timing

As a second experiment, in order to illustrate the near-linear time complexity of the Lanczos procedure with respect to edges in the network, we measure the time taken to apply $m = 160$ iterates of the Lanczos procedure to each network against the number of edges in the network. Note that this m is sufficient to compute $\sin(A)\mathbf{b}$ or $\exp(-L)\mathbf{b}$ for every network in the dataset excepting the YouTube network, which required $m \approx 240$. The results are depicted in Figure 6.4, where we plot the time t in seconds against the number of edges in the network. Note once again that this is a loglog plot due to the wide range of values; nevertheless, it appears each family of networks basically follows the reference line $y = 10^{-5}x$, where y is the time and x is the number of edges.

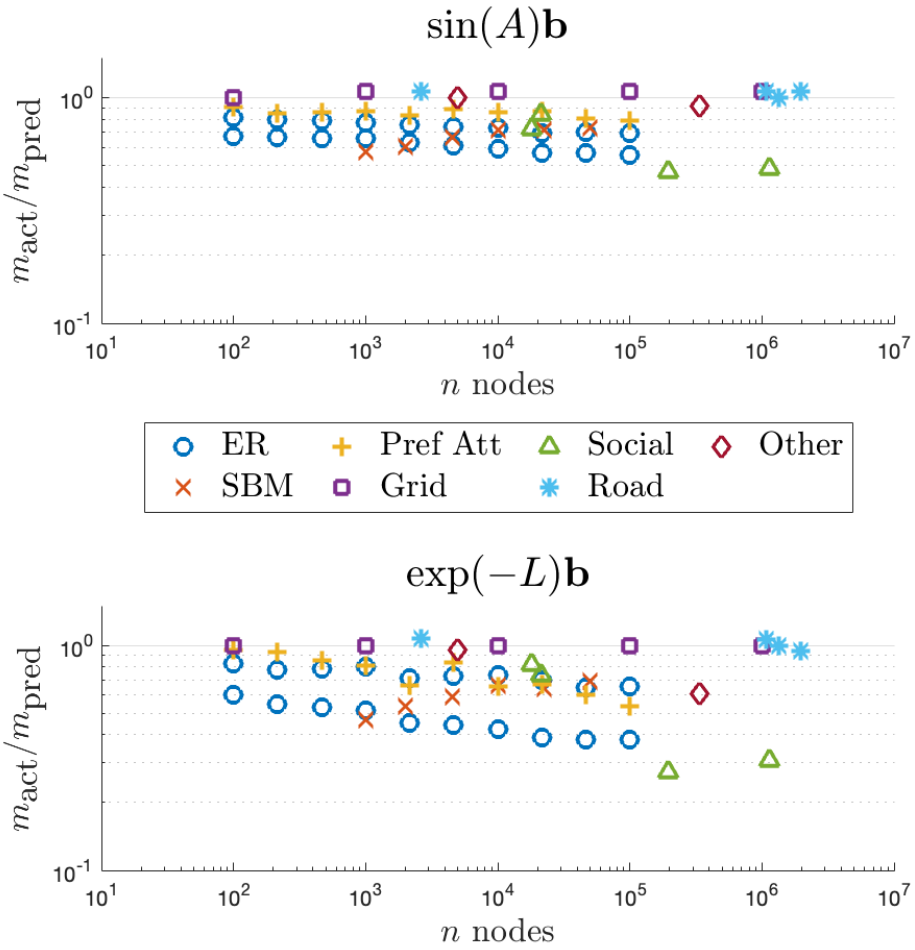

Figure 6.3: Second comparison of Lanczos convergence with *a priori* expectation.

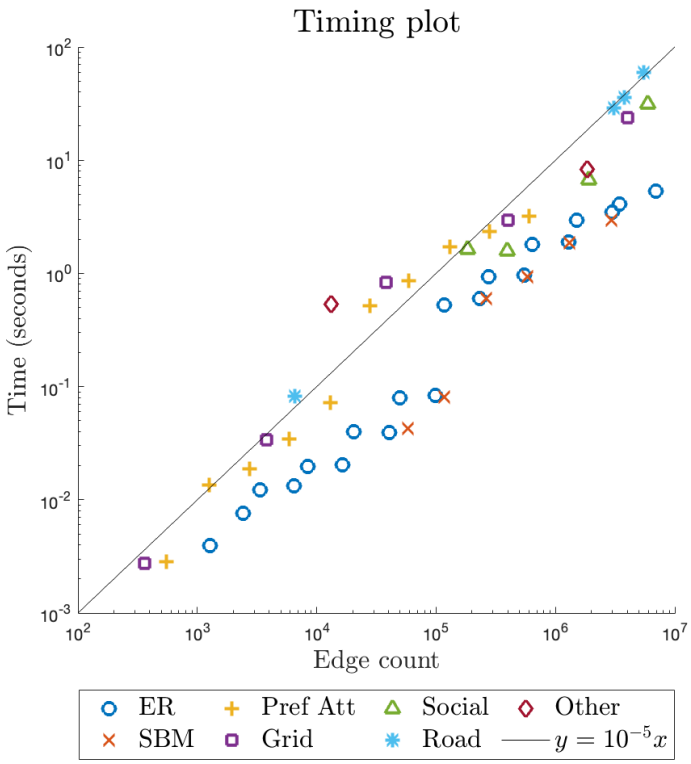

Figure 6.4: Timing experiment. Note the axes are both log-scale.

6.5 Experiment 3: Extra digits

As a final experiment, we consider the computation of the bump function $f(x) = e^{-1/x^2}$ applied to the Laplacian L. Notably, this function is smooth (having infinitely many derivatives) but is not analytic in any neighborhood of the Laplacian's spectrum $[0, \lambda_{max}(L)]$. This means that we may expect convergence faster than $O(m^{-k})$ for any $k \geq 1$. This is strictly worse than the convergence rate of complex analytic functions, and m may be rather large regardless of the network. Still, we might ask how much better the Lanczos procedure performs than an a priori method. One way to measure this is to compare the the ratio of the error in the Lanczos procedure to the error of the minimax polynomial. We track this error for increasing values of m. The result is depicted in Figure 6.5. This plot effectively shows *how many extra digits of accuracy do we get as a function of m by using Lanczos instead of a priori methods.* We note that there is a consistent downward trend for each of the networks tested, and by $m \approx 160$ each example seems to have gained an extra digit or two of accuracy.

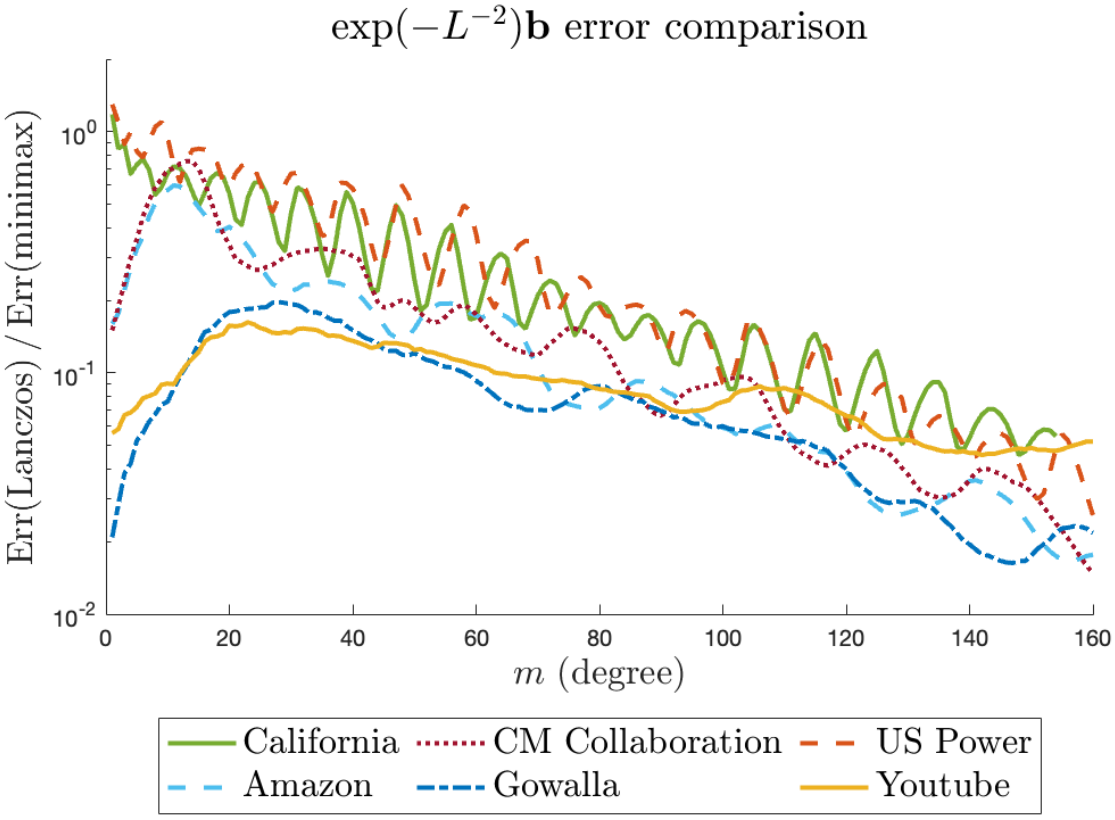

Figure 6.5: Extra digits experiment.

Chapter 7

Spectral sensitivities

Chapter overview

In this chapter we propose a framework for computing network edge centralities based on the derivative of the trace of a matrix function of the graph Laplacian, with a focus on the matrix exponential and the resulting "heat kernel." We show that the proposed measure interpolates between a local degree-based edge centrality and a global eigenvector-based edge centrality. Moreover, we show that the centrality is efficiently and accurately computed for individual edges via Algorithm 7; perhaps most notably, however, we propose a procedure based on randomized diagonal estimation that estimates the measure for all edges at once.

7.1 Spectral sensitivities

A simple yet profound problem in network science is the identification of nodes and edges that contribute significantly to the structural properties of the network [FSS16, New18, Est12b]. This includes, for example, locating the best spreader nodes in epidemic processes, bottlenecks or bridges that control information flow, and vulnerable regions whose removal leads to network fragmentation. As we have mentioned in Section 2.4, matrix functions have been successful in generalizing measures based on counting random walks or other dynamical processes. Although emphasis has traditionally been placed on identifying nodes of interest, it is also important to characterize edges, such as those whose removal disrupts connectivity or alters the behavior of dynamics. Many existing measures are either global, summarizing the overall structure of the network, or local, depending only on a small region of the network. Ideally, one seeks measures that interpolate between these two extremes, capturing contributions from both a local neighborhood and the global positioning

in the network. In the remainder of this section, we will describe two examples of edge centralities, one purely "local" based on degrees and one purely "global" based on an extremal eigenvalue of the graph Laplacian. Finally, we will introduce a more general framework based on the traces of matrix functions that can be viewed as interpolating between these two. In the subsequent sections, we discuss computation of these measures.

We begin with a very simple degree-based local measure of edge centrality. Informally, an edge is important if both of the nodes upon which it is incident are also important, leading to an edge-degree centrality that simply sums the degrees of the nodes upon which the edge is incident. Formally, we may assign a centrality D_k to the kth edge incident on nodes i and j via

$$D_k = d_i + d_j.$$

On the other hand, an edge may be considered vulnerable if this measure is small, since removal of the edge would strongly impact the nodes to which it is connected.

For a more global measure, we follow the idea of *dynamical importance* introduced in [ROH06] and further studied as *spectral impact* in [MSN10]. Since the second smallest eigenvalue of the graph Laplacian, the algebraic connectivity, is a strong measure of the connectivity of the network, we may ask how much the removal of an edge decreases the algebraic connectivity. We may then measure the importance of the kth edge

$$I_k = -\frac{d}{dt}\left[\lambda_2(L - t\mathbf{x}_k\mathbf{x}_k^T)\right]_{t=0},$$

where $\mathbf{x}_k$ is the transpose of kth row of the incidence matrix X from Section 2.4. Recall that if the kth edge is oriented from node i to node j then $\mathbf{x}(i) = 1$ and $\mathbf{x}(j) = -1$ with other entries zero. Note the negative sign ensures the measure is positive. Using Theorem 3.7, so long as λ_2 is unique with associated eigenvector $\mathbf{v}_2$,

$$\frac{d}{dt}\lambda_2(L - \mathbf{x}_k\mathbf{x}_k^T) = -\mathbf{v}_2^T\mathbf{x}_k\mathbf{x}_k^T\mathbf{v}_2 = -(\mathbf{v}_2(i) - \mathbf{v}_2(j))^2.$$

Thus, we obtain finally

$$I_k = (\mathbf{v}_2(i) - \mathbf{v}_2(j))^2,$$

where $\mathbf{v}_2$ is the normalized eigenvector associated with the second smallest eigenvalue of the graph Laplacian. Often, λ_2 is referred to as the Fiedler eigenvalue and $\mathbf{v}_2$ is referred to as the Fiedler eigenvector.

Notably, both of these measures are computable. The degrees of the nodes are given by $A\mathbb{1}$, which can be computed in time linear in the number of edges. Computing

extremal eigenvectors of a sparse matrix is often feasible especially if the extremal eigenvalue is well-separated from the remainder, for example using the Lanczos procedure or other polynomial filtering techniques [Saa03]. Nevertheless, there is a weakness for both of these measures. The local measure may be short-sighted, and the global measure may not capture well the significance of edges that are locally important. This can be a problem for networks that have multimodal structure, for example leading to a large number of small eigenvalues.

Now we finish this section by introducing measures based on the trace of a matrix function. We call these *spectral sensitivities*. For concreteness, we will use the heat kernel $f(L) = e^{-tL}$, though similar comments apply to any function that is positive on the positive real axis and rapidly decaying away from $x = 0$. Recall that the traces of matrix functions are closely related to measures of network robustness as described in Section 2.4. Therefore, we may view edges as important if their removal would drastically reduce the trace of the matrix function. Our proposed measure evaluates the sensitivity of the trace of the matrix function to perturbation:

$$S_k = -\frac{d}{dt}\left[\mathrm{Tr}(f(L - t\mathbf{x}_k\mathbf{x}_k^T))\right]_{t=0}.$$

Notably, due to Proposition 3.5, this quantity is given by the quadratic form

$$S_k = -\mathbf{x}_k^T f'(L)\mathbf{x}_k. \tag{7.1}$$

This means that we can efficiently evaluate the importance of an edge using Algorithm 7. If we wished to compute the impact of the full edge removal, we could replace the derivative with the discrete change in the trace $\mathrm{Tr}(f(L - \mathbf{x}_k\mathbf{x}_k^T) - f(L))$ and compute the result via Algorithm 8.

One of the attractive features of matrix-function-based measures is that they interpolate between local and global behavior. In Figure 7.1 we show D_k, I_k and S_k on a rectangular grid. Notably, S_k can be viewed as blending the measures D_k and I_k. We make this precise by showing now that S_k may be viewed as interpolating between D_k and I_k (up to a linear transformation and some edge-independent constants).

Proposition 7.1. *Let $f(x) = e^{-tx}$ and let edge k be incident on nodes i and j Suppose $0 < t \ll 1$. Then*

$$S_k = 2t - t^2(D_k + 2) + O(t^3). \tag{7.2}$$

Moreover, if $t \gg 1$, then

$$S_k = te^{-t\lambda_2}(I_k + O(e^{-t(\lambda_3 - \lambda_2)})). \tag{7.3}$$

Thus, S_k may be viewed as interpolating between D_k when $t \approx 0$ and I_k when $t \gg 1$.

Proof. First, suppose t is small, so that using the taylor expansion of $f'(x) = -te^{-tx}$ we have

$$S_k = -\mathbf{x}_k^T(-t(I - tL + O(t^2L)))\mathbf{x}_k = 2t - t^2\mathbf{x}_k^T L\mathbf{x}_k + O(t^3).$$

Recall that $L = D - A$, so that $\mathbf{x}_k^T L\mathbf{x}_k = d_i + d_j + 2 = D_k + 2$. Thus, (7.2) follows.

Next, recall from Equation (2.1) that we may write

$$f'(L) = \sum_{i=1}^{n} f'(\lambda_i)\mathbf{v}_i\mathbf{v}_i^T.$$

Moreover, $\mathbf{x}_k^T\mathbf{v}_1 = 0$ since the null-space of the graph Laplacian of a connected network is one-dimensional and proportional to the vector of all ones. Thus,

$$S_k = -\mathbf{x}_k^T f'(L)\mathbf{x}_k = t\sum_{i=2}^{n} e^{-t\lambda_i}(\mathbf{x}_k^T\mathbf{v}_i)^2.$$

Notably, $(\mathbf{x}_k^T\mathbf{v}_2)^2 = I_k$, so that factoring out $e^{-t\lambda_2}$ yields (7.3). □

To conclude this section, we remark that the idea of using the derivative of the trace of a matrix function in the direction of an edge removal in order to rank edges is not a particularly new idea, but we believe that it will be invaluable to the network science community to point out the generality of this framework and to connect it to the computational tools presented within this work. Measuring the sensitivity of a generic measure of criticality under removal of an edge has been considered in many contexts [FSS16, Chapter 4]. In fact, the famous effective resistance of an edge [KR93], which is given in our notation as $\mathbf{x}_k^T L^\dagger \mathbf{x}_k$ (where $L^\dagger$ indicates the Moore-Penrose pseudoinverse) may be related to the sensitivity of $\mathrm{Tr}(\log^\dagger(L))$ to edge perturbation (where the $\log^\dagger$ indicates that the function is only applied to the positive eigenvalues of the matrix). Note that $\mathrm{Tr}(\log^\dagger(L))$ is the logarithm of the product of the non-zero eigenvalues of L, which measures the number of spanning trees in a network according to Kirchoff's famous theorem. Relatedly, the derivative of $\mathrm{Tr}(L^\dagger)$ under edge removal is related to power dissipation in the network [FSS16, Chapter 4]. A similar idea has been applied more recently to the Von-Neumann entropy of a network, which uses $\mathrm{Tr}(L\log L)$ [KDDMT24]. It is often suggested (for example see [KDDMT24]) that evaluation of such edge metrics would require repeated eigendecomposition of large-scale matrices for every edge removal in the network. The proposed solution to make the metrics computable is often to use a low-order polynomial approximation such as a low-order Taylor expansion, or to base the approximation on first-order perturbation of a small number of extremal eigenvectors.

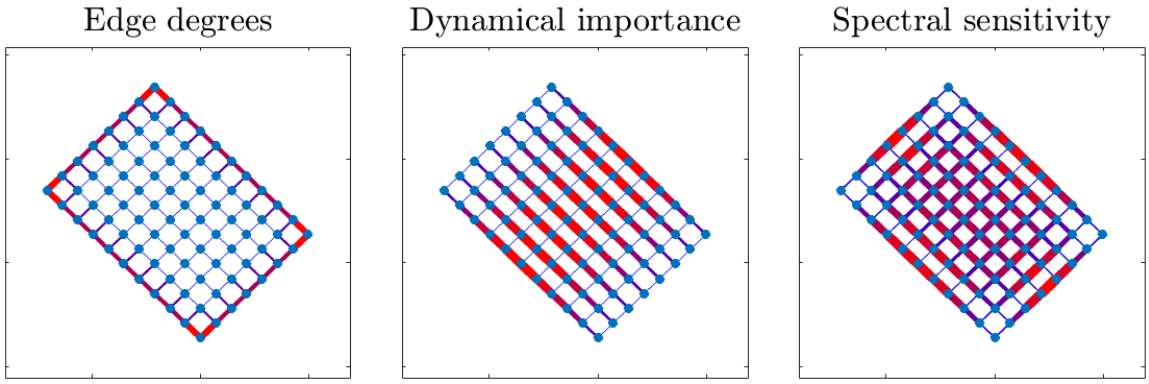

Figure 7.1: Edge importances on a rectangular grid.

We push back strongly against these assumptions! One does not need to compute repeated eigendecompositions—in fact, one does not need to compute a single eigendecomposition. As we have shown, the change in the trace or its sensitivity to edge removal can be estimated in $O(m \cdot \operatorname{nnz} L)$ time using the methods from Chapter 3 without explicitly forming a polynomial approximation and without knowledge of the eigenvalues of the matrix.

Of course, this is still slow (though much faster than repeated eigendecomposition) if one wishes to rank all of the edges in the network. Our next section intends to remedy this by showing that in fact we can provide a rough estimate of all of the edge importances simultaneously using randomized diagonal estimation.

7.2 Computation of all sensitivities "all-at-once"

Now, let us show how we can extend this approach to the computation of the importance of all edges all-at-once. Importantly, the definition of S_k with

$$S_k = -\mathbf{x}_k^T f'(L)\mathbf{x}_k$$

means that the list of all sensitivities S is given by

$$S = -\operatorname{Diag}(X f'(L) X^T). \tag{7.4}$$

We could compute these diagonal entries one by one using methods from Chapter 3. It should be noted that if the graph is small enough, this provides a way for us to obtain all of the edge centralities after only one matrix function computation of $f'(L)$. Alternatively, when L is big, we may try to estimate the diagonal. Even though $X f'(L) X^T$ may be an enormous matrix, it can be queried for matrix-vector products

via application of X^T, application of $f'(L)$ implicitly through Algorithm 5, and then reapplication of X. Each of these applications will be fast for sparse networks assuming f' can be approximated by polynomials.

The tool that we need to realize this idea is a randomized diagonal estimator [BKS07, BN22]. One idea is to sample s random Rademacher vectors $\mathbf{r}_1, \cdots, \mathbf{r}_s$, where each entry of $\mathbf{r}_j$ is ± 1 with equal probability. Then an unbiased estimator of the diagonal of a matrix B may be constructed via

$$\mathrm{Diag}(B) \approx \frac{1}{s} \sum_{j=1}^{s} (\mathbf{r}_j \odot B\mathbf{r}_j).$$

Note that B is only accessed through matrix-vector products. For a careful analysis of the properties of this estimator, as well as alternative estimators, we refer the reader to [BN22]. We note only that if $s = O(1/\epsilon^2)$, then the estimated diagonal entries have a relative error of ϵ with high probability.[1] Even though this means that obtaining high accuracy edge importance via this procedure is may be difficult, there are many networks for which the structures are so varied that the edge importances may span several orders of magnitude, so that even $\epsilon = 0.1$ for example may be sufficient for obtaining approximate rankings of the "most important" edges.

We propose in Algorithm 11 an algorithm for estimating (7.4). It requires s matrix-vector products with $Xf'(L)X^T$. If there are E edges in the network, then multiplication by X, X^T requires $O(E)$ time, and application of $f'(L)$ to a vector requires $O(m^2 n) + O(mE) + O(m^3)$ time when we use Algorithm 5. Since we apply $Xf'(L)X^T$ to s vectors, this requires an overall complexity of $O(m^2 ns) + O(smE) + O(sm^3)$. Note that the accuracy guarantees of s are independent of the size of the matrix, and as discussed in Chapter 6 m remains remarkably small even for large networks.

7.3 Examples

We conclude this chapter with some examples. First, we display the measures D_k, I_k, S_k, and $\hat{S}_k$ in Figure 7.2, where we have used $f(x) = e^{-5x}$ as the function in Equation (7.1). Since this network has only $E \approx 3300$ edges, it is possible to compute S_k to machine precision via Algorithm 7 for all of the edges in just a few seconds. We include also the randomized approximation computed via Algorithm 11 with $s = 50$.

[1] There are better estimators available, but in a desire to keep the dissertation self-contained we introduce this since it seems to be the simplest procedure.

Algorithm 11 EdgeImportances(f', L, X, m, s)
Estimating all spectral sensitivities simultaneously

Input: Laplacian L, incidence matrix X, derivative f', Lanczos steps m, number of Rademacher probes s.
Output: $\widehat{S} \approx S = -\operatorname{Diag}(X f'(L) X^T)$.
Sample $R \in \{\pm 1\}^{E \times s}$ with i.i.d. Rademacher entries.
Compute $T \leftarrow X^\top R \in \mathbb{R}^{n \times s}$
Compute $W \approx f'(L)T$ using m steps of Lanczos for each column
$Q \leftarrow XW \in \mathbb{R}^{E \times s}$
$\widehat{S} \leftarrow \dfrac{1}{s}(Q \odot R)\,\mathbb{1}$
Complexity summary: $O(m^2 ns) + O(sm \operatorname{nnz} L) + O(sm^3)$

Notably, there is little visual difference between rankings obtained from Algorithm 11 from their true values.

In Figure 7.3, we include scatter plots of each of $D_k, I_k, \hat{S}_k$ and S_k against S_k for the Minnesota road network. This shows that S_k is somewhat correlated with both D_k and I_k for some of the edges, but provides very different results for the majority of the network. Moreover, we see that $\hat{S}$ as computed by Algorithm 11 provides a "noisy" approximation to S_k. We follow this up with similar plots on several larger networks in Figure 7.4. We cannot compute S_k accurately for all edges in a reasonable amount of time so we only compare D_k and I_k to $\hat{S}_k$. The largest network used was the CA Road network with $n \approx 2 \times 10^6$ nodes and $E \approx 3 \times 10^6$ edges, which ran with $m = 60$ and $s = 50$ in about 15 minutes.

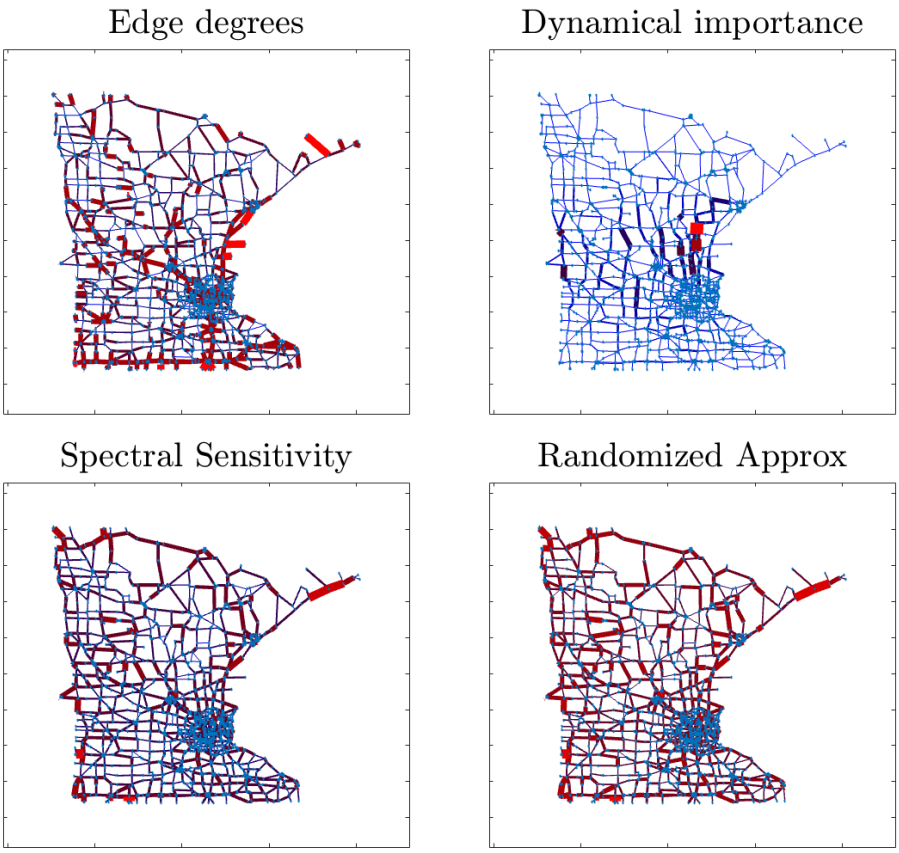

Figure 7.2: Edge importances on Minnesota road network.

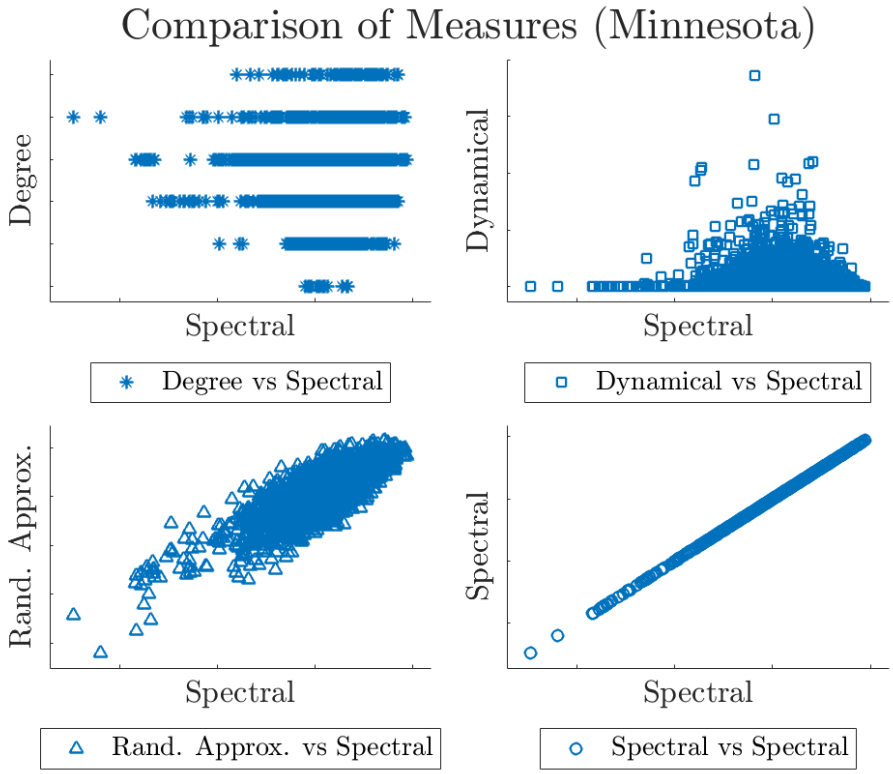

Figure 7.3: Comparison of edge importances on Minnesota road network.

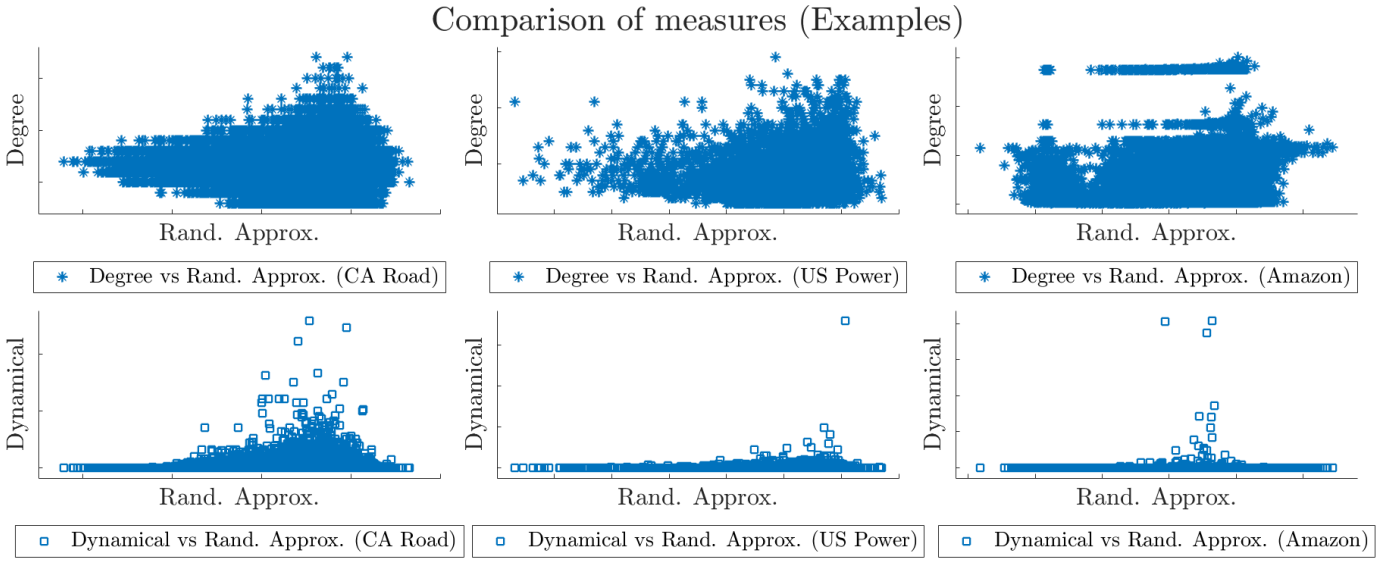

Figure 7.4: Comparison of edge importances on additional example networks. We compare D_k and I_k to $\hat{S}_k$ as computed by Algorithm 11 with $s = 50$ for a number of other real-world networks. We use the CA road network, US Power, and Amazon product networks from Table 6.1.

Chapter 8

Conclusion

This dissertation followed a simple trajectory. We began in Chapter 2 with a review of matrix functions, some iterative methods for computing matrix-function related quantities, a tour of polynomial approximation theory, and matrix representations of networks together with some applications of matrix functions for network analysis. In Chapter 3, we studied more carefully the relationship between scalar Gauss quadrature and its use in approximating quantities related to the matrix function. Perhaps most notable was the observation in Section 3.3 that the derivative of the trace of a matrix function in a rank-one direction can be represented and computed with Gauss quadrature via the Lanczos procedure, and that the same analysis applies to rank-one updates of the trace as discussed in Section 3.4. We continued in Chapter 4 by showing that each of the methods presented in Chapter 3 has a parent procedure that can be tied precisely to a polynomial interpolation problem. In particular, we showed that the recently introduced algorithms from [BKS18, Kre19, KKRS21], when applied to symmetric matrices, have a precise characterization via bivariate interpolation on a tensor product grid of zeros of orthogonal polynomials. We followed in Chapter 5 and Chapter 6 with numerical experiments investigating how the algorithms behave when applied to various matrices; notably, these procedures adapt to the spectral properties of the input matrix, and in Chapter 6 this manifested in the number of iterates required by the Lanczos procedure outpacing the number of iterates that would be demanded of *a priori* methods that approximate the polynomial using only the extremal eigenvalues of the matrix. Finally, in Chapter 7, we described how matrix functions provide a theoretically attractive way to assess the importance of an edge in a network via *spectral sensitivities*. Crucially, the computation of the spectral sensitivity of a single edge can be done using the methods we introduced in Chapter 3 in time nearly linear in the number of edges of the network. No large-scale eigendecomposition, low-order polynomial approximation, or assumption about the spectral distribution of the matrix

was necessary. Moreover, we discussed how randomized diagonal estimation could be used to approximate the spectral sensitivities of all of the edges simultaneously with similar complexity, albeit with low accuracy.

There are many topics that we didn't cover, both on the algorithmic side and on the side of applications. We wish to conclude here by outlining some directions of further reading and further research. First, we observe that bounds based on uniform approximation theory, although they may explain in part the accelerated convergence of Lanczos procedures (as we saw in Chapter 5), they are very pessimistic. One approach to do better is based on integral representations and relating the procedures to the solution of linear systems [CGMM22]. Nevertheless, this requires assumptions of smoothness that may not hold. It would be very interesting to provide a more direct treatment of the adaptivity of Lanczos procedures by studying more carefully the adaptivity of Gauss quadrature and polynomial interpolation on the zero sets of polynomials for atomic measures. Classically, the focus has been on certain weight functions related to famous families of orthogonal polynomials. Since these weight functions have dense support, there has not previously been a need to study atomic measures; however, it seems that it should be possible to provide a more precise characterization of the error in Gauss quadrature and polynomial interpolation when a measure is supported on an arbitrarily large but discrete measure.

A key assumption that we made throughout this dissertation was that the function f should be well-approximated by polynomials and that the input matrix B was symmetric. Many matrices are not symmetric, and many functions cannot be approximated well by polynomials. It is possible however to extend these approaches to incorporate rational functions and variants of the Arnoldi procedure can be used for non-symmetric problems [Gü13, Sch16]; however, the use of rational Krylov methods requires the solution of linear systems. Recently, tremendous strides have been made in finding generic but effective preconditioners for Laplacian matrices [KS16]. Another potential avenue for further research would be to investigate how the ability to solve linear systems with Laplacian matrices can be combined with the theory of rational Krylov methods to construct even more powerful methods for matrix function computation on networks.

We only considered one application in Chapter 7, but the ability to compute low-rank matrix function differences and derivatives should be powerful for numerous areas of network science. For example, an important problem is the embedding problem, in which one associates with each node of the network a low-dimensional embedding in Euclidean space [Ham20]. We discussed in Section 2.4 how matrix functions

have geometric interpretations on networks, and how they can define generalized kernels measuring similarity on the network [FSS16]. One of the most powerful interpretable approaches to embedding is kernel-based PCA, and indeed many data-driven embedding procedures have recently been shown to be closely related to low-rank factorizations of matrix functions of the adjacency or Laplacian matrix [QDM+18]. Perhaps the ability to update matrix functions under small perturbations of the network structure could lead to more efficient methods of updating matrix-function-based graph embeddings.

Another significant problem is the network robustness optimization problem, in which one wishes to select edges for removal or addition in order to improve or diminish some global measure of robustness or connectivity. Eigenvector-based measures have been very popular as for this problem [TPER+12]. Matrix functions provide a more general theoretical framework but are often seen as intractable on large networks (see e.g. [WPKVM14]). Recently, some work has been done to use polynomial Krylov methods for this problem in order to optimize measures such as the total communicability or natural connectivity of a network [AB16, WSMB21, MT23, Sch23]. Nevertheless, there are many possible measures of network robustness, and there are potentially many avenues in this direction to study.

The future for matrix functions is bright. Along with the network science community, they have found numerous applications in the natural and data sciences, and the numerical community is determined to make them accessible. Our hope for the reader is that we made clear some of the principles of polynomial approximation theory that underpin Lanczos-based procedures for computing matrix functions, and that the resulting algorithms seem especially accessible for applications in network science. Thank you for reading!

Bibliography

[AB16] Francesca Arrigo and Michele Benzi. Updating and Downdating Techniques for Optimizing Network Communicability. *SIAM Journal on Scientific Computing*, 38(1):B25–B49, 2016. _eprint: https://doi.org/10.1137/140991923.

[Abb18] Emmanuel Abbe. Community detection and stochastic block models: recent developments. *Journal of Machine Learning Research*, 18(177):1–86, 2018.

[AS14] Charu Aggarwal and Karthik Subbian. Evolutionary Network Analysis: A Survey. *ACM Computing Surveys*, 47(1):10:1–10:36, May 2014.

[BB03] Albert-Laszlo Barabasi and Eric Bonabeau. Scale-Free Networks. *Scientific American*, 288(5):60–69, 2003. Publisher: Scientific American, a division of Nature America, Inc.

[BB20] Michele Benzi and Paola Boito. Matrix functions in network analysis. *GAMM-Mitteilungen*, 43(3):e202000012, 2020.

[BBR13] Michele Benzi, Paola Boito, and Nader Razouk. Decay properties of spectral projectors with applications to electronic structure. *SIAM review*, 55(1):3–64, 2013. Publisher: SIAM.

[BCKS21] Bernhard Beckermann, Alice Cortinovis, Daniel Kressner, and Marcel Schweitzer. Low-Rank Updates of Matrix Functions II: Rational Krylov Methods. *SIAM Journal on Numerical Analysis*, 59(3):1325–1347, 2021. _eprint: https://doi.org/10.1137/20M1362553.

[BK13] Michele Benzi and Christine Klymko. Total communicability as a centrality measure. *Journal of Complex Networks*, 1(2):124–149, December 2013.

[BKS07] Costas Bekas, Effrosyni Kokiopoulou, and Yousef Saad. An estimator for the diagonal of a matrix. *Applied numerical mathematics*, 57(11-12):1214–1229, 2007. Publisher: Elsevier.

[BKS18] Bernhard Beckermann, Daniel Kressner, and Marcel Schweitzer. Low-Rank Updates of Matrix Functions. *SIAM Journal on Matrix Analysis and Applications*, 39(1):539–565, 2018. _eprint: https://doi.org/10.1137/17M1140108.

[BN22] Robert A. Baston and Yuji Nakatsukasa. Stochastic diagonal estimation: probabilistic bounds and an improved algorithm, January 2022. arXiv:2201.10684 [cs, math].

[BR07] Michele Benzi and Nader Razouk. Decay bounds and O(n) algorithms for approximating functions of sparse matrices. *ETNA. Electronic Transactions on Numerical Analysis [electronic only]*, 28, January 2007.

[BVL00] Daniel S. Bernstein and Charles F. Van Loan. Rational Matrix Functions and Rank-1 Updates. *SIAM Journal on Matrix Analysis and Applications*, 22(1):145–154, 2000. _eprint: https://doi.org/10.1137/S0895479898333636.

[CGMM22] Tyler Chen, Anne Greenbaum, Cameron Musco, and Christopher Musco. Error Bounds for Lanczos-Based Matrix Function Approximation. *SIAM Journal on Matrix Analysis and Applications*, 43(2):787–811, June 2022. Publisher: Society for Industrial and Applied Mathematics.

[Che66] E. W. (Elliott Ward) Cheney. *Introduction to approximation theory.* New York, McGraw-Hill Book Co, 1966.

[Che22] Tyler Chen. *Lanczos-based methods for matrix functions.* PhD thesis, University of Washington, September 2022.

[Chu07] Fan Chung. The heat kernel as the pagerank of a graph. *Proceedings of the National Academy of Sciences*, 104(50):19735–19740, December 2007.

[CKM22] Alice Cortinovis, Daniel Kressner, and Stefano Massei. Divide-and-Conquer Methods for Functions of Matrices with Banded

or Hierarchical Low-Rank Structure. *SIAM Journal on Matrix Analysis and Applications*, 43(1):151–177, 2022. _eprint: https://doi.org/10.1137/21M1432594.

[CR13] Ed S. Coakley and Vladimir Rokhlin. A fast divide-and-conquer algorithm for computing the spectra of real symmetric tridiagonal matrices. *Applied and Computational Harmonic Analysis*, 34(3):379–414, May 2013.

[CTU21] Tyler Chen, Thomas Trogdon, and Shashanka Ubaru. Analysis of stochastic Lanczos quadrature for spectrum approximation. In *Proceedings of the 38th International Conference on Machine Learning*, pages 1728–1739. PMLR, July 2021. ISSN: 2640-3498.

[Cup80] J. J. M. Cuppen. A divide and conquer method for the symmetric tridiagonal eigenproblem. *Numerische Mathematik*, 36(2):177–195, June 1980.

[dB05] C. de Boor. Divided Differences, February 2005. arXiv:math/0502036.

[DTT98] Tobin A. Driscoll, Kim-Chuan Toh, and Lloyd N. Trefethen. From Potential Theory to Matrix Iterations in Six Steps. *SIAM Review*, 40(3):547–578, January 1998. Publisher: Society for Industrial and Applied Mathematics.

[EG17] Nicole Eikmeier and David F. Gleich. Revisiting Power-law Distributions in Spectra of Real World Networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 817–826, Halifax NS Canada, August 2017. ACM.

[EH10] Ernesto Estrada and Desmond J. Higham. Network Properties Revealed through Matrix Functions. *SIAM Review*, 52(4):696–714, 2010. _eprint: https://doi.org/10.1137/090761070.

[EH16] Ernesto Estrada and Naomichi Hatano. Communicability Angle and the Spatial Efficiency of Networks. *SIAM Review*, 58(4):692–715, 2016. Publisher: Society for Industrial and Applied Mathematics.

[EKYY13] László Erdős, Antti Knowles, Horng-Tzer Yau, and Jun Yin. Spectral statistics of Erdos-Renyi graphs I: Local semicircle law. *The Annals of Probability*, 41(3B), May 2013. arXiv:1103.1919 [math-ph].

[Est12a] Ernesto Estrada. Complex networks in the Euclidean space of communicability distances. *Physical Review E*, 85(6):066122, June 2012. Publisher: American Physical Society.

[Est12b] Ernesto Estrada. *The structure of complex networks: theory and applications*. Oxford ; New York : Oxford University Press, 2012.

[FS09] Andreas Frommer and Valeria Simoncini. Error Bounds for Lanczos Approximations of Rational Functions of Matrices. In Annie Cuyt, Walter Krämer, Wolfram Luther, and Peter Markstein, editors, *Numerical Validation in Current Hardware Architectures*, pages 203–216, Berlin, Heidelberg, 2009. Springer.

[FS16] Andreas Frommer and Marcel Schweitzer. Error bounds and estimates for Krylov subspace approximations of Stieltjes matrix functions. *BIT Numerical Mathematics*, 56(3):865–892, September 2016.

[FSS16] François Fouss, Marco Saerens, and Masashi Shimbo. *Algorithms and Models for Network Data and Link Analysis*. Cambridge University Press, Cambridge, 2016.

[Gau81] Walter Gautschi. A Survey of Gauss-Christoffel Quadrature Formulae. In P. L. Butzer and F. Fehér, editors, *E. B. Christoffel*, pages 72–147. Birkhäuser Basel, Basel, 1981.

[Gau04] Walter Gautschi. *Orthogonal Polynomials: Computation and Approximation*. Oxford University Press, April 2004.

[GKL20] Stefan Güttel, Daniel Kressner, and Kathryn Lund. Limited-memory polynomial methods for large-scale matrix functions. *GAMM-Mitteilungen*, 43(3):e202000019, 2020.

[GLR05] L. Giraud, J. Langou, and M. Rozloznik. The loss of orthogonality in the Gram-Schmidt orthogonalization process. *Computers & Mathematics with Applications*, 50(7):1069–1075, October 2005.

[GLRE05] Luc Giraud, Julien Langou, Miroslav Rozložník, and Jasper Van Den Eshof. Rounding error analysis of the classical Gram-Schmidt orthogonalization process. *Numerische Mathematik*, 101(1):87–100, July 2005.

[GM09] Gene H. Golub and Gérard Meurant. *Matrices, Moments and Quadrature with Applications*. Princeton University Press, Princeton, UNITED STATES, 2009.

[GVL13] Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU press, 2013.

[Gü13] Stefan Güttel. Rational Krylov approximation of matrix functions: Numerical methods and optimal pole selection. *GAMM-Mitteilungen*, 36(1):8–31, August 2013.

[Ham20] William L. Hamilton. *Graph Representation Learning*. Morgan & Claypool Publishers, September 2020.

[Hig08] Nicholas J Higham. *Functions of matrices: theory and computation*. SIAM, 2008.

[HL97] Marlis Hochbruck and Christian Lubich. On Krylov Subspace Approximations to the Matrix Exponential Operator. *SIAM Journal on Numerical Analysis*, 34(5):1911–1925, October 1997. Publisher: Society for Industrial and Applied Mathematics.

[IK66] Eugene Isaacson and Herbert Bishop Keller. *Analysis of numerical methods*. New York, Wiley, 1966.

[KAB+19] Scott Kolodziej, Mohsen Aznaveh, Matthew Bullock, Jarrett David, Timothy Davis, Matthew Henderson, Yifan Hu, and Read Sandstrom. The SuiteSparse Matrix Collection Website Interface. *Journal of Open Source Software*, 4(35):1244, March 2019.

[Kat53] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, March 1953.

[KDDMT24] Jeremy Kazimer, Manlio De Domenico, Peter J. Mucha, and Dane Taylor. Ranking Edges by Their Impact on the Spectral Complexity of Information Diffusion over Networks. *Multiscale Modeling & Simulation*, 22(3):925–955, September 2024. Publisher: Society for Industrial and Applied Mathematics.

[KKRS21] Peter Kandolf, Antti Koskela, Samuel D. Relton, and Marcel Schweitzer. Computing low-rank approximations of the Fréchet derivative of a matrix function using Krylov subspace methods. *Numerical Linear Algebra with Applications*, 28(6):e2401, December 2021.

[KR93] D. J. Klein and M. Randić. Resistance distance. *Journal of Mathematical Chemistry*, 12(1):81–95, December 1993.

[Kre10] Daniel Kressner. Bivariate matrix functions. *Operators and Matrices*, 2, September 2010.

[Kre19] Daniel Kressner. A Krylov Subspace Method for the Approximation of Bivariate Matrix Functions. In Dario Andrea Bini, Fabio Di Benedetto, Eugene Tyrtyshnikov, and Marc Van Barel, editors, *Structured Matrices in Numerical Linear Algebra*, volume 30, pages 197–214. Springer International Publishing, Cham, 2019. Series Title: Springer INdAM Series.

[KS16] Rasmus Kyng and Sushant Sachdeva. Approximate Gaussian Elimination for Laplacians - Fast, Sparse, and Simple. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 573–582, October 2016. ISSN: 0272-5428.

[Kui06] Arno B. J. Kuijlaars. Convergence Analysis of Krylov Subspace Iterations with Methods from Potential Theory. *SIAM Review*, 48(1):3–40, January 2006. Publisher: Society for Industrial and Applied Mathematics.

[Lan50] C. 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):255–282, 1950.

[LS12] Jörg Liesen and Zdenek Strakos. *Krylov Subspace Methods: Principles and Analysis*. Oxford University Press, October 2012.

[ML16] Naoki Masuda and Renaud Lambiotte. *A guide to temporal networks*. World Scientific, 2016.

[MSN10] Attilio Milanese, Jie Sun, and Takashi Nishikawa. Approximating spectral impact of structural perturbations in large networks. *Physical*

Review E, 81(4):046112, April 2010. Publisher: American Physical Society.

[MT20] Per-Gunnar Martinsson and Joel A. Tropp. Randomized numerical linear algebra: Foundations and algorithms. *Acta Numerica*, 29:403–572, May 2020.

[MT23] Stefano Massei and Francesco Tudisco. Optimizing network robustness via Krylov subspaces, September 2023. arXiv:2303.04971 [cs, math].

[New18] Mark Newman. *Networks*. Oxford University Press, July 2018.

[Par98] Beresford N. Parlett. *The Symmetric Eigenvalue Problem*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, January 1998.

[QDM+18] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network Embedding as Matrix Factorization: Unifying DeepWalk, LINE, PTE, and node2vec. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, pages 459–467, New York, NY, USA, February 2018. Association for Computing Machinery.

[Riv81] Theodore J. Rivlin. *An introduction to the approximation of functions*. Courier Corporation, 1981.

[ROH06] Juan G. Restrepo, Edward Ott, and Brian R. Hunt. Characterizing the Dynamical Importance of Network Nodes and Links. *Physical Review Letters*, 97(9):094102, September 2006. Publisher: American Physical Society.

[Saa92] Y. Saad. Analysis of Some Krylov Subspace Approximations to the Matrix Exponential Operator. *SIAM Journal on Numerical Analysis*, 29(1):209–228, 1992. Publisher: Society for Industrial and Applied Mathematics.

[Saa03] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, January 2003.

[Saa11] Yousef Saad. *Numerical Methods for Large Eigenvalue Problems*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, January 2011.

[Sch16] Marcel Schweitzer. *Restarting and error estimation in polynomial and extended Krylov subspace methods for the approximation of matrix functions*. PhD Thesis, Wuppertal, Univ., Diss., 2016, 2016.

[Sch23] Marcel Schweitzer. Sensitivity of matrix function based network communicability measures: Computational methods and a priori bounds, June 2023. arXiv:2303.01339 [cs, math].

[Spi19] Daniel Spielman. Spectral and algebraic graph theory. *Yale lecture notes, draft of December*, 4:47, 2019.

[Ste90] G. W. Stewart. *Matrix perturbation theory*. Academic Press, Boston, 1990. Open Library ID: OL1873615M.

[Sze39] Gabor Szego. *Orthogonal polynomials*, volume 23. American Mathematical Soc., 1939.

[Tim63] A. F. Timan. *Theory Of Approximation Of Functions Of A Real Variable*. Pergamon Press., 1963.

[TPER+12] Hanghang Tong, B. Aditya Prakash, Tina Eliassi-Rad, Michalis Faloutsos, and Christos Faloutsos. Gelling, and melting, large graphs by edge manipulation. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, CIKM '12, pages 245–254, New York, NY, USA, October 2012. Association for Computing Machinery.

[Tre17] Lloyd Trefethen. Multivariate polynomial approximation in the hypercube. *Proceedings of the American Mathematical Society*, 145(11):4837–4844, November 2017.

[Tre19] Lloyd N Trefethen. *Approximation Theory and Approximation Practice, Extended Edition*. SIAM, 2019.

[UCS17] Shashanka Ubaru, Jie Chen, and Yousef Saad. Fast Estimation of tr(f(A)) via Stochastic Lanczos Quadrature. *SIAM Journal on Matrix*

Analysis and Applications, 38(4):1075–1099, January 2017. Publisher: Society for Industrial and Applied Mathematics.

[vL07] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, December 2007.

[Wes23] Nicholas West. *Krylov methods for low-rank updates of matrix-functions with applications in networks*. MSc in Mathematical Modelling and Scientific Computing, University of Oxford, August 2023.

[Wil65] J. H. (James Hardy) Wilkinson. *The algebraic eigenvalue problem*. Oxford, Clarendon Press, 1965.

[WPKVM14] Xiangrong Wang, Evangelos Pournaras, Robert E. Kooij, and Piet Van Mieghem. Improving robustness of complex networks via the effective graph resistance. *The European Physical Journal B*, 87(9):221, September 2014.

[WSMB21] Sheng Wang, Yuan Sun, Christopher Musco, and Zhifeng Bao. Public Transport Planning: When Transit Network Connectivity Meets Commuting Demand. In *Proceedings of the 2021 International Conference on Management of Data*, SIGMOD '21, pages 1906–1919, New York, NY, USA, June 2021. Association for Computing Machinery.

[Xia12] Shuhuang Xiang. On Error Bounds for Orthogonal Polynomial Expansions and Gauss-Type Quadrature. *SIAM Journal on Numerical Analysis*, 50(3):1240–1263, January 2012. Publisher: Society for Industrial and Applied Mathematics.