Connections Between Lanczos Iteration and Orthogonal Polynomials

Christopher G. Green

A thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science

University of Washington

2001

Program Authorized to Offer Degree: Mathematics

University of Washington
Graduate School

This is to certify that I have examined this copy of a master's thesis by

Christopher G. Green

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

____
Anne Greenbaum

____
Gunther Uhlmann

Date: ______________________

Signature__________________________

Date______________________________

University of Washington

Abstract

Connections Between Lanczos Iteration and Orthogonal Polynomials

by Christopher G. Green

Chair of Supervisory Committee:

Professor Anne Greenbaum
Mathematics

In this thesis we examine the connections between orthogonal polynomials and the Lanczos algorithm for tridiagonalizing a Hermitian matrix. The Lanczos algorithm provides an easy way to calculate and to estimate the eigenvalues and eigenvectors of such a matrix. It also forms the basis of several popular iterative methods for solving linear systems of the form $Ax = b$, where A is an $m \times m$ Hermitian matrix and b is an $m \times 1$ column vector. Iterative methods often provide significant computational savings when solving such systems.

We demonstrate how the Lanczos algorithm gives rise to a three-term recurrence, from which a family of orthogonal polynomials may be derived. We explore two of the more important consequences of this line of thought: the behavior of the Lanczos iteration in the presence of finite-precision arithmetic, and the ability of the Lanczos iteration to compute zeros of orthogonal polynomials. A deep understanding of the former is crucial to actual software implementation of the algorithm, while knowledge of the latter provides an easy and efficient means of constructing quadrature rules for approximating integrals.

TABLE OF CONTENTS

LIST OF ALGORITHMS

ACKNOWLEDGMENTS

First and foremost I must thank my advisor, Anne Greenbaum, without whom this thesis would never have been completed. Her expertise and experience in this field was invaluable. I must also express my gratitude to her for giving me creative freedom with this project and, at the same time, keeping me from straying too far from my professed topic. Finally, I would be remiss if I did not thank her for pouring over numerous lengthy drafts of this document with a watchful eye.

I would like to thank Gunther Uhlmann, who read several drafts of this thesis and attended its defense. I will send you a copy of Mr. Shetty's book when it is done.

Writing a thesis is often a difficult and lonely experience, and I do not think I could have completed it so quickly without the help of those around me. Thus my sincere thanks goes to my friends and my parents for their support throughout this effort. In particular, I owe a great debt to Christine Howard, who helped me through some of the more difficult times (and kept me from giving up), and to SB, who was always willing to listen to me complain.

I would also like to take this opportunity to thank those who influenced me earlier in my career. I thank William Connett and Alan Schwartz of the University of Missouri (St. Louis), for introducing me to the wonderful field of orthogonal polynomials (and for putting up with one arrogant college sophomore). I thank the Mathematics Department at Washington University in St. Louis, for believing in me and encouraging me to pursue advanced studies. In particular, I must thank my former advisor, Steven G. Krantz, for his insights on mathematics and on life and for his candor. I would also like to thank my aunt, Sally Newman, who saw fit to give me the best job

I've ever had, a job that exposed talents I never knew I had.

I would like to express my gratitude to AKB Consolidated for their generous financial support. I wish them continued success in their business.

Finally, thanks to Orbital and Underworld for making the tedious LaTeX'ing of this thesis much easier, to Linus Torvalds for making it possible for me to work at home, and to Don Knuth and Leslie Lamport for making such a wonderful typesetting package.

DEDICATION

This thesis is dedicated to AMG, without whom I would not be the person I am today.

NOTATION

In the sequel, capital letters will be used to denote matrices, and lower case letters will denote vectors. Furthermore,

- A will denote a *Hermitian* square matrix of dimension $m \times m$, and b will denote a given column vector of dimension $m \times 1$. Both A and b will be assumed to have complex entries unless otherwise noted.

- Unless otherwise noted, all vector norms will be the usual Euclidean norm, $\|\cdot\|_2$, and all matrix norms will be the induced spectral norm,

  $$\|A\| = \max_{\|v\|=1} \|Av\| .$$

  We will, on occasion, make use of the Frobenius norm $\|\cdot\|_F$, which is defined by

  $$\|A\|_F = \left\{ \sum_i \sum_j |a_{ij}|^2 \right\}^{1/2} .$$

- The $m \times m$ identity matrix will be denoted I_m; its n-th column, which corresponds to the standard unit basis vector in the n-th direction, will be denoted by e_n.

- The transpose of a matrix will be denoted by a superscript "T", and the conjugate transpose will be denoted by a superscript "*".

- The space of (complex-valued) continuous functions on a set E will be denoted $C(E)$, and the space of (complex-valued) square-integrable functions on E will be denoted $L^2(E)$. We will usually assume that $L^2(E)$ has been endowed with its usual inner product,

$$\langle f(x), g(x) \rangle = \int_E f(x)\overline{g(x)}\, dx.$$

- The rounding unit or machine precision will be denoted ϵ.

- The symbol := will be used to denote a definitional equality.

INTRODUCTION

The Lanczos algorithm, in essence, is a means for constructing an approximate tridiagonalization of a Hermitian matrix. It forms the basis of several popular iterative methods for solving linear systems of the form $Ax = b$, where A is an $m \times m$ Hermitian matrix and b is an $m \times 1$ column vector. In many practical applications (such as solving a discretized differential equation), A is a large (i.e., $m \sim 10,000$) sparse matrix; here iterative methods can often reduce the work required to solve $Ax = b$ to $O(m^2)$ or even $O(m)$ operations, whereas Gaussian elimination can require up to a computationally intractable $O(m^3)$ operations.

The Lanczos algorithm also provides an easy way to compute and to approximate the eigenvalues of Hermitian matrices. From the exact tridiagonal factorization of a Hermitian matrix A we can find its eigenvalues easily. Moreover, the approximate factorizations produced by the Lanczos algorithm yield very good approximations to the true eigenvalues of A, often with far less work. This is especially useful when we are only interested in a few of A's eigenvalues.

The power of the Lanczos algorithm stems from its inherent three-term recurrence, which states that a certain linear combination of three consecutive vectors is identically zero. It is a well-known fact from the theory of orthogonal polynomials that certain three-term recurrences give rise to families of polynomials that are orthogonal with respect to some weight function. Thus, the Lanczos algorithm effectively constructs families of orthogonal polynomials.

This line of thought gives us one method of analyzing the effects of finite-precision arithmetic on the Lanczos iteration. While in exact arithmetic the Lanczos algorithm is guaranteed to converge in at most m steps, this is not necessarily the case when the algorithm is performed on a standard computer. Much effort has been put into the task of quantifying exactly how adversely rounding errors affect the Lanczos algorithm, and, even though there are still several unanswered questions in this area, we have a solid understanding of the general behavior of the algorithm.

Another consequence of the connection between orthogonal polynomials and the Lanczos iteration is that the algorithm provides a computationally stable way to compute the zeros of certain families of orthogonal polynomials. These zeros are important in constructing quadrature rules for approximating integrals.

In this thesis, we will explore these connections. We have attempted to make as few assumptions as possible as to the background of the reader. Basic knowledge of linear algebra and analysis, at the level of an advanced undergraduate course, should be sufficient.

Chapter 1

RUDIMENTS OF ITERATIVE METHODS

We begin with a review of the machinery underlying the Lanczos iteration, namely, Krylov subspaces. Krylov subspaces were originally introduced by Alexei N. Krylov in a 1931 paper [21] as a tool for investigating the characteristic polynomials of matrices of small dimension [2, 26]. As we shall see, they have become a very powerful tool in numerical analysis and approximation theory.

The underlying idea for many iterative methods is to reduce the original problem to a sequence of matrix problems of smaller dimension by projecting it onto lower dimensional Krylov spaces. In order to make this concept more plain, we pause briefly to review a few facts about approximations from subspaces in general.

1.1 Approximations from Subspaces

Throughout the present section, $\mathcal{S}$ will denote an arbitrary subspace of $\mathbb{R}^m$. Its dimension will be denoted by n and its set of basis elements will be written $\{s_1, \ldots, s_n\}$. When explicit mention of the dimension of $\mathcal{S}$ is necessary to avoid confusion, we shall use a superscript (e.g., $\mathcal{S}^n$).

Recall that an **A-invariant** subspace is a subspace satisfying $A\mathcal{S} \subset \mathcal{S}$. It is not hard to show (see [26]) that

1. an A-invariant subspace has a basis of eigenvectors of A; and

2. $A|_{\mathcal{S}}$, the restriction of A to $\mathcal{S}$, is a self-adjoint operator.

Given an $m \times n$ matrix $Q = [q_1| \dots |q_n]$, we may ask whether its column space

$$\mathcal{Q} = \text{span}\,\{q_1, \dots, q_n\} \tag{1.1}$$

is A-invariant. By the definition of A-invariance, we know that if $\mathcal{Q}$ were A-invariant, there would exist constants c_{ij} such that

$$Aq_j = \sum_i q_i c_{ij}, \quad j = 1, \dots, n. \tag{1.2}$$

In light of this, it makes sense to define the **residual matrix** of Q by

$$R(Q) = AQ - QC, \tag{1.3}$$

where $C = [c_{ij}]$ is the $n \times n$ matrix that minimizes $\|R(Q)\|$ (in the least squares sense) [26]. When Q has full rank, the unique solution to this problem is given by $C = (Q^*Q)^{-1}Q^*AQ$ [31]. Clearly, we will have $R(Q) = 0$ iff Q is A-invariant. Furthermore, if y is an eigenvector of C corresponding to the eigenvalue λ, then Qy is an eigenvector for A that also corresponds to λ.

If the columns of Q are orthonormal, then the formula for C reduces to Q^*AQ, which is Hermitian when A is Hermitian. Since $Q^*Q = I$, we recognize C in this special case as the **matrix Rayleigh quotient** of Q.

1.1.1 Rayleigh-Ritz Approximation

Often, the subspace in question is not quite A-invariant. Since an invariant subspace would give us the exact eigenvalues of A, we might suspect that a subspace that just fails to be A-invariant might provide us with good approximations to the eigenvalues of A. This reasoning underlies **Rayleigh-Ritz Approximation**, a very powerful tool for computing approximations to eigenvalues and eigenvectors of matrices.

The full Rayleigh-Ritz procedure may be found in [26]; we summarize it here briefly. Given our arbitrary subspace $\mathcal{S}$, let S denote the matrix whose columns are the basis elements of $\mathcal{S}$. The Rayleigh-Ritz algorithm begins by transforming S to a matrix Q with orthonormal columns. It then forms the matrix $C = Q^*AQ$ defined above, and computes the eigenvalues θ_i and corresponding eigenvectors g_i of C. The corresponding approximate eigenvectors of A are then given by $y_i = Qg_i$. Finally, residual errors $r_i = Ay_i - \theta_i y_i$ are computed as a measure of the accuracy of the approximation.

In this context, the "eigenvalues" θ_i are known as **Ritz values**, and the "eigenvectors" y_i are known as **Ritz vectors**. The pair (θ_i, y_i) is collectively known as a **Ritz pair**.

There is much to be said about the optimality (given no other *a priori* information about A) of the Rayleigh-Ritz approximations (θ_i, y_i) to the true eigenpairs (α_i, z_i) of A, and unfortunately, a thorough treatment of this subject would take us too far astray. We only state a few of the major results here; the reader is encouraged to read the excellent book of Parlett [26] for more details.

One sense in which the Rayleigh-Ritz approximations are optimal is immediate from the description of the algorithm. As the reader may recall, the Rayleigh quotient $C = Q^*AQ$ minimizes the quantity $\|AQ - QZ\|$ (where Q is a fixed orthonormal matrix) over all matrices Z. Thus

$$\min_Z \|AQ - QZ\| = \|AQ - QC\| = \|AQ - QG\Theta G^*\|,$$

where $G = [g_1|\dots|g_n]$ and $\Theta = \operatorname{diag}(\theta_i)$;

$$= \|AY - Y\Theta\|, \tag{1.4}$$

where $Y = QG$ is the matrix of Ritz vectors. Conversely, suppose that S is an $m \times n$ matrix having orthonormal columns that span the same space as the columns of Q. Since both sets of columns span the same n-dimensional space, we know that $S = QU$ for some $n \times n$ unitary matrix U. Therefore, if Δ is any $n \times n$ diagonal matrix, we have

$$\begin{aligned}\|AS - S\Delta\| = \|AQU - QU\Delta\| &= \|AQ - QU\Delta U^*\| \\ &\geq \|AQ - QC\|\end{aligned}$$

by (1.4). Thus we have shown that the quantity $\|AS - S\Delta\|$ is minimized over all pairs (S, Δ), where S and Δ have the form described above, precisely when $S = Y$ and $\Delta = \Theta$ [12].

The Ritz pairs are also optimal approximations in the sense of a minimax problem. The well-known Courant-Fischer theorem gives us a characterization of the eigenvalues of a matrix in terms of a variational problem:

Theorem 1.1 (Courant-Fischer). *Let $\{\alpha_j\}_{j=1}^m$ be the eigenvalues of A. Assume that the eigenvalues have been ordered so that $\alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_m$. Then for each j,*

$$\alpha_j = \min_{\substack{\mathcal{T} \subset \mathbb{R}^m \\ \dim \mathcal{T} = j}} \max_{\substack{t \in \mathcal{T} \\ t \neq 0}} \rho(t), \tag{1.5}$$

where $\rho(x)$ is the Rayleigh quotient of A.

See [18] or [26] for a proof of this theorem.

As demonstrated in [26], it is a direct consequence of the Rayleigh-Ritz construction that the Ritz values satisfy a similar variational characterization, namely

$$\theta_j = \min_{\substack{\mathcal{T} \subset \mathcal{S} \\ \dim \mathcal{T} = j}} \max_{\substack{t \in \mathcal{T} \\ t \neq 0}} \rho(t). \tag{1.6}$$

Finally, the Rayleigh-Ritz approximations from $\mathcal{S}$ are also optimal in the following sense. Let P be the orthogonal projector onto $\mathcal{S}$ and consider PA, the projection of A onto $\mathcal{S}$. It is easy to see that $\mathcal{S}$ is PA-invariant: if $s \in \mathcal{S}$, then $(PA)s = P(As)$ is always a vector in $\mathcal{S}$. Whereas the restriction of A to $\mathcal{S}$ is not an operator from $\mathcal{S}$ into itself (since $\mathcal{S}$ is not A-invariant), the restriction $PA|_{\mathcal{S}}$ of PA to $\mathcal{S}$ is. It can be shown (see [26]) that the Ritz pairs (θ_i, y_i) are the eigenpairs of $PA|_{\mathcal{S}}$. This fact will be of use to us in our discussion of Krylov spaces.

We have now laid a solid foundation for our future discussion of the Lanczos algorithm. Before we can breach this topic, however, a few words must be said about Krylov subspaces, the framework upon which the Lanczos algorithm is built.

1.2 Approximations from Krylov Subspaces

We first define the notion of a Krylov sequence. The **Krylov sequence** associated with A and b is the sequence

$$b, Ab, A^2b, A^3b, \ldots \tag{1.7}$$

The **n-th Krylov subspace** $\mathcal{K}_n(b; A)$ (or simply $\mathcal{K}_n$ when there is no opportunity for confusion) is then the span of the first n vectors of the Krylov sequence:

$$\mathcal{K}_n := \text{span}\left\{b, Ab, A^2b, \ldots, A^{n-1}b\right\}. \tag{1.8}$$

We will also make use of the associated **n-th Krylov matrix** K_n, which is defined as the matrix whose i-th column is the i-th element of the Krylov sequence:

$$K_n = \left[\begin{array}{c|c|c|c} \\ \\ b & Ab & \ldots & A^{n-1}b \\ \\ \\ \end{array}\right].$$

Krylov subspaces have many nice properties which make them the ideal subspace from which to construct approximations to many matrix problems. For instance, notice that we do not actually need to know the matrix A to form $\mathcal{K}_n$, only how to form products of the form Av, where v is a vector.

It is also clear that each vector $x \in \mathcal{K}_n$ can be expressed in the form $p(A)b$, for some polynomial p of degree $\leq n - 1$. Conversely, if p is a polynomial of degree $\leq n - 1$, then $p(A)b$ is an element of $\mathcal{K}_n$. Thus we have the following alternative characterization of $\mathcal{K}_n$:

$$\mathcal{K}_n = \{p(A)b : p \text{ a polynomial of degree } \leq n-1\}\,. \tag{1.9}$$

Perhaps the most important application of Krylov subspaces is the Lanczos algorithm, to which we now turn our attention.

1.3 The Lanczos Iteration

The Lanczos algorithm made its debut in a 1952 paper by Cornelius Lanczos [22]. The Lanczos algorithm constructs an orthonormal basis for the Krylov subspace $\mathcal{K}_n$, and, in the process, reduces A to a tridiagonal matrix via a series of orthogonal similarity transformations. The algorithm is quite simple and can be implemented in a few lines of MATLAB.

There are several approaches to deriving the Lanczos algorithm; each gives a useful insight into the method. First, we can view the Lanczos algorithm as an approximate reduction of A to tridiagonal form. Recall that an arbitrary matrix A can be reduced to upper Hessenberg form (i.e., all entries below the first subdiagonal are zero) by series of Householder transformations [31]. The net effect of these transformations is to construct an orthogonal matrix Q and an upper Hessenberg matrix H such that $A = QHQ^*$.

When A is a Hermitian matrix, the constructed Hessenberg matrix H reduces to a tridiagonal matrix T. Ideally, we would much rather work with the simpler matrix T than with the full matrix A. However, in typical applications the dimension m of A is prohibitively large; we must, instead, content ourselves with an approximate reduction.

Let q_i denote the i-th column of Q and define the $m \times n$ matrix Q_n to be the matrix whose columns are $q_1, q_2, \ldots, q_n$:

$$Q_n = \left[\begin{array}{c|c|c|c} & & & \\ q_1 & q_2 & \ldots & q_n \\ & & & \end{array} \right].$$

Let T_n be the $n \times n$ matrix formed from the first n rows and n columns of T:

$$T_n = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \beta_2 & \ddots & \ddots & \\ & & \ddots & \alpha_{n-1} & \beta_{n-1} \\ & & & \beta_{n-1} & \alpha_n \end{bmatrix}.$$

(In other contexts, T_n is known as a **Jacobi matrix**.)

A straightforward calculation now shows that

$$AQ_n = Q_n T_n + \beta_n q_{n+1} e_n^T. \tag{1.10}$$

By equating the n-th columns of each side of (1.10), we obtain the **Lanczos recurrence**:

$$Aq_n = \beta_{n-1} q_{n-1} + \alpha_n q_n + \beta_n q_{n+1}. \tag{1.11}$$

The Lanczos recurrence expresses q_{n+1} in terms of the previous n columns of Q. This simple fact gives rise to the powerful **Lanczos Iteration**, which allows us to construct the columns of Q iteratively [31]:

```
$\beta_0 = 0, q_0 = 0$
$b =$ arbitrary, $q_1 = b / \|b\|$
for $n = 1$ to MaxIterations do
    $u = Aq_n$
    $\alpha_n = q_n^* u$
    $u = u - \beta_{n-1} q_{n-1} - \alpha_n q_n$
    $\beta_n = \|u\|$
    $q_{n+1} = u / \beta_n$
```

Algorithm 1.1: Lanczos Iteration

The Lanczos recurrence also tells us that the vectors $q_1, \dots q_n$ form an orthonormal basis for $\mathcal{K}_n(b; A)$: since $q_1 = b / \|b\|$, equation (1.11) tells us that $Ab = \alpha_1 \|b\| q_1 + \beta_1 \|b\| q_2$, so $Ab \in \operatorname{span}\{q_1, q_2\}$. Continuing in this fashion, we see that

$$\mathcal{K}_n = \operatorname{span}\{b, Ab, \dots, A^{n-1}b\} \subseteq \operatorname{span}\{q_1, \dots, q_n\}.$$

The reverse containment also follows from similar reasoning (exchanging the roles of the q_i's and the $A^{i-1}b$'s). Therefore, the spans of the two sets are identical. Since both sets have the same dimension, they span the same space, namely, $\mathcal{K}_n$. The orthonormality of the q_i's is clear, since they are the columns of the orthogonal matrix Q.

Stated slightly differently, the above argument tells us that the Lanczos iteration performs a QR-factorization of the associated Krylov matrix K_n without explicitly forming K_n or the upper triangular factor "R" [31].

The astute reader will notice that the Lanczos algorithm can also be thought of as a modified version of the Gram-Schmidt algorithm applied to columns of K_n. The underlying vector space is now the Krylov space $\mathcal{K}_n$, and the resulting orthonormal basis vectors are the columns q_i of the matrix Q_n. This basis is sometimes called the **Lanczos basis** of $\mathcal{K}_n$. In this basis, the orthogonal projection of A onto $\mathcal{K}_n$ is precisely T_n [26, 31].

Yet another characterization of the Lanczos algorithm comes from the Rayleigh-Ritz procedure described in Subsection 1.1.1. If we apply the Rayleigh-Ritz algorithm to the *sequence* of Krylov spaces $\mathcal{K}_1, \mathcal{K}_2, \ldots,$ we will obtain the Lanczos algorithm. While for an arbitrary sequence of subspaces this might be a computationally expensive undertaking, the procedure simplifies dramatically for a sequence of Krylov spaces [26]: for each successive subspace, we already have an orthonormal basis $\{q_1, \ldots, q_{n-1}\}$ of one less dimension, so we need only add one vector q_n. This vector, however, has already been computed during the previous iteration, and needs only to be normalized. The matrix Rayleigh quotient $Q_n^* A Q_n$ at each step is merely the tridiagonal matrix T_n; T_n, however, contains T_{n-1} as its upper $(n-1) \times (n-1)$ submatrix, so only the remaining two new elements must be computed. Finally, the computation of the Ritz pairs (θ_i, y_i) is greatly simplified by the tridiagonal structure of T_n.

When all the computations involved in the algorithm are performed in exact arithmetic, the Lanczos algorithm will converge in at most m steps, since the span of the columns of Q_m has the same dimension as the ambient space $\mathbb{R}^m$. It should be noted, however, that this guarantee is not valid in the context of finite-precision arithmetic. Indeed, the orthogonality condition $Q_m^* Q_m = I_m$ can be completely destroyed by rounding error. Much effort has been put into determining just how adversely finite-precision arithmetic affects the Lanczos algorithm. It turns out that the algorithm still obtains accurate approximations to the eigenpairs of A, but now obtains multiple (though equally accurate) approximations to each eigenpair of A. This matter will be explored in more detail in Chapter 5.

One of the primary uses of the Lanczos algorithm is to compute some of the eigenvalues of A. Since the eigenvalues of A and of T are the same, we use the Lanczos iteration to approximate A by T_n for some choice of n. The eigenvalues of T_n are the Ritz values resulting from applying the Rayleigh-Ritz procedure to $\mathcal{K}_n$, and as we have seen these values give very good approximations to the eigenvalues of A. Under certain conditions (namely, when the constructed $\mathcal{K}_n$ is A-invariant), the

Lanczos algorithm will actually terminate early (i.e., $\beta_j = 0$ for some $j \ll n$). In this case each eigenvalue of the resulting T_j is an eigenvalue of A.

1.4 Conjugate Gradients

Another important application of the Lanczos iteration is the method of **conjugate gradients**. The conjugate gradients algorithm is an iterative method for solving linear systems of the form $Ax = b$, where A is a Hermitian positive-definite matrix. It was introduced in a 1952 paper by Magnus Hestenes and Eduard Stiefel [16]. It is widely used in numerical computations arising from discretized partial differential equations and in finite element analysis, where the structure of the matrices involved often permits $O(m)$ computations of matrix-vector products.

Conjugate gradients, like the Lanczos iteration, finds its answers by iterating over Krylov subspaces. For conjugate gradients, the task at hand is to find an approximate solution to a linear system $Ax = b$ that is optimal in the sense that the **A-norm** of the error is minimized, where the A-inner product and A-norm are defined by

$$\langle x, y\rangle_A := \langle x, Ay\rangle \quad \text{and} \quad \|x\|_A := \sqrt{\langle x, x\rangle_A}.$$

(The fact that the A-norm is actually a norm is a simple consequence of A's positive-definiteness and Hermitianness.)

Thus if x_* is the true solution of the system, then at its n-th step the conjugate gradients algorithm will minimize the quantity $\|e_n\|_A$, where the error e_n at the n-th step equals $x_* - x_n$.

Like the Lanczos algorithm, the conjugate gradients algorithm is amazingly simple to program.

```
$x_0 = 0, r_0 = b, p_0 = r_0$
for $n = 1$ to MaxIterations do
    $\alpha_n = \langle r_{n-1}, r_{n-1} \rangle / \langle p_{n-1}, p_{n-1} \rangle_A$
    $x_n = x_{n-1} + \alpha_n p_{n-1}$
    $r_n = r_{n-1} - \alpha_n A p_{n-1}$
    $\beta_n = \langle r_n, r_n \rangle / \langle r_{n-1}, r_{n-1} \rangle$
    $p_n = r_n + \beta_n p_{n-1}$
```

Algorithm 1.2: Conjugate Gradients

The method of conjugate gradients simultaneously constructs several different bases for the Krylov subspace K_n generated by A and b: the approximate solutions x_n, the residuals r_n, and the "search directions" p_n all satisfy

$$\begin{aligned} K_n &= \operatorname{span}\left\{b, Ab, \ldots, A^{n-1}b\right\} \\ &= \operatorname{span}\left\{x_1, \ldots, x_n\right\} \\ &= \operatorname{span}\left\{p_0, \ldots, p_{n-1}\right\} \\ &= \operatorname{span}\left\{r_0, \ldots, r_{n-1}\right\}. \end{aligned} \tag{1.12}$$

Moreover, the n-th residual r_n is orthogonal to all previous residuals, and the n-th search direction p_n is A-**conjugate** to all previous search directions, i.e., $\langle p_n, p_j \rangle_A = 0$ for $j < n$. The proofs of these facts can be easily established using induction and the conjugate gradients recurrences [31].

From the orthogonality properties of conjugate gradients, we may prove the following theorem [31].

Theorem 1.2 (Convergence of Conjugate Gradients)**.** *Suppose that the conjugate gradients algorithm has not yet converged to a solution. (That is, suppose $r_{n-1} \neq 0$.) Then the n-th conjugate gradients approximation x_n is the unique point of K_n that minimizes the A-norm of the error e_n. Furthermore, conjugate gradients converges monotonically, i.e., $\|e_n\|_A \leq \|e_{n-1}\|_A$, and in at most m steps.*

Proof. From (1.12) it is clear that $x_n \in K_n$. To see that x_n is the unique minimizer of $\|e_n\|_A$, let $e = x^* - x$ for $x \in K_n$ and calculate

$$\begin{aligned}\|e\|_A^2 &= \langle e, Ae\rangle = \langle x^* - x, b - Ax\rangle = \langle x^* - x_n + x_n - x, b - Ax_n + Ax_n - Ax\rangle \\ &= \langle e_n + (x_n - x), r_n + A(x_n - x)\rangle \\ &= \langle e_n, r_n\rangle + \langle r_n, x_n - x\rangle + \langle e_n, A(x_n - x)\rangle + \langle x_n - x, A(x_n - x)\rangle \\ &= \|e_n\|_A^2 + 2\langle r_n, x_n - x\rangle + \|x_n - x\|_A^2 = \|e_n\|_A^2 + \|x_n - x\|_A^2,\end{aligned}$$

where we have used the facts that $r_n = Ae_n$ and that r_n is orthogonal to K_n (by (1.12)). Since A is positive definite, the last expression is clearly minimized iff $x = x_n$.

Weak monotonicity of convergence is a consequence of the obvious inclusion $K_n \subset K_{n+1}$. The bound on the number of iterations needed for convergence follows from the fact that $K_m = \mathbb{R}^m$. □

It should be noted that the bound on the number of iterations needed for convergence does not account for the effects of finite-precision arithmetic. The analysis of the behavior of conjugate gradients in the presence of finite-precision arithmetic is quite complicated; the interested reader should see the text of Greenbaum [14] for a gentle introduction, or the article [13] by the same author for the gory details.

The rate of convergence of conjugate gradients (in exact arithmetic) is directly related to the width and location of the spectrum of A [31]. Conjugate gradients is well-suited to matrices having spectra that are either well-separated from the origin and/or grouped in small clusters. This is due to the following theorem that bounds the relative error e_n/e_0 of the computation by a minimax criterion involving polynomials p of degree $\leq n$ with $p(0) = 1$ [31].

Theorem 1.3. *Let P_n be the set of all polynomials p of degree $\leq n$ with $p(0) = 1$. If e_0 is the initial error in the conjugate gradients computation and e_n is the error at the n-th step, then, if the algorithm has not already converged to a solution, the relative* *error e_n/e_0 satisfies*

$$\frac{\|e_n\|_A}{\|e_0\|_A} \le \inf_{p \in P_n} \max_{\alpha \in \sigma(A)} |p(\alpha)|. \tag{1.13}$$

Loosely speaking, if a matrix has eigenvalues near 0, there is no hope of finding a polynomial that is small at those eigenvalues (because of the condition $p(0) = 1$). Likewise, if a matrix has m eigenvalues that are very far apart, it will be difficult to construct a polynomial of degree $n < m$ that is small at all the eigenvalues. This last point can be clarified further with the following theorem [31].

Theorem 1.4. *Assume the eigenvalues of A are ordered such that $\alpha_1 \le \cdots \le \alpha_n$. Let $\kappa = \alpha_n/\alpha_1$ denote the 2-norm condition number of A. Then the relative error e_n/e_0 satisfies*

$$\frac{\|e_n\|_A}{\|e_0\|_A} \le \frac{2}{\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^n + \left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^{-n}} \le 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^n. \tag{1.14}$$

Proof Sketch. By Theorem 1.3 it is sufficient to find a polynomial p of degree $\le n$ with $p(0) = 1$ whose maximum absolute value on $[\alpha_1, \alpha_n]$ is precisely the middle quantity in the inequality above. Thus we seek a polynomial $p \in P_n$ such that

$$\max_{\alpha \in [\alpha_1, \alpha_n]} |p(\alpha)| = \frac{2}{\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^n + \left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^{-n}}.$$

As we will see in the next chapter, the polynomial that minimizes the expression on the left is precisely the scaled and translated Chebyshev polynomial

$$\frac{T_n\left(\gamma - 2x/(\alpha_n - \alpha_1)\right)}{T_n(\gamma)}, \quad \text{where}\, \gamma := \frac{\alpha_n + \alpha_1}{\alpha_n - \alpha_1} = \frac{\kappa+1}{\kappa-1}. \tag{1.15}$$

The numerator of (1.15) is clearly bounded above by 1 in absolute value for $x \in [\alpha_1, \alpha_n]$. Furthermore, using the three-term recurrence for the Chebyshev polynomials (see the next chapter), we may derive the recurrence

$$T_n(\gamma) = 2\gamma T_{n-1}(\gamma) - T_{n-2}(\gamma), \quad n \ge 2.$$

This is now a linear second-order constant-coefficient recurrence for $T_n(\gamma)$. By the standard substitution $T_n(\gamma) = r^n$, we find that

$$T_n(\gamma) = s\left(\gamma + \sqrt{\gamma^2 - 1}\right)^n + t\left(\gamma - \sqrt{\gamma^2 - 1}\right)^n. \tag{1.16}$$

From the conditions that

$$\begin{aligned} 1 &= T_0(\gamma) = s + t \\ 2\gamma^2 - 1 &= T_2(\gamma) = 2(s+t)\gamma^2 + 2(s-t)\sqrt{\gamma^2 - 1} - (s+t), \end{aligned}$$

we find that $s = t = 1/2$. Manipulating (1.16) gives us the desired result, namely that

$$T_n(\gamma) = \frac{1}{2}\left[\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^n + \left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^{-n}\right].$$

□

Since the fraction in parentheses is asymptotic to $1 - 2/\sqrt{\kappa}$, we see that a "wide" spectrum could cause conjugate gradients to converge slowly. (Although the above theorem is no guarantee of that, as it only gives an *upper* bound on the relative error [12].)

The convergence of conjugate gradients for matrices with "troublesome" spectra as discussed above can sometimes be accelerated through the use of a preconditioner; see [31] for a more detailed discussion of the situations under which this technique is helpful.

Chapter 2

ORTHOGONAL POLYNOMIALS

Orthogonal polynomials play an important role in mathematics and in physics, often as solutions to differential equations or as eigenfunctions of differential operators. In this chapter, we will review the fundamentals of the theory of orthogonal polynomials. We will also examine two of the more frequently encountered families of orthogonal polynomials, the Legendre polynomials and the Chebyshev[1] polynomials.

2.1 General Theory

Given an inner product space V over a field $\mathbb{F}$, we may define a **family of orthogonal polynomials** $\{p_n(x)\}_{n=0}^{\infty}$ by the conditions

$$\deg p_n = n \tag{2.1}$$

$$\langle p_n, x^m \rangle = 0, \quad \text{for } 0 \leq m \leq n-1. \tag{2.2}$$

These conditions, however, only determine the polynomial p_n up to a multiplicative constant. In order to determine p_n uniquely, it is common to impose one of the following additional conditions:

(i). $p_n(1) = 1$;

(ii). $p_n(x)$ is a monic polynomial;

(iii). $\|p_n\| = 1$, where $\|\cdot\| := \langle \cdot, \cdot \rangle^{1/2}$.

[1] The reader should be advised that the spelling of the name "Chebyshev" is not universally agreed upon. Chebyshev was a Russian mathematician, and there is no canonical way of transliterating the Cyrillic alphabet to the Latin one.

One method of computing a family of orthogonal polynomials is to use this definition to set up a system of linear equations for the coefficients of p_n. For example, to compute p_2 we assume that $p_2(x) = ax^2 + bx + c$ for some $a, b, c \in \mathbb{F}$ and use (2.2) to obtain the equations

$$\langle ax^2 + bx + c, 1\rangle = 0 \quad \text{and} \quad \langle ax^2 + bx + c, x\rangle = 0.$$

The third equation comes from our choice of normalization condition.

Another way to construct the polynomials p_n satisfying the above definition is to perform the Gram-Schmidt orthogonalization procedure on the monomial basis

$$1, x, x^2, x^3, \ldots \tag{2.3}$$

Again, this process only defines the polynomials up to a multiplicative constant, and one of the conditions (i)-(iii) must be imposed for uniqueness.

These two procedures, however, are quite tedious and are not the most efficient means of calculating orthogonal polynomials. A better method can be obtained by taking advantage of the fact that the polynomials constructed previous to p_n are already mutually orthogonal. For convenience of presentation, let us assume that the vector space in question is the space $P([-1, 1])$ of all polynomials in one real variable, defined on the interval $[-1, 1]$. The ambient field will be $\mathbb{C}$, the field of complex numbers.

First, notice that a family of orthogonal polynomials (properly normalized) forms a basis for $P([-1, 1])$: each p_n is a finite linear combination of elements from the monomial basis (2.3). Therefore, any polynomial defined on $[-1, 1]$ can be expressed as a finite linear combination of p_n's. In particular, there exist scalars $c_k^{(n)} \in \mathbb{C}$ (depending on n) such that

$$xp_n(x) = \sum_{k=0}^{n+1} c_k^{(n)} p_k(x), \quad n \geq 1. \tag{2.4}$$

The coefficients are easily seen to be given by the formula

$$c_k^{(n)} = \frac{\langle xp_k(x), p_k(x)\rangle}{\langle p_k, p_k\rangle}.$$

Now consider $\langle xp_n(x), p_j(x)\rangle$, for $0 \le j \le n+1$. Since x is a real variable, we have

$$\langle xp_n(x), p_j(x)\rangle = \langle p_n(x), xp_j(x)\rangle.$$

It therefore follows from the condition (2.2) that

$$\langle xp_n(x), p_j(x)\rangle = 0, \quad \text{for } 0 \le j \le n-2.$$

Hence the sum in (2.4) reduces to only three terms

$$xp_n(x) = c_{n+1}^{(n)} p_{n+1}(x) + c_n^{(n)} p_n(x) + c_{n-1}^{(n)} p_{n-1}(x). \tag{2.5}$$

Upon rearranging (2.5), we obtain the three-term recurrence [4]

$$\begin{aligned} p_{n+1}(x) &= \left(\frac{x - c_n^{(n)}}{c_{n+1}^{(n)}}\right) p_n(x) - \left(\frac{c_{n-1}^{(n)}}{c_{n+1}^{(n)}}\right) p_{n-1}(x) \\ &= \left(A_n x + B_n\right) p_n(x) - C_n p_{n-1}(x), \quad n \ge 1. \end{aligned}$$

It is customary to define $p_{-1}(x) \equiv 0$, so that the above recurrence will also hold for $n = 0$.

The coefficients A_n, B_n and C_n are given explicitly by the formulae

$$\begin{aligned} A_n &= \frac{\langle p_{n+1}(x), p_{n+1}(x)\rangle}{\langle xp_n(x), p_{n+1}(x)\rangle} \\ B_n &= -\frac{\langle xp_n(x), p_n(x)\rangle}{\langle xp_n(x), p_{n+1}(x)\rangle} \frac{\langle p_{n+1}(x), p_{n+1}(x)\rangle}{\langle p_n(x), p_n(x)\rangle} \\ C_n &= \frac{\langle xp_n(x), p_{n-1}(x)\rangle}{\langle xp_n(x), p_{n+1}(x)\rangle} \frac{\langle p_{n+1}(x), p_{n+1}(x)\rangle}{\langle p_{n-1}(x), p_{n-1}(x)\rangle}. \end{aligned}$$

From these formulae it is obvious that the coefficients of the recurrence are all real. It is also true that A_n and C_n are positive [30]; the exact proof of this depends on the normalization condition in force.

These formulae, as written, are troublesome, for they involve the unknown polynomial p_{n+1}. Fortunately, this can be remedied; the exact method again depends on the normalization condition used. For example, suppose we assume that the polynomials are monic; then by degree considerations in the recurrence we must have $A_n = 1$ for all n. Similar simplifications will occur for the other coefficients.

We summarize the above results in the following theorem.

Theorem 2.1. *Given a family of orthogonal polynomials $\{p_n(x)\}_{n=0}^{\infty}$ defined on the interval $[-1, 1]$, there exist real constants A_n, B_n and C_n, with A_n and C_n positive, such that*

$$p_{n+1}(x) = (A_n x + B_n)\, p_n(x) - C_n p_{n-1}(x), \quad n \geq 0, \tag{2.6}$$

where we define $p_{-1}(x) \equiv 0$.

The three-term recurrence gives us a very efficient means of constructing the orthogonal polynomials: we only need to compute at most 4 new quantities at each stage, namely $\langle x p_n(x), p_k(x)\rangle$, $k = n-1, n, n+1$, and $\langle p_{n+1}(x), p_{n+1}(x)\rangle$. For certain normalizations the computations simplify even further.

The three-term recurrence is fundamental to the theory of orthogonal polynomials. Many important results in the field rest upon this single identity. As we shall see later in this chapter, the existence of a three-term recurrence among a family of polynomials forces them to be an orthogonal family. We pause now, however, to reinforce what we have discussed to this point with a more detailed look at some well-known families of orthogonal polynomials.

2.2 Examples of Orthogonal Polynomials

2.2.1 The Legendre Polynomials

The Legendre polynomials $P_n(x)$ are a family of orthogonal polynomials defined on $[-1, 1]$ and orthogonal with respect to the standard L^2 inner product. They are typically normalized by the condition $P_n(1) = 1$. With this normalization they satisfy the three-term recurrence[2]

$$P_{n+1}(x) = \left(\frac{2n+1}{n+1}\right) xP_n(x) - \left(\frac{n}{n+1}\right) P_{n-1}(x), \quad n \geq 1 \tag{2.7}$$

From (2.7) we may calculate the first few Legendre polynomials:

$$\begin{array}{ll} P_0(x) = 1 & P_1(x) = x \\ P_2(x) = \frac{3}{2}x^2 - \frac{1}{2} & P_3(x) = \frac{5}{2}x^3 - \frac{3}{2}x \end{array}$$

The Legendre polynomials also satisfy a differential equation [20]

$$(1 - x^2)y'' - 2xy' + n(n+1)y = 0 \tag{2.8}$$

and as such appear frequently in physics and engineering. For example, in the solution of Laplace's equation $\Delta u = 0$ in the unit ball of $\mathbb{R}^3$, the above differential equation governs the longitudinal component of the solution [17]. (That is, if we express the Laplace operator in spherical coordinates (r, θ, ϕ) and use the method of separation of variables to solve Laplace's equation, the resulting equation in the θ variable will resemble (2.8).)

2.2.2 The Chebyshev Polynomials

The Chebyshev polynomials[3] occur frequently in numerical analysis due to the fact that they satisfy a minimax equation on $[-1, 1]$ [26, 27]. They are most commonly defined by the relation

$$T_n(x) = \cos(n \arccos x), \quad -1 \leq x \leq 1, \, n \geq 0. \tag{2.10}$$

[2] This recurrence is sometimes called Bonnet's Recursion.

[3] Technically, these are the Chebyshev polynomials of the *first* kind. There exist Chebyshev polynomials of the *second* kind, given by the formulas

$$U_n(\cos\theta) = \frac{\sin(n+1)\theta}{\sin\theta}, \quad 0 \leq \theta \leq \pi \tag{2.9}$$

but they will not be needed here.

The Chebyshev polynomials are also orthogonal on the interval $[-1,1]$ but with respect to the weight function $w(x) = (1-x^2)^{1/2}$. As one may easily verify using trigonometric identities, the Chebyshev polynomials satisfy the recurrence

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \quad n \geq 1. \tag{2.11}$$

Using this recurrence we may calculate the first few Chebyshev polynomials:

$$\begin{array}{ll} T_0(x) = 1 & T_1(x) = x \\ T_2(x) = 2x^2 - 1 & T_3(x) = 4x^3 - 3x \end{array}$$

Notice that if we make the change of variables $x = \cos\theta$ (which is valid, since $-1 \leq x \leq 1$, then we have $T_n(\cos\theta) = \cos n\theta$ and

$$\begin{aligned} 1 &= T_0(\cos\theta) = 1 \\ \cos\theta &= T_1(\cos\theta) = \cos\theta \\ \cos 2\theta &= T_2(\cos\theta) = 2\cos^2\theta - 1 \\ \cos 3\theta &= T_3(\cos\theta) = 4\cos^3\theta - 3\cos\theta, \end{aligned}$$

which are precisely the standard multiple angle formulas for cosine. Thus we may also compute the Chebyshev polynomials by examining the real part of $e^{in\theta}$.

From the recurrence (2.11) we may also ascertain that the leading coefficient of T_n (when $n \geq 1$) is 2^{n-1}: this is clearly true for $n = 1$, and by induction, we see that the leading coefficient of T_{n+1} is twice that of T_n, so the claim follows. Consequently, the polynomial $2^{1-n}T_n(x)$ is monic. This particular polynomial has some very nice extremal properties, one of which is the following: of all degree n monic polynomials on $[-1,1]$, the monic Chebyshev polynomial $2^{1-n}T_n(x)$ has the smallest extrema. The following theorem makes this statement more precise.

Theorem 2.2. *Let P^n be the set of all monic polynomials of degree n. The monic Chebyshev polynomial $2^{1-n}T_n(x)$ satisfies the following minimax criterion.*

$$\frac{T_n(x)}{2^{n-1}} = \min_{p \in P^n} \max_{[-1,1]} |p(x)|, \tag{2.12}$$

Note that by rescaling we can transfer this property to any closed subinterval of the real line.

Proof of Theorem 2.2. [27] Suppose to the contrary that there exists $p \in P^n$ such that

$$\max_{[-1,1]} |p(x)| < \max_{[-1,1]} \frac{|T_n(x)|}{2^{n-1}}.$$

From the definition of T_n we have

$$\max_{[-1,1]} |T_n(x)| = \max_{[0,2\pi]} |T_n(\cos(\theta))| = \max_{[0,2\pi]} |\cos(n\theta)| = 1.$$

Furthermore, the extrema of $T_n(x)$ are clearly attained at the points

$$\xi_m = \cos(m\frac{\pi}{n}), \quad m = 0, \ldots, n,$$

and at these points we have $T_n(\xi_m) = (-1)^m 2^{n-1}$.

Since $p \neq 2^{1-n}T_n$, the residual $q = 2^{1-n}T_n - p$ is not the zero polynomial. Consider the values of q at the extrema of T_n: let the integers s and t be such that $0 \leq s \leq t \leq n$ and

$$q(\xi_0) = q(\xi_1) = \cdots = q(\xi_{s-1}) = 0$$
$$q(\xi_s) \neq 0$$
$$q(\xi_t) \neq 0$$
$$q(\xi_{t+1}) = \cdots = q(\xi_n) = 0.$$

Since q is not identically 0, we have $s < n$ and $t > 0$.

Suppose now that

$$q(\xi_{s+1}) = \cdots = q(\xi_{s+j-1}) = 0$$

for some j with $q(\xi_{s+j}) \neq 0$, so that q has at least $j-1$ zeros in $[\xi_s, \xi_{s+j}]$. Notice that for any m, if $q(\xi_m) \neq 0$, then $\operatorname{sgn}(q(\xi_m)) = (-1)^m$ by our definition of q. Therefore, if j is even, $q(\xi_s)$ and $q(\xi_{s+j})$ have the same sign, so q has an even number of zeros in $[\xi_s, \xi_{s+j}]$ (where we have counted zeros according to their multiplicity). Likewise, if j is odd, $q(\xi_s)$ and $q(\xi_{s+j})$ are of opposite sign, so q has an odd number of roots in $[\xi_s, \xi_{s+j}]$.

It follows that q has at least $t-s$ zeros in $[\xi_s, \xi_t]$. But by hypothesis, q has s zeros in $[\xi_0, \xi_{s-1}]$ and $n-t$ zeros in $[\xi_{t+1}, \xi_n]$, so q has at least $s+(t-s)+(n-t) = n$ zeros. Since $\deg q = n-1$, $q \equiv 0$, a contradiction. Hence no polynomial in P^n has smaller extrema than $2^{1-n}T_n(x)$. □

2.3 Favard's Theorem

We have seen how a family of orthogonal polynomials gives rise to a three-term recurrence. Perhaps surprisingly, the converse is also true: a three-term recurrence similar to (2.6) gives rise to orthogonal polynomials. This fact is commonly known as Favard's Theorem [3, 7]:

Theorem 2.3 (Favard)**.** *Let $\{p_n(x)\}$ be a family of polynomials defined on $[-1,1]$ and satisfying a three-term recurrence of the form*

$$p_{n+1}(x) = (A_n x + B_n)p_n(x) - C_n p_{n-1}(x), \quad n \geq 1 \tag{2.13}$$

where the coefficients A_n, B_n, and C_n are real with A_n and C_n positive. Then there exists an inner product on $C([-1,1])$ with respect to which $\{p_n\}$ is a family of orthogonal polynomials. Furthermore, this inner product is given by a Riemann-Stieltjes integral of the form

$$\langle f, g\rangle = \int_{-1}^{1} f(x)\overline{g(x)}\, dw(x), \tag{2.14}$$

where $w(x)$ is a nonnegative, increasing, right-continuous function on $[-1,1]$.

In order to simplify the proof of Favard's Theorem we now introduce a few new concepts. Recall that $P([-1,1])$ denotes the vector space of polynomials defined on $[-1,1]$.

Given a sequence $\{\mu_n\}_{n=0}^{\infty}$ of complex numbers, we may define a linear functional $\mathcal{L}$ on $P([-1,1])$ by setting

$$\mathcal{L}(x^n) = \mu_n, \quad n = 0, 1, \ldots \tag{2.15}$$

and extending linearly to the rest of the space. We refer to $\mathcal{L}$ as a **moment functional** [3].

The term "moment" comes from the fact that under certain circumstances, $\mathcal{L}$ can be realized as an integration against a suitable weight function. In such a situation we recognize (2.15) as the n-th moment of this weight function. The following theorem details one set of circumstances under which this is true [3].

Theorem 2.4 (Representation Theorem)**.** *Let $\mathcal{L}$ be a moment functional defined on $P([-1,1])$. Suppose that $\mathcal{L}$ is **positive definite**, i.e., that $\mathcal{L}(p(x)) > 0$ whenever $p(x) \neq 0$ and $p(x) \geq 0$ for all $x \in [-1,1]$. Then there exists a nonnegative, increasing, right-continuous function $w(x)$ such that*

$$\mathcal{L}(p(x)) = \int_{-1}^{1} p(x)\, dw(x) \tag{2.16}$$

for all polynomials $p(x) \in P([-1,1])$.

Proof Sketch. An easy proof of this theorem can be given using basic functional analysis. From the Stone-Weierstrass theorem we know that $P([-1,1])$ is dense in $C([-1,1])$. Therefore (using the Hahn-Banach theorem) we may extend $\mathcal{L}$ to a linear functional $\overline{\mathcal{L}}$ on $C([-1,1])$. We claim that this extended functional is positive.

To see this, note that any nonnegative continuous function f on $[-1,1]$ is the uniform limit of a sequence of polynomials $p_n \in P([-1,1])$, so clearly f is the pointwise limit of the p_n's. Since $f(x_0) \geq 0$ at every $x_0 \in [-1,1]$, there exists an integer N_0 (which depends on x_0) such that $p_n(x_0) \geq 0$ for $n \geq N_0$. By continuity, we know that $p_n(x_0) \geq 0$ in a neighborhood of x_0. Since the interval $[-1,1]$ is compact, we can cover it by finitely many such neighborhoods U_j, in each of which we have

$$p_n(x) \geq 0, \quad x \in U_j, n \geq N_j.$$

Thus whenever $n \geq \max_j N_j$ we have $p_n(x) \geq 0$ for all $x \in [-1,1]$. Hence f is actually a uniform limit of nonnegative polynomials; by our assumption of positive definiteness, $\mathcal{L}$ is nonnegative for such polynomials, so $\mathcal{L}(f)$ is nonnegative.

The Riesz Representation theorem (see [8]) now allows us to conclude that $\overline{\mathcal{L}}$ is given by a Lebesgue integral with respect to a (unique) Radon measure μ. Since $[-1,1]$ is compact and μ is finite on compact sets, however, we know that μ must be finite on all Borel subsets of $[-1,1]$. Therefore, there exists a nonnegative, increasing, right-continuous function $w(x)$ such that $d\mu = dw(x)$. Since all integrands f in question are continuous, the Lebesgue integral agrees with the Riemann-Stieltjes integral, and we have

$$\mathcal{L}(p(x)) = \int_{-1}^{1} p(x)\, dw(x)$$

for all polynomials $p(x) \in P([-1,1])$, as claimed. □

For a proof of Theorem 2.4 that does not use functional analysis, see [3].

We now possess the proper machinery to prove Favard's Theorem.

Proof of Favard's Theorem. The essence of the proof is this: we use (2.13) to define a suitable moment functional, then apply the lemma above to get our result. To wit, define a moment functional $\mathcal{L}$ by setting

$$\mathcal{L}(1) = \frac{C_1}{A_1}$$
$$\mathcal{L}(p_n(x)) = 0, \quad n \geq 1.$$

and extending linearly. We claim that

$$\mathcal{L}(x^k p_n(x)) = 0, \quad 0 \leq k \leq n,\ n \geq 1. \tag{2.17}$$

The $k = 0$ case is clear from our construction of $\mathcal{L}$. Write the given recurrence (2.13) in the form

$$A_n x p_n(x) = p_{n+1}(x) - B_n p_n(x) + C_n p_{n-1}(x). \tag{2.18}$$

If we apply the moment functional $\mathcal{L}$ to both sides, we obtain $\mathcal{L}(xp_n(x)) = 0$ for all $n \geq 1$. Multiplying both sides of (2.18) by x^{k-1} and inducting on k, we see that $\mathcal{L}(x^k p_n(x)) = 0$ holds for n fixed and $0 \leq k < n$. Since we defined $\mathcal{L}(p_n(x))$ to be zero for $n \geq 1$, the claim will hold for any $n \geq 1$.

Next, from (2.13) it follows that

$$A_n x^n p_n(x) = x^{n-1} p_{n+1}(x) - B_n x^{n-1} p_n(x) + C_n x^{n-1} p_{n-1}(x),$$

so from (2.17) we have

$$\mathcal{L}(x^n p_n(x)) = \frac{C_n}{A_n} \mathcal{L}(x^{n-1} p_{n-1}(x)), \quad n \geq 1.$$

Given these observations we readily conclude that $\mathcal{L}(p_m(x) p_n(x)) = 0$ for $m \neq n$, and

$$\mathcal{L}(p_n(x) p_n(x)) = k_n \left(\frac{C_n C_{n-1} \cdots C_1}{A_n A_{n-1} \cdots A_1} \right), \tag{2.19}$$

where k_n is the leading coefficient of p_n. A simple induction shows that k_n is positive (since A_n is), and the A_n's and C_n were positive by hypothesis, so the right hand side of (2.19) is positive. It follows that $\mathcal{L}$ is positive-definite. From Theorem 2.4, we may conclude that there exists a nonnegative increasing right-continuous function $w(x)$ such that

$$\mathcal{L}(f) = \int_{-1}^{1} f(x)\, dw(x).$$

It is clear that $\langle f, g \rangle := \mathcal{L}(f\bar{g})$ will define a valid inner product. The polynomials p_n are orthogonal with respect to this inner product by construction. □

Favard's Theorem will be of great use to us in Chapter 5, where we will investigate the effects of finite-precision arithmetic on the Lanczos algorithm.

Chapter 3

THE LANCZOS ITERATION AND ORTHOGONAL POLYNOMIALS

We now examine the Lanczos recurrence (1.11) presented in Chapter 2 in more detail. Notice that it is a three-term recurrence, similar to those we discussed in Chapter 3. Based upon our discussion of orthogonal polynomials in the previous chapter, the reader might expect that there are some orthogonal polynomials lurking about. This is indeed the case, as we will now demonstrate. We assume here that all computations are performed in exact arithmetic; the case of finite-precision computations will be covered in the next chapter.

3.1 The Lanczos Iteration Generates Orthogonal Polynomials

Let us first rearrange the Lanczos recurrence (1.11) to a more convenient form by moving all terms involving q_{n+1} to one side of the equality:

$$\beta_n q_{n+1} = Aq_n - \alpha_n q_n - \beta_{n-1} q_{n-1}. \tag{3.1}$$

From the definition of Lanczos Algorithm (Algorithm 1.1), we have the following explicit formulae for the coefficients α_n and β_n:

$$\alpha_n = \langle Aq_n - \beta_{n-1} q_{n-1}, q_n \rangle \quad \text{and} \quad \beta_n = \| Aq_n - \alpha_n q_n - \beta_{n-1} q_{n-1} \| .$$

To simplify our analysis, let us replace A by its eigendecomposition $U\Lambda U^*$, where Λ is a diagonal matrix of eigenvalues and U has the corresponding eigenvectors of A as its columns. Define $\hat{q}_n = U^* q_n$; then since U is a unitary matrix (and therefore respects inner products), (3.1) becomes [14]

$$\beta_n \hat{q}_{n+1} = \Lambda \hat{q}_n - \alpha_n \hat{q}_n - \beta_{n-1} \hat{q}_{n-1}, \tag{3.2}$$

with

$$\alpha_n = \langle \Lambda \hat{q}_n - \beta_{n-1} \hat{q}_{n-1}, \hat{q}_n \rangle \quad \text{and} \quad \beta_n = \left\| \Lambda \hat{q}_n - \alpha_n \hat{q}_n - \beta_{n-1} \hat{q}_{n-1} \right\|.$$

From (3.2) we may read off the recurrence satisfied by the j-th component of $\hat{q}_{n+1}$:

$$\beta_n (\hat{q}_{n+1})_j = \lambda_j (\hat{q}_n)_j - \alpha_n (\hat{q}_n)_j - \beta_{n-1} (\hat{q}_{n-1})_j. \tag{3.3}$$

The "seeds" for the Lanczos iteration were the vectors $\hat{q}_0 = 0$ and $\hat{q}_1 = U^*(b/\left\|b\right\|)$ (where b was an arbitrary nonzero vector). It is easy to see that if we repeatedly back-substitute into (3.3) the corresponding recurrences for $(\hat{q}_{n-1})_j$ and $(\hat{q}_{n-2})_j$, we can express each of the Lanczos basis vectors $\hat{q}_n$ in terms of $\hat{q}_0$ and $\hat{q}_1$. (More precisely, the solution $\hat{q}_n$ to the recurrence (3.2) is uniquely determined by $\hat{q}_0$ and $\hat{q}_1$: our recurrence is a variable-coefficient, second-order difference equation, and it is a standard fact that such an equation will have a unique solution for each choice of initial values $\hat{q}_0$ and $\hat{q}_1$.) In doing so, we will discover that $(\hat{q}_{n+1})_j$ is equal to an n-th degree polynomial $\phi_n(x)$ evaluated at the eigenvalue λ_j, times the initial value $(\hat{q}_1)_j$ [14]. We formalize this assertion in the following lemma.

Lemma 3.1. *If $(\hat{q}_{n+1})_j$ is the j-th component of the $(n+1)$-th Lanczos basis vector, then there exists a polynomial $\phi_n(x)$ such that* $\deg \phi_n = n$ *and*

$$(\hat{q}_{n+1})_j = \phi_n(\lambda_j)(\hat{q}_1)_j, \quad n \geq 1, \tag{3.4}$$

where λ_j is the j-th eigenvalue of A and $\hat{q}_1$ is the initial Lanczos basis vector.

Proof. Consider the $n = 1$ case: from (3.3) we have

$$(\hat{q}_2)_j = \frac{1}{\beta_1} \left(\lambda_j (\hat{q}_1)_j - \alpha_1 (\hat{q}_1)_j - \beta_0 (\hat{q}_0)_j \right) = \frac{1}{\beta_1} \left(\lambda_j - \alpha_1 \right) (\hat{q}_1)_j := \phi_1(\lambda_j)(\hat{q}_1)_j.$$

Similarly, for $n = 2$ we have

$$\begin{aligned}(\hat{q}_3)_j = \frac{1}{\beta_2}\left(\lambda_j(\hat{q}_2)_j - \alpha_2(\hat{q}_2)_j - \beta_1(\hat{q}_1)_j\right) = \frac{1}{\beta_2}\left(\frac{1}{\beta_1}\left(\lambda_j - \alpha_2\right)\left(\lambda_j - \alpha_1\right) - \beta_1\right)(\hat{q}_1)_j \\ := \phi_2(\lambda_j)(\hat{q}_1)_j\end{aligned}$$

The general case follows by induction on n:

$$\begin{aligned}(\hat{q}_{n+1})_j &= \frac{1}{\beta_n}\left(\lambda_j(\hat{q}_n)_j - \alpha_n(\hat{q}_n)_j - \beta_{n-1}(\hat{q}_{n-1})_j\right) \\ &= \frac{1}{\beta_n}\left(\left(\lambda_j - \alpha_n\right)\phi_{n-1}(\lambda_j) - \beta_{n-1}\phi_{n-2}(\lambda_j)\right)(\hat{q}_1)_j \\ &:= \phi_n(\lambda_j)(\hat{q}_1)_j.\end{aligned}$$

Since $\deg \phi_{n-1}(x) = n - 1$, we have $\deg \phi_n(x) = n$. □

It now follows immediately that the ϕ_n's satisfy a three-term recurrence: simply substitute (3.4) into (3.3) to obtain

$$\begin{aligned}\phi_{-1}(\lambda_j) &\equiv 0 \\ \phi_0(\lambda_j) &\equiv 1 \\ \beta_n\phi_n(\lambda_j) &= \left(\lambda_j - \alpha_n\right)\phi_{n-1}(\lambda_j) - \beta_{n-1}\phi_{n-2}(\lambda_j), \quad n \geq 1,\end{aligned} \tag{3.5}$$

for $j = 1, \ldots, n$. This recurrence is only valid, however, on the spectrum of A, which is a discrete set. The ϕ_n's are not necessarily orthogonal with respect to the L^2 inner product we used in the previous chapter, since that inner product was defined on an interval of the real line. One might wonder if perhaps there is some other inner product with respect to which the ϕ_n's *are* orthogonal. The answer, as it turns out, is a resounding yes.

Let us define an inner product on the vector space of all polynomials of degree at most $m - 1$ as follows [14].

$$\langle f(x), g(x)\rangle_w = \sum_{j=1}^{m} f(\lambda_j)\overline{g(\lambda_j)}\,|(\hat{q}_1)_j|^2 \tag{3.6}$$

This is clearly an inner product, for

- it is linear in each "slot";
- it is conjugate symmetric (i.e. $\langle f, g\rangle = \overline{\langle g, f\rangle}$); and
- $\langle f, f\rangle > 0$ for all $f \neq 0$ (since $\hat{q}_1 \neq 0$ and $\deg f < m$).

It is with respect to this w-inner product that the ϕ_n's are orthogonal [14].

Theorem 3.1. *The polynomials ϕ_n defined above are orthogonal with respect to the inner product $\langle \cdot, \cdot\rangle_w$ defined in* (3.6)*.*

Proof.

$$\begin{aligned}
\langle \phi_s, \phi_t\rangle_w &= \sum_{j=1}^{m} \phi_s(\lambda_j)\overline{\phi_t(\lambda_j)}|(\hat{q}_1)_j|^2 \\
&= \sum_{j=1}^{m} [\phi_s(\lambda_j)(\hat{q}_1)_j]\left[\overline{\phi_t(\lambda_j)(\hat{q}_1)_j}\right] \\
&= \langle q_{s+1}, q_{t+1}\rangle = \delta_{st},
\end{aligned}$$

since the Lanczos vectors are orthonormal. □

We can also rewrite α_n and β_n in terms of the w-inner product [14]:

$$\begin{aligned}
\alpha_n &= \langle \Lambda\hat{q}_n - \beta_{n-1}\hat{q}_{n-1}, \hat{q}_n\rangle \\
&= \sum_{j=1}^{m} (\Lambda\hat{q}_n - \beta_{n-1}\hat{q}_{n-1})_j \overline{(\hat{q}_n)_j} = \sum_{j=1}^{m} \lambda_j|(\hat{q}_n)_j|^2 - \beta_{n-1}(\hat{q}_{n-1})_j \overline{(\hat{q}_n)_j} \\
&= \sum_{j=1}^{m} \lambda_j [\phi_{n-1}(\lambda_j)(\hat{q}_1)_j]^2 - \beta_{n-1}[\phi_{n-2}(\lambda_j)(\hat{q}_1)_j]\left[\overline{\phi_{n-1}(\lambda_j)(\hat{q}_1)_j}\right] \\
&= \sum_{j=1}^{m} \left[\lambda_j |\phi_{n-1}(\lambda_j)|^2 - \beta_{n-1}\phi_{n-2}(\lambda_i)\overline{\phi_{n-1}(\lambda_j)}\right] |(\hat{q}_1)_j|^2 \\
&= \langle x\phi_{n-1}(x) - \beta_{n-1}\phi_{n-2}(x), \phi_{n-1}(x)\rangle_w. \qquad (3.7)
\end{aligned}$$

Likewise,

$$\begin{aligned}\beta_n &= \|\Lambda\hat{q}_n - \alpha_n\hat{q}_n - \beta_{n-1}\hat{q}_{n-1}\| \\ &= \left[\sum_{j=1}^{m} |\lambda_j(\hat{q}_n)_j - \alpha_n(\hat{q}_n)_j - \beta_{n-1}(\hat{q}_{n-1})_j|^2\right]^{1/2} \\ &= \left[\sum_{j=1}^{m} |\lambda_j\phi_{n-1}(\lambda_j) - \alpha_n\phi_{n-1}(\lambda_j) - \beta_{n-1}\phi_{n-2}(\lambda_j)|^2 \, |(\hat{q}_1)_j|^2\right]^{1/2} \\ &= \|x\phi_{n-1}(x) - \alpha_n\phi_{n-1}(x) - \beta_{n-1}\phi_{n-2}(x)\|_w \,, \qquad (3.8)\end{aligned}$$

where $\|\cdot\|_w = \langle\cdot,\cdot\rangle_w^{1/2}$.

To summarize, we have shown that the Lanczos iteration, in exact arithmetic, produces a family $\{\phi_n\}$ of polynomials that are orthogonal with respect to the w-inner product defined in (3.6). Moreover, these polynomials satisfy the three-term recurrence given below

$$\begin{aligned}\phi_{-1}(x) &\equiv 0 \\ \phi_0(x) &\equiv 1 \\ \beta_n\phi_n(x) &= (x-\alpha_n)\,\phi_{n-1}(x) - \beta_{n-1}\phi_{n-2}(x), \quad n \geq 1. \qquad (3.9)\end{aligned}$$

3.2 Eigenvalues of Jacobi Matrices

Let us now consider the ramifications of these developments upon the Jacobi matrix T_n constructed by the Lanczos algorithm. Recall that T_n is defined by

$$T_n = Q_n^* A Q_n.$$

Its entries are given by $t_{ik} = q_i^* A q_k$. If we once again make use of the eigendecomposition of A, then we have

$$T_n = \hat{Q}_n^* \Lambda \hat{Q}_n,$$

where $\hat{Q}_n = UQ_n$. The entries of T_n are now given by

$$\begin{aligned} t_{ik} = \hat{q}_i^* \Lambda \hat{q}_k &= \sum_{j=1}^{m} \lambda_j \overline{(\hat{q}_i)_j} (\hat{q}_k)_j \\ &= \sum_{j=1}^{m} \lambda_j \overline{\phi_{i-1}(\lambda_j)} \phi_{k-1}(\lambda_j) |(\hat{q}_1)_j|^2 \\ &= \langle x\phi_{k-1}(x), \phi_{i-1}(x) \rangle_w \end{aligned} \tag{3.10}$$

(which agrees with equations (3.7) and (3.8)).

Often, we use the Lanczos algorithm as a stepping stone on our path towards the eigenvalues of A. Clearly, it is sufficient to find the eigenvalues of T, which is usually easier than finding the eigenvalues of A directly since T is tridiagonal. In fact, we know exactly what the eigenvalues of T are: they are the zeros of the w-orthogonal polynomials $\phi_n(x)$.

Theorem 3.2. *The characteristic polynomial of T_n is a multiple of $\phi_n(x)$. More precisely,*

$$\det(xI - T_n) = (\beta_1 \beta_2 \dots \beta_n)\, \phi_n(x). \tag{3.11}$$

Proof. We proceed by induction on n. We have

$$\det(xI - T_n) = \begin{vmatrix} x-\alpha_1 & -\beta_1 & & & \\ -\beta_1 & x-\alpha_2 & -\beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & -\beta_{n-2} & x-\alpha_{n-1} & -\beta_{n-1} \\ & & & -\beta_{n-1} & x-\alpha_n \end{vmatrix}$$

Expand the determinant by minors of the last column; then we have

$$\det(xI - T_n) = \beta_{n-1} \begin{vmatrix} x-\alpha_1 & -\beta_1 & & & & \\ -\beta_1 & x-\alpha_2 & -\beta_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & -\beta_{n-3} & x-\alpha_{n-2} & -\beta_{n-2} \\ & & & 0 & -\beta_{n-1} \end{vmatrix}$$

$$+ (x-\alpha_n)\det(xI - T_{n-1}).$$

Now expand the first determinant by minors of the last row to obtain

$$\det(xI - T_n) = (x-\alpha_n)\det(xI - T_{n-1}) - \beta_{n-1}^2 \det(xI - T_{n-2})$$

which is a three-term recurrence for the n-th degree polynomial $\det(xI - T_n)$. Substitute the inductive hypothesis into the right-hand side to obtain

$$\begin{aligned} \det(xI - T_n) &= (\beta_1 \dots \beta_{n-1})\,(x-\alpha_n)\phi_{n-1}(x) - (\beta_1 \dots \beta_{n-2})\,\beta_{n-1}^2 \phi_{n-2}(x) \\ &= (\beta_1 \dots \beta_{n-1})\,[(x-\alpha_n)\phi_{n-1}(x) - \beta_{n-1}\phi_{n-2}(x)] \\ &= (\beta_1 \dots \beta_{n-1})\,[\beta_n \phi_n(x)]\,, \end{aligned}$$

by the recurrence formula for the ϕ_n's (equation (3.1)). □

Corollary 3.2.1. *The eigenvalues of T_n are precisely the zeros of the ϕ_n's.*

Proof. By Theorem 3.2, the characteristic polynomial of T_n has the same roots as ϕ_n. □

In practice, this equivalence is used in the reverse manner: we may calculate the zeros of orthogonal polynomials by finding the eigenvalues of their associated Jacobi matrices. Given a family of orthogonal polynomials, we can use the coefficients of their three-term recurrence to construct a sequence of tridiagonal matrices T_n, just as we did here. The zeros of the polynomials in this family are simply the eigenvalues of the T_n.

This is an extremely valuable piece of information from the standpoint of numerical analysis: the direct computation of the zeros of polynomials is numerically unstable while the computation of the eigenvalues of an $m \times m$ Hermitian tridiagonal matrix is a well-conditioned problem that can be solved in $O(m^2)$ time.[1] The zeros of orthogonal polynomials are of great importance in numerical integration, as they are the nodes for Gauss-Christoffel quadrature formulas. (A more detailed explanation of this can be found in Appendix A.) The Lanczos algorithm thus provides a computationally stable method of computing the nodes required for Gaussian quadrature.

[1] In fact, Dario Bini and Victor Pan [1] developed a method to solve this problem in $O(m \log m)$ time. More recently, S Eisenstat and Ming Gu [6] have constructed an $O(m \log m)$ algorithm for solving this problem based on the Fast Multipole Method of Greengard and Rokhlin [15, 28].

Chapter 4

THE LANCZOS ALGORITHM IN FINITE-PRECISION ARITHMETIC

Our discussion of the Lanczos algorithm in the previous chapters assumed the use of exact arithmetic. On a computer, however, we often must settle for finite-precision computations, i.e., computations in which quantities are only known to within certain tolerances. It is therefore of interest to know how finite-precision arithmetic affects the Lanczos algorithm. Will the computed basis vectors q_n still be orthogonal? Will the constructed tridiagonal matrices T_n still give accurate approximations to the eigenvalues of A? Will the algorithm even converge, or will it grind on blindly, forever spewing forth meaningless quantities?

The complications introduced by working in inexact arithmetic were known to Lanczos when he presented his algorithm in the early 1950's. It took almost two decades before the first rigorous analysis of the effects of finite-precision arithmetic appeared. The 1971 Ph.D. thesis of Chris Paige [23] was the first major examination of the situation, and is still considered one of the most authoritative treatments of the matter.

A precise understanding of the behavior of the Lanczos algorithm on a computer requires a deeper analysis of the Lanczos iteration. To simplify our analysis, we consider the following implementation of the Lanczos algorithm.

```
Let $q_1$ be a given vector of unit norm.
$u_1 = Aq_1$
for $n = 1$ to MaxIterations do
    $\alpha_n = q_n^* u_n$
    $w_n = u_n - \alpha_n q_n$
    $\beta_{n+1} = +\sqrt{w_n^* w_n}$
    if $\beta_{n+1} = 0$ then
        stop
    $q_{n+1} = w_n / \beta_{n+1}$
    $u_{n+1} = Aq_{n+1} - \beta_{n+1} q_n$
```

Algorithm 4.1: Lanczos Iteration (variant)

It should be clear that Algorithm 4.1 will produce the same results as Algorithm 1.1 in exact arithmetic.

From a cursory glance, the reader might conclude that all hope of a convergent algorithm is lost in finite-precision, for the orthogonality of the generated basis vectors relies upon an inductive calculation, namely, a recurrence. In finite-precision arithmetic, rounding errors will accumulate during the computation of the recurrence and destroy orthogonality. With no guarantee of orthogonality, the answers produced by the Lanczos iteration are seemingly meaningless.

In fact, however, the outlook is far from bleak. As we will see in this chapter, the Lanczos algorithm in inexact arithmetic will still generate Ritz pairs that are very good approximations to the eigenpairs of A despite the loss of orthogonality. Moreover, we will determine precisely when orthogonality is lost, and what effect this loss has upon the eigenpairs of A. We will also see how the Lanczos algorithm still generates orthogonal polynomials, although now the weight function will look slightly different. Finally, we will comment briefly on the eerie phenomenon of “ghost eigenvalues” that is often observed in finite-precision implementations of the Lanczos algorithm.

Our analysis of Lanczos algorithm will require a few simple facts about finite-precision arithmetic. In order to streamline our presentation, we only state them here. A more detailed discussion of these facts can be found in Appendix B. In these two propositions and for the remainder of this chapter, $fl(x)$ will denote the floating point representation of x. Also, we will neglect terms in ϵ^2 and higher, as the effect of these terms upon our analysis is negligible.

Proposition 4.1. *Suppose that $\mathbf{A}$ is an $m \times m$ matrix, $\mathbf{x}$ and $\mathbf{y}$ are $m \times 1$ vectors, and c is a scalar. Let ν denote the maximum number of non-zero elements in any row of $\mathbf{A}$. Finally, assume that the quantities*

$$\sigma := \|\mathbf{A}\| \quad \textit{and} \quad \beta\sigma := \||\mathbf{A}|\|, \tag{4.1}$$

where $|\mathbf{A}| = |a_{ij}|$, are known a priori. *Then the following statements hold in finite-precision arithmetic:*

$$fl\left(fl(\mathbf{x}) - fl(c)\, fl(\mathbf{y})\right) = (\mathbf{x} - c\mathbf{y}) + \delta\mathbf{z}, \quad \textit{where } \|\delta\mathbf{z}\| \leq (\|\mathbf{x}\| + 2|c|\,\|\mathbf{y}\|)\,\epsilon \tag{4.2}$$

$$fl\left(fl(\mathbf{y})^*\, fl(\mathbf{x})\right) = (\mathbf{y} + \delta\mathbf{y})^*\, x \quad \textit{where } \|\delta\mathbf{y}\| \leq m\epsilon\,\|\mathbf{y}\| \tag{4.3}$$

$$fl\left(fl(\mathbf{A})\, fl(\mathbf{x})\right) = (\mathbf{A} + \delta\mathbf{A})\,\mathbf{x}, \quad \textit{where } |\delta\mathbf{A}| \leq \nu\epsilon|\mathbf{A}|. \tag{4.4}$$

Proposition 4.2. *Assume that taking square roots introduces a relative error no greater than ϵ. Then*

$$c = +fl\left(\sqrt{fl(\mathbf{x})^*\, fl(\mathbf{x})}\right) = \left(1 + \frac{1}{2}(m+2)\zeta\right)\|\mathbf{x}\| \tag{4.5}$$

$$y = fl\left(\frac{fl(\mathbf{x})}{c}\right) = \operatorname{diag}\left(1+\zeta\right)\mathbf{x}/c \tag{4.6}$$

$$y^* y = 1 + (m+4)\zeta, \tag{4.7}$$

where $|\zeta| \leq \epsilon$.

We now have all we need to analyze the behavior of Algorithm 4.1 when implemented on a computer.

4.1 Paige's Analysis

Consider the situation at the end of the n-th iteration of the Algorithm 4.1. The algorithm has generated the matrix Q_n of Lanczos basis vectors

$$Q_n = \begin{bmatrix} q_1 & q_2 & \dots & q_{n-1} & q_n \end{bmatrix}$$

and has reduced the original matrix A to the tridiagonal matrix T_n:

$$T_n = \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \beta_3 & \ddots & & \\ & & \ddots & \alpha_{n-1} & \beta_n \\ & & & \beta_n & \alpha_n \end{bmatrix}.$$

Due to rounding errors, however, the recurrence is no longer exact, and must be adjusted by an error term F_n:

$$AQ_n = Q_n T_n + \beta_{n+1} q_{n+1} e_n^T + F_n \tag{4.8}$$

where

$$F_n = \begin{bmatrix} f_1 & f_2 & \dots & f_{n-1} & f_n \end{bmatrix}. \tag{4.9}$$

Moreover, the basis vectors q_j no longer have norm exactly equal to 1; the following proposition quantifies how far the finite-precision q_j's are from having unit norm [24]:

Proposition 4.3. *Assume Algorithm 4.1 has not yet converged (i.e., $\beta_{n+1} \neq 0$). If Algorithm 4.1 is performed in finite-precision arithmetic, then, at the n-th step of the computation, the following estimate will hold for $j = 1, \dots, n$:*

$$|q_{j+1}^* q_{j+1} - 1| \leq (m+4)\,\epsilon. \tag{4.10}$$

Proof. Equation (4.10) follows immediately from (4.7). □

A simple consequence of (4.10) is the inequality

$$1 - \frac{1}{2}(m+4)\epsilon \le fl\left(\|q_j\|\right) \le 1 + \frac{1}{2}\left(m+4\right)\epsilon, \quad j = 1, \ldots, n \tag{4.11}$$

which will be used frequently in the sequel.

4.1.1 Step-By-Step Analysis of the Lanczos Algorithm

Let us now proceed step by step through Algorithm 4.1, constructing estimates on the quantities involved. Our goal is to gain further insight into how the roundoff errors propagate and accumulate through the algorithm. This will ultimately allow us to assess the performance of the Lanczos algorithm on a computer.

The given initial vector q_1 is assumed to have unit norm in exact arithmetic. In the computer, however, it is possible that q_1 cannot be represented exactly. From (4.11), though, we are assured that the finite-precision representation of q_1 satisfies

$$1 - \frac{1}{2}\left(m+4\right)\epsilon \le fl\left(\|q_1\|\right) \le 1 + \frac{1}{2}\left(m+4\right)\epsilon. \tag{4.12}$$

In the second step of the algorithm, we construct $u_1 = Aq_1$. In the computer, however, we actually have $u_1 = fl(fl(A)\,fl(q_1))$. Using (4.4) we have

$$u_1 = fl\left(fl(A)\,fl(q_1)\right) = Aq_1 + \delta u_1, \quad \|\delta u_1\| = \|\delta A q_1\| \le \nu\beta\epsilon\sigma. \tag{4.13}$$

Therefore

$$\|u_1\| \le \|A\|\,fl\left(\|q_1\|\right) + \|\delta u_1\| = \sigma\,fl\left(\|q_1\|\right) + \nu\beta\epsilon\sigma.$$

Using the bound on $\|q_1\|$ we developed in (4.12), we conclude that

$$\|u_1\| \le \sigma\left(1 + \frac{1}{2}\left(m+4\right)\epsilon\right) + \nu\beta\epsilon\sigma = \left[1 + \epsilon\left(\frac{m + 2\nu\beta + 4}{2}\right)\right]\sigma. \tag{4.14}$$

The next steps of the algorithm occur inside a **for** loop. Ultimately, we would like to construct bounds on the quantities inside the loop that are independent of the loop index n. This is quite hard to do all in one go. It is significantly easier to construct bounds that depend on n first, and to then bound all these quantities at once. This is the approach taken by Paige in [24], and it is the approach we take: we will bound all quantities in terms of u_n, and afterwards construct a bound on the u_n's that is independent of n.

Step 4 of Algorithm 4.1 constructs the diagonal element α_n of the matrix T_n. From (4.3) we have

$$\alpha_n = fl\left(fl(q_n)^* \, fl(u_n)\right) = q_n^* u_n - \delta\alpha_n, \tag{4.15}$$

where

$$|\delta\alpha_n| = |\delta q_n^* u_n| \leq \|\delta q_n\| \, \|u_n\| \leq m\epsilon \, \|u_n\| \,, \tag{4.16}$$

since $\|q_n\| \leq (1 + (m+4)/2)\epsilon$ by (4.11). Therefore, using (4.11) once more we have

$$|\alpha_n| \leq \|q_n\| \, \|u_n\| + m\epsilon \, \|u_n\| \leq \left(1 + \frac{3m+4}{2}\epsilon\right) \|u_n\| \,. \tag{4.17}$$

Next we construct $w_n = u_n - \alpha_n q_n$. Its finite-precision representation has the form

$$fl(w_n) = fl\left(fl(u_n) - fl(\alpha_n)\, fl(q_n)\right) = u_n - \alpha_n q_n - \delta w_n, \tag{4.18}$$

where from (4.2) the error term δw_n is bounded in norm by $3 \|u_n\| \epsilon$:

$$\begin{aligned} \|\delta w_n\| &:= \left(\|u_n\| + 2\, \|\alpha_n q_n\|\right) \epsilon \\ &\leq \|u_n\| \, \epsilon + 2 \left[\left(1 + \frac{3m+4}{2}\epsilon\right) \|u_n\|\right] \left[1 + \frac{m+4}{2}\epsilon\right] \epsilon \\ &= 3\, \|u_n\| \, \epsilon. \end{aligned} \tag{4.19}$$

It will be helpful to us shortly to have a bound on $\|w_n\|^2$, so we calculate:

$$\begin{aligned} \|w_n\|^2 &= \|u_n - \alpha_n q_n - \delta w_n\|^2 \\ &= \|u_n\|^2 + \alpha_n^2 \left(\|q_n\|^2 - 2\right) - 2\alpha_n \delta\alpha_n - 2\delta w_n^* \left(u_n - \alpha_n q_n\right) + \|\delta w_n\|^2 \,. \end{aligned} \tag{4.20}$$

Therefore, using (4.10) and equations (4.15)-(4.19) we have

$$\begin{aligned} | \left\|w_n\right\|^2 + \alpha_n^2 - \left\|u_n\right\|^2 | &\leq (m+4) \left\|u_n\right\|^2 \epsilon + 2m \left\|u_n\right\|^2 \epsilon + 6\epsilon \left\|u_n\right\|^2 - 6\epsilon|\alpha_n| \left\|u_n\right\| \left\|q_n\right\| \\ &\leq (3m+10) \left\|u_n\right\|^2 \epsilon. \qquad (4.21) \end{aligned}$$

In step 6 of the algorithm, we set the next off-diagonal element β_{n+1} equal to the norm of w_n. In (4.5) we established that in finite-precision arithmetic,

$$|\beta_{n+1}| = \left[1 + \frac{m+2}{2}\,\epsilon\right] \left\|w_n\right\|.$$

From equation (4.21) above, it follows that

$$\left\|w_n\right\|^2 \leq (1 + (3m+10)\epsilon) \left\|u_n\right\|^2.$$

Combining these two results yields the bound

$$|\beta_{n+1}| \leq (1 + (2m+6)\epsilon) \left\|u_n\right\|. \qquad (4.22)$$

Since we are implicitly assuming the algorithm has not yet converged, the next step to be analyzed in Algorithm 4.1 is step 9. In this step, we construct the next Lanczos basis vector by normalizing w_n. From (4.2) we have

$$fl\left(fl(\beta_{n+1})\, fl(q_{n+1})\right) = w_n + \delta w_n', \qquad (4.23)$$

where

$$\left\|\delta w_n'\right\| \leq \left(|\beta_{n+1}| \left\|q_{n+1}\right\|\right)\epsilon \leq (1+(2m+6)\epsilon) \left\|u_n\right\| \left(1 + \frac{m+4}{2}\,\epsilon\right)\epsilon = \left\|u_n\right\| \epsilon. \qquad (4.24)$$

It remains to bound the final step of Algorithm 4.1. Using (4.2) and (4.4), we have

$$fl(u_n) = Aq_n - \beta_n q_{n-1} + \delta u_n + \delta A q_n, \qquad (4.25)$$

where the norm of the error term is bounded as follows:

$$\begin{aligned} \left\|\delta u_n + \delta A q_n\right\| &\leq \left(\left\|A\right\| \left\|q_n\right\| + 2|\beta_n| \left\|q_{n-1}\right\| + \nu\beta\sigma \left\|q_n\right\|\right)\epsilon \\ &\leq \left[\sigma + 2\left(1 + (2m+6)\epsilon\right) \left\|u_{n-1}\right\| + \nu\beta\sigma\right] \left(1 + \frac{m+4}{2}\,\epsilon\right)\epsilon \\ &= (1+\nu\beta)\,\sigma\epsilon + 2\left\|u_{n-1}\right\| \epsilon. \qquad (4.26) \end{aligned}$$

At this point, let us stop and reflect upon our efforts so far. We have developed estimates for all the rounding errors that are introduced in each step of Algorithm 4.1. Our estimates, however, are in terms of $\|u_n\|$'s, and we would like to rid ourselves of this dependence. Our next goal will thus be to develop a bound on $\|u_n\|$ independent of n that will remedy this situation. In the process, we will take a closer look at how finite-precision affects the orthogonality of the basis vectors.

4.1.2 The Effect of Finite-Precision Arithmetic on Orthogonality

The performance of the Lanczos algorithm depends on the basis vectors q_n maintaining orthogonality to within machine accuracy. As the reader may recall, the Lanczos iteration, in exact arithmetic, constructs a basis $q_1, \ldots, q_n$ for the Krylov space $\mathcal{K}_n(b; A)$. In exact arithmetic, the algorithm was guaranteed to converge in m steps because m mutually orthogonal basis vectors spanned all of $\mathbb{R}^m$. In finite-precision arithmetic, the constructed vectors might not be mutually orthogonal, so convergence is no longer guaranteed to occur in m steps. In this subsection we will examine several of the critical steps in the algorithm, steps where a significant loss of orthogonality could have dire consequences.

During each iteration of the algorithm we construct the basis vector q_{n+1} according to the prescription of (4.23):

$$\beta_{n+1} q_{n+1} = w_n + \delta w_n'.$$

Using (4.18) and (4.25), this reduces to

$$\beta_{n+1} q_{n+1} = (A q_n - \alpha_n q_n - \beta_n q_{n-1}) + (\delta w_n' - \delta w_n - (\delta u_n + \delta A q_n)), \tag{4.27}$$

which is precisely the n-th column of (4.8), suitably rearranged. The error term is bounded easily using (4.24), (4.19), and (4.26):

$$\begin{aligned}\|f_n\| &= \|\delta w_n' - \delta w_n - (\delta u_n + \delta A q_n)\| \\ &\leq \|u_n\| \epsilon + 3 \|u_n\| \epsilon + (1 + \nu\beta) \sigma\epsilon + 2 \|u_{n-1}\| \epsilon \\ &= (1 + \nu\beta) \sigma\epsilon + (4 \|u_n\| + 2 \|u_{n-1}\|) \epsilon. \end{aligned} \tag{4.28}$$

In order to understand the conditions under which a loss of orthogonality between consecutive basis vectors can occur, we would like to estimate the size of $q_n^* q_{n+1}$. Using (4.23), (4.18), and (4.15), we have

$$\beta_{n+1} q_n^* q_{n+1} = \delta\alpha_n - \alpha_n (q_n^* q_n - 1) + q_n^* (\delta w_n' - \delta w_n).$$

Therefore, using the bounds we have developed so far, we have

$$\beta_{n+1} |q_n^* q_{n+1}| \leq 2(m + 4) \|u_n\| \epsilon. \tag{4.29}$$

This bound is not satisfactory, however, as we do not know anything about the size of $\|u_n\|$. Later in this chapter, however, we will develop a bound on the size of $\|u_n\|$ that will allow us to quantify precisely how sensitive the inner product $q_n^* q_{n+1}$ is to rounding error.

What about orthogonality among all the vectors generated so far by the algorithm? The matrix Q_n may no longer be orthogonal due to the roundoff errors that are now involved. We can express how far Q_n is from orthogonal using the decomposition

$$Q_n^* Q_n = R_n^* + \operatorname{diag}\left(q_j^* q_j\right) + R_n, \tag{4.30}$$

where the matrix $R_n = [\rho_{ij}]$ is strictly upper triangular. First, we establish a fact about the structure of R_n.

Proposition 4.4.

$$T_n R_n - R_n T_n = \beta_{n+1} Q_n^* q_{n+1} e_n^T + H_n, \tag{4.31}$$

where $H_n = [\eta_{ij}]$ is an upper triangular matrix.

Proof. Multiply (4.8) on the left by Q_n^* to obtain

$$Q_n^* A Q_n = Q_n^* Q_n T_n + \beta_{n+1} Q_n^* q_{n+1} e_n^T + Q_n^* F_n.$$

The left hand side is Hermitian, so if we take the conjugate transpose of each side and equate the resulting right hand sides, we conclude that

$$Q_n^* Q_n T_n + \beta_{n+1} Q_n^* q_{n+1} e_n^T + Q_n^* F_n = T_n Q_n^* Q_n + \beta_{n+1} e_n q_{n+1}^* Q_n + F_n^* Q_n.$$

Upon regrouping the terms and using the definition of R_n this becomes

$$T_n \left(R_n^* + R_n\right) - \left(R_n^* + R_n\right) T_n = \beta_{n+1} \left(Q_n^* q_{n+1} e_n^T - e_n q_{n+1}^* Q_n\right) + Q_n^* F_n - F_n^* Q_n + \\ \operatorname{diag}\left(q_i^* q_i\right) T_n - T_n \operatorname{diag}\left(q_i^* q_i\right). \quad (4.32)$$

The diagonal elements of each side of this equality are zero, since both sides are of the form $P - P^*$.

To simplify our analysis of (4.32), let us define

$$M_n := T_n R_n - R_n T_n. \quad (4.33)$$

This matrix is clearly upper triangular, and using this definition the left hand side of (4.32) becomes $M_n - M_n^*$. The diagonal entries of M_n can be found by direct calculation of the right hand side of (4.33):

$$\begin{aligned} \mu_{jj} = [M_n]_{jj} &= [T_n R_n]_{jj} - [R_n T_n]_{jj} \\ &= (0, \beta_2 \rho_{12}, \beta_3 \rho_{23}, \ldots, \beta_n \rho_{n-1,n}) - (\beta_2 \rho_{12}, \beta_3 \rho_{23}, \ldots, \beta_n \rho_{n,n-1}) \\ &= \begin{cases} \beta_2 \rho_{12}, & \text{if } j = 1; \\ \beta_j \rho_{j-1,j} - \beta_{j+1} \rho_{j,j+1}, & \text{for } j = 2, \ldots, n-1; \\ \beta_n \rho_{n,n-1}, & \text{if } j = n. \end{cases} \end{aligned}$$

Finally, we claim that

$$M_n = \beta_{n+1} Q_n^* q_{n+1} e_n^* + H_n \quad (4.34)$$

where H_n is an upper triangular matrix. This is easy to see, however, for direct calculation shows that the quantities $Q_n^* F_n - F_n^* Q_n$ and $\operatorname{diag}(q_i^* q_i)\, T_n - T_n \operatorname{diag}(q_i^* q_i)$ can be written as $K_n - K_n^*$ and $N_n - N_n^*$ respectively, where K_n and N_n are strictly upper triangular [24]. Thus $H_n = \operatorname{diag}(M_n) + N_n + K_n$, which establishes that H_n is upper triangular. □

By equating the corresponding entries on each side of (4.33) we see that the entries η_{ij} of H_n are given by the formulae

$$\left.\begin{aligned} \eta_{11} &= -\beta_2 \rho_{12} \\ \eta_{jj} &= -\beta_j \rho_{j-1,j} - \beta_{j+1} \rho_{j,j+1}, \quad \text{for } 2 \le j \le n-1; \\ \eta_{nn} &= \beta_n \rho_{n-1,n} \\ \eta_{j-1,j} &= (q_{j-1}^* f_j - f_{j-1}^* q_j) + \beta_j (q_{j-1}^* q_{j-1} - q_j^* q_j) \\ \eta_{ij} &= q_i^* f_j - f_i^* q_j \quad \text{for all other } i, j. \end{aligned}\right\} \tag{4.35}$$

As it turns out, the structure of H_n will provide us with an easy way to bound u_n.

4.1.3 Bounds on $\|u_n\|$

It remains only to construct a bound on $\|u_n\|$, from which all our desired estimates will follow. Define

$$\mu_n = \max_{1 \le i \le n} \|u_i\| . \tag{4.36}$$

It is a simple task to establish estimates on the size of the entries of H_n in terms of μ_n using the estimates (4.29), (4.28), (4.11), and (4.22):

$$\left.\begin{aligned} |\eta_{11}| &\le |\beta_2||\rho_{12}| = \beta_2|q_1^* q_2| \le 2(m+4)\mu_n\epsilon \\ |\eta_{nn}| &\le |\beta_n||\rho_{n-1,n}| = \beta_n|q_{n-1}^* q_n| \le 2(m+4)\mu_n\epsilon \\ |\eta_{jj}| &\le |\beta_j||\rho_{j-1,j}| + \beta_{j+1}|\rho_{j,j-1}| \le 4(m+4)\mu_n\epsilon \\ |\eta_{ij}| &\le |q_i^* f_j - f_i^* q_j| \le 2\left[(1+\nu\beta)\sigma + 6\mu_n\right]\epsilon \\ |\eta_{j-1,j}| &= |q_{j-1}^* f_j - f_{j-1}^* q_j + \beta_j\left(q_{j-1}^* q_{j-1} - q_j^* q_j\right)| \\ &\le 2\left[(1+\nu\beta)\sigma + (m+10)\mu_n\right]\epsilon. \end{aligned}\right\} \tag{4.37}$$

Using these bounds on the elements of H_n and a few clever tricks, Paige [24] proved the following bound.

Proposition 4.5. *Suppose that*

$$4n\left\{3\,(m+4)\,\epsilon + (7+\nu\beta)\,\epsilon\right\} \ll 1. \tag{4.38}$$

Then the bound

$$\|u_n\| \le \sigma\left\{1 + 2n\left[3\,(m+4)\,\epsilon + (7+\nu\beta)\,\epsilon\right]\right\} \tag{4.39}$$

holds at each step of Algorithm 4.1.

For the proof of this statement, see Appendix C.

Now that we have a bound on u_n, we can complete our analysis of Algorithm 4.1.

4.1.4 Conclusion of Paige's Analysis

Proposition 4.6. *Assume* (4.38)*. Then*

$$\|f_n\| \le \sigma\,(7+\nu\beta)\,\epsilon. \tag{4.40}$$

Proof.

$$\|f_n\| \le (1+\nu\beta)\sigma\epsilon + \left(4\,\|u_n\| + 2\,\|u_{n-1}\|\right)\epsilon \le \sigma(1+\nu\beta)\epsilon + 6\sigma\epsilon = \sigma(7+\nu\beta)\epsilon.$$

□

This gives a bound on the size of the error in the recurrence at each step. It also gives us a bound on the Frobenius norm of the error matrix F_n:

$$\|F_n\|_F = \left\{ \sum_{j=1}^{n} \|f_j\|^2 \} \right\}^{1/2} \leq \sqrt{n}(7 + \nu\beta)\sigma\epsilon. \tag{4.41}$$

We can also finish the analysis we started in equation (4.29) of how finite-precision affects the orthogonality of consecutive Lanczos vectors.

Proposition 4.7. *Assume* (4.38)*. Then*

$$|\beta_{n+1}||q_n^* q_{n+1}| \leq 2\sigma\,(m+4)\,\epsilon. \tag{4.42}$$

Proof.

$$|\beta_{n+1}||q_n^* q_{n+1}| \leq 2(m+4)\;\|u_n\|\,\epsilon \leq 2(m+4)\sigma\epsilon.$$

□

Orthogonality between consecutive basis vectors is hence only lost if β_{n+1} is small, i.e. if there is significant cancellation in step 5 of Algorithm 4.1. (But note that if $\beta_{n+1} = 0$ to machine precision, the algorithm is considered to have converged, and the whole matter is moot[12].)

Finally, we can now establish an easy bound on the size of the matrix H.

Proposition 4.8. *Assume* (4.38)*. Then the elements η_{ij} of the matrix H satisfy*

$$\begin{aligned}
|\eta_{11}| &\leq 2\sigma\,(m+4)\,\epsilon \\
|\eta_{jj}| &\leq 4\sigma\,(m+4)\,\epsilon \quad \textit{for } j = 2,\ldots,n-1; \\
|\eta_{nn}| &\leq 2\sigma\,(m+4)\,\epsilon \\
|\eta_{j-1,j}| &\leq 2\sigma((m+4)\,\epsilon + (7+\nu\beta)\,\epsilon) \quad \textit{for } j = 2,\ldots,n; \\
|\eta_{ij}| &\leq 2\sigma\,(7+\nu\beta)\,\epsilon, \quad \textit{for } i = 1,2,\ldots,j-2.
\end{aligned} \tag{4.43}$$

Proof. This follows immediately from (4.37) and (4.39). □

From this proposition we may derive the following bound on the Frobenius norm of H_n:

$$\|H_n\|_F \leq \sigma\epsilon \left\{8(n-1)(m+4) + n(n-1)(7+\nu\beta)\right\}. \tag{4.44}$$

This will be useful to us in our analysis of the behavior of the Ritz vectors in finite-precision arithmetic.

4.2 Effects of Finite-Precision on the Ritz Vectors

We can perform a similar analysis on the Ritz vectors generated by the Lanczos algorithm at the n-th step. Let us denote the eigenvalue estimates and Ritz vectors at the n-th step by $\theta_j^{(n)}$ and $y_j^{(n)}$, respectively. The eigenvectors of T_n will be denoted $s_j^{(n)}$, so that $y_j^{(n)} = Q_n s_j^{(n)}$. In order to simplify the analysis, we will assume that the eigenvalues $\theta_j^{(n)}$ and eigenvectors $s_j^{(n)}$ of T_n are exact, so that the eigendecomposition

$$T_n = S_n \Theta_n S_n^*, \quad S_n = [s_1^{(n)}|\dots|s_n^{(n)}], \quad \Theta_n = \mathrm{diag}(\theta_1^{(n)}, \dots, \theta_n^{(n)}) \tag{4.45}$$

is exact.

Let us first consider the effect of finite-precision arithmetic on the Ritz vectors $y_j^{(n)}$. The following proposition, which describes the behavior of the Ritz vectors, appears in Paige's thesis [23]; our treatment is based upon that of Parlett [26].

Proposition 4.9. *The Ritz vectors $y_j^{(n)}$ satisfy*

$$\left(y_j^{(n)}\right)^* q_{n+1} = \frac{(s_j^{(n)})^* H_n s_j^{(n)}}{\beta_{n+1} s_{jn}}, \tag{4.46}$$

where s_{jn} denotes the n-th component of $s_j^{(n)}$.

Proof. Multiply equation (4.34) on the left by $(s_j^{(n)})^*$ and on the right by $s_j^{(n)}$ to obtain the identity

$$(s_j^{(n)})^* \left(\beta_{n+1} Q_n^* q_{n+1} e^T\right) s_j^{(n)} = (s_j^{(n)})^* H_n s_j^{(n)} - (s_j^{(n)})^* M_n s_j^{(n)}. \tag{4.47}$$

The left hand side equals

$$\left((s_j^{(n)})^* Q_n^*\right)(q_{n+1}\beta_n)\left(e^T s_j^{(n)}\right) = (y_j^{(n)})^* q_{n+1}\beta_n s_{jn} \tag{4.48}$$

and the $(s_j^{(n)})^* M_n s_j^{(n)}$ term on the right hand side vanishes, since

$$(s_j^{(n)})^* M_n s_j^{(n)} = (s_j^{(n)})^* \left(T_n R_n - R_n T_n\right) s_j^{(n)} = 0.$$

Thus we conclude

$$\left(y_j^{(n)}\right)^* q_{n+1} = \frac{(s_j^{(n)})^* H_n s_j^{(n)}}{\beta_{n+1} s_{jn}},$$

as desired. □

Proposition 4.9 relates the orthogonality of the Ritz vectors $y_j^{(n)}$ that we have already computed to the next Lanczos basis vector. Let us now consider what happens as a Ritz vector begins to converge. From the Lanczos recurrence (1.10) we have

$$AQ_n = Q_n T_n + \beta_{n+1} q_{n+1} e_n^T.$$

Multiply both sides of this equation by the eigenvector $s_j^{(n)}$ to obtain

$$Ay_j^{(n)} = y_j^{(n)}\theta_j + \beta_{n+1} q_{n+1} e_n^T s_j^{(n)}.$$

Hence we have

$$\left\|Ay_j^{(n)} - y_j^{(n)}\theta_j\right\| = |\beta_{n+1}||e_n^T s_j^{(n)}|, \tag{4.49}$$

since q_{n+1} has unit norm. The left hand side of this equation measures how well the Ritz vector $y_j^{(n)}$ approximates an eigenvector of A. In particular, when $y_j^{(n)}$ has converged (to within machine accuracy) to an eigenvector of A, we see that the right hand side of (4.49) is small.

Combining this fact with Proposition 4.9, it is clear that q_{n+1} is not orthogonal to the converged Ritz vector. Thus, the Lanczos basis vectors lose their orthogonality in the direction of converged Ritz vectors [13, 23, 25]. Furthermore, it is only when a Ritz vector begins to converge that the basis vectors lose their orthogonality.

4.3 What about the orthogonal polynomials?

As we showed in Chapter 3, the Lanczos iteration, in exact arithmetic, generates orthogonal polynomials for a set of weights on the eigenvalues of A. These polynomials satisfy a three-term recurrence and are orthogonal with respect to the w-inner product defined in (3.6). Furthermore, the weights are the squared components of the initial vector q_1 in the direction of each eigenvector of A[14].

Let us suppose now that we perform the calculation in finite-precision arithmetic. Assume once again that the Lanczos recurrence can be expressed as

$$\beta_n q_{n+1} = Aq_n - \alpha_n q_n - \beta_{n-1} q_{n-1} + f_n, \tag{4.50}$$

where f_n represents the error incurred by using finite-precision arithmetic. Let $A = U\Lambda U^*$ be the eigendecomposition of A as before. By an analysis identical to that performed in Chapter 4, we may show that (4.50) gives rise to a three-term recurrence

$$\beta_n \phi_n(z) = z\phi_{n-1}(z) - \alpha_n \phi_{n-1}(z) - \beta_{n-1}\phi_{n-2}(z) - \xi_n(z), \tag{4.51}$$

where the ϕ_n's are polynomials of degree n and ξ_n is a function (not necessarily polynomial) satisfying $\xi_n(\lambda_j)\hat{q}_{j1} = U^* f_{ij}$ [14]. The coefficients α_n and β_n are given explicitly by the formulae

$$\begin{aligned} \alpha_n &= \langle z\phi_{n-1}(z) - \beta_{n-1}\phi_{n-2}(z), \phi_{n-1}(z)\rangle_w \\ \beta_n &= \left\| z\phi_{n-1}(z) - \alpha_n\phi_{n-1}(z) - \beta_{n-1}\phi_{n-2}(z) \right\|_w . \end{aligned}$$

From these formulae it is clear that β_n is nonnegative for every n. Thus, by Favard's Theorem, these are the recurrence coefficients for a family of polynomials ψ_n that are orthogonal with respect to some weight function $\tilde{w}(x)$ [14]. This weight function, however, is not necessarily related to our earlier weight $w(x)$, and the ψ_n's may not be w-orthogonal.

In [13] Greenbaum showed that the weight function $\tilde{w}(x)$ resembles a "smeared-out" version of our original weight. This statement will be made more precise in the next section.

4.4 Greenbaum's Analysis

Another significant contributor to our understanding of how the Lanczos algorithm behaves in inexact arithmetic was Anne Greenbaum. In [13] she demonstrated that the finite-precision version of the Lanczos algorithm applied to A generates the same tridiagonal matrices T_n as the *exact* algorithm applied to a larger matrix $\tilde{A}$. The eigenvalues of this larger matrix are distributed in tiny intervals about the true eigenvalues of A, and may be more numerous than those of A [13].

One implication of this is that the finite-precision Lanczos algorithm generates polynomials that are orthogonal with respect to weights on the eigenvalues of the larger matrix $\tilde{A}$ [13]. Since the true eigenvalues of A lie near those of $\tilde{A}$, the "true" weight function corresponding to the eigenvalues of A appears to have been "smeared" over tiny intervals by the rounding errors.

In [13] Greenbaum established bounds on the size of the intervals in terms of the machine precision ϵ and the loop index n. To date, it is not known if these are the best possible bounds. While it is (highly) improbable that a bound independent of ϵ exists, Greenbaum suggested that an (interesting) bound independent of n might exist [14, 13]. What this bound is, however, remains an unanswered question.

4.5 Ghost Eigenvalues

Finally, we briefly discuss the often observed phenomenon of "ghost" eigenvalues. These are extra Ritz values approximating an eigenvalue of A to which a Ritz approximation has already been found. These values are not indications of the multiplicity of the true eigenvalue, as one might think; rather, they are nothing more than artifacts of our inexact implementation [14, 31].

It is possible to provide a rigorous explanation of this phenomenon; however, as this explanation is rather involved and hard-to-follow, we summarize some of its more salient points here.

In the previous section we described how the finite-precision version of the Lanczos algorithm constructs a family of polynomials that are orthogonal with respect to weights that live near, but not necessarily on, the eigenvalues of A. It is a fact of orthogonal polynomial theory (see [30]) that the zeros of these orthogonal polynomials interlace the points of increase of the weight function (the points on which the weight function is defined). Whereas in the exact case the associated orthogonal polynomials can have at most two roots near an eigenvalue of A (one on either side), the associated polynomials in the inexact case can have multiple roots near a true eigenvalue [12]. Since the computed eigenvalues are the roots of the associated polynomials, we see that multiple approximations to the same eigenvalue of A can and will occur if the algorithm is allowed to run long enough.

There are methods of preventing the appearance of "ghost" eigenvalues, but they are not without cost. One easy way to do this is to modify Algorithm 4.1 to save all the computed Lanczos basis vectors instead of overwriting them at each step. New basis vectors are then explicitly reorthogonalized against all the previous vectors. A similar modification saves the converged Ritz vectors instead of the basis vectors. While each of these methods will "ward off ghosts", they require more work and more storage. Similar statements are true for other "ghostbusting" methods. If storage requirements are critical (but processing time is not), the simplest solution to this problem is just to ignore the superfluous solutions.

CONCLUSION

We have seen how the Lanczos algorithm is intimately related to the theory of orthogonal polynomials. The tridiagonal matrices generated by applying the Lanczos algorithm to a Hermitian matrix have a family of orthogonal polynomials as their characteristic polynomials. Conversely, a family of orthogonal polynomials gives rise to a family of tridiagonal matrices which have those polynomials as their characteristic polynomials. This provides a useful connection between the zeros of the polynomials and the eigenvalues of the matrices.

We have also seen how orthogonal polynomials provide an easy explanation for the observed behavior of the Lanczos algorithm in finite-precision arithmetic. The Lanczos algorithm, in inexact arithmetic, produces polynomials orthogonal with respect to a weight defined near the eigenvalues of A. This concept also provides a simple explanation for the observed phenomenon of ghost eigenvalues.

There are many more consequences of the connection between the Lanczos algorithm and orthogonal polynomials, and, unfortunately, we can only cover so much here. Many things were omitted from this thesis due to a lack of time. In these final paragraphs we shall mention some of the other work that has been done in this area.

We unfortunately did not have time to construct numerical examples of the phenomena we discussed in Chapter 4. The interested reader can see [13], [14], or [31] for examples, or can conduct their own experiments (using MATLAB, for instance).

The finite-precision analysis we presented here has also been carried out for the conjugate gradients algorithm. The paper by Greenbaum [13] is the most authoritative treatment of the subject.

Golub and Strakos [10] have explored connections between the Lanczos algorithm, conjugate gradients, and mechanical quadrature. In their paper they explore a means of estimating quadratic forms via Gauss quadrature and the Lanczos algorithm. They then use their method to investigate the convergence of conjugate gradients in finite-precision arithmetic.

Appendix A

A BRIEF PRIMER OF MECHANICAL QUADRATURE

Mechanical quadrature, the process of computing integrals numerically, has a long and interesting history. One would not think that such a seemingly simple task as calculating the value of an integral would attract some of the greatest minds of the nineteenth- and twentieth-centuries. In this chapter, we will explore the development of quadrature rules and their intimate connections to orthogonal polynomials.

A.1 Newton-Cotes Quadrature

As do many other branches of mathematics, the history of mechanical quadrature begins with Newton. Sir Isaac Newton, the renowned mathematician, physicist, and philosopher, was the first to devise a general method for calculating approximate values of integrals. Roger Cotes, who, independently of Newton, developed methods similar to Newton's, refined these ideas into a workable theory of approximate integration.

In the last quarter of the seventeenth-century, Newton devised a method of interpolating a function at a given set $\{\xi_j : j = 1, \ldots, n\}$ of distinct points by a polynomial. He originally derived a formula for his interpolating polynomial in terms of divided differences; we will, however, employ the more modern tool of Lagrange interpolating polynomials.

The **Lagrange interpolating polynomial** $l_j(x)$ of degree $n-1$ is defined by the equation

$$l_j(x) = \frac{(x-\xi_1)\cdots(x-\xi_{j-1})(x-\xi_{j+1})\cdots(x-\xi_n)}{(\xi_j-\xi_1)\cdots(\xi_j-\xi_{j-1})(\xi_j-\xi_{j+1})\cdots(\xi_j-\xi_n)}. \tag{A.1}$$

The Lagrange interpolant is more succinctly expressed in the form

$$l_j(x) = \frac{\omega_n(x)}{\omega_n'(\xi_j)(x-\xi_j)}, \tag{A.2}$$

where

$$\omega_n(x) = \prod_{j=1}^{n}(x-\xi_j) \tag{A.3}$$

is the **node polynomial** of degree n.

When expressed in terms of Lagrange interpolants, Newton's interpolating polynomial becomes

$$p_{n-1}(f;x) = \sum_{j=1}^{n} l_j(x)f(\xi_j). \tag{A.4}$$

Newton then writes

$$f(x) = p_{n-1}(f;x) + r_n(f;x), \tag{A.5}$$

where $r_n(f;x)$ denotes the error in the interpolation. Since the Lagrange interpolant is unique (it is the only polynomial of degree $n-1$ that satisfies $l_j(\xi_i) = \delta_{ij}$), we know that $r_n(f;x) \equiv 0$ for all polynomials f of degree at most $n-1$. Newton integrates (A.5) over a nondegenerate finite interval $[a,b]$ to obtain the **n-point quadrature formula**

$$I(f) := \int_a^b f(x)\,dx = Q_n(f) + R_n(f), \tag{A.6}$$

where

$$Q_n(f) = \int_a^b p_{n-1}(f;x)\,dx = \sum_{j=1}^{n}\lambda_j f(\xi_j) \tag{A.7}$$

is the **quadrature sum**,

$$\lambda_j = I(l_j) = \int_a^b l_j(x)\,dx \tag{A.8}$$

are the **weights** of the quadrature formula, and

$$R_n(f) = I(r_n(f;x)) = \int_a^b r_n(f;x)\,dx \tag{A.9}$$

is the **remainder** or error in the approximation. The points ξ_j are called the **nodes** of the quadrature formula.

By construction, R_n vanishes for all polynomials of degree $n-1$ or less. This is often expressed by saying that Q_n has **degree of exactness** $n-1$, and we write (after Radau) $d(Q_n) = n-1$. (It follows that $d(Q_n) = k$ for any integer k with $0 \leq k < n$.) The quadrature rule Q_n is also called **interpolatory**, since it is obtained by interpolation of n points. It is clear that Q_n is interpolatory iff it has degree of exactness $n-1$.

Roger Cotes, who derived similar expressions for approximate integrals independently of Newton, computed the weights λ_j for quadrature rules with $n \leq 11$ and equally spaced nodes. The λ_j are often called **Cotes numbers** in his honor.

One way of calculating the Cotes numbers is to observe that since $R_n \equiv 0$ for all polynomials of degree $n-1$ or less, we have the n equations

$$\sum_{j=1}^{n} \lambda_j \tau_j^k = \int_a^b x^k\,dx, \quad k = 0, 1, \ldots, n-1. \tag{A.10}$$

When written in matrix form, this system becomes

$$\begin{bmatrix} 1 & 1 & \ldots & 1 \\ \tau_1 & \tau_2 & \ldots & \tau_n \\ \tau_1^2 & \tau_2^2 & \ldots & \tau_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \tau_1^{n-1} & \tau_2^{n-1} & \ldots & \tau_n^{n-1} \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \vdots \\ \lambda_n \end{bmatrix} = \begin{bmatrix} b-a \\ \int_a^b x\,dx \\ \int_a^b x^2\,dx \\ \vdots \\ \int_a^b x^{n-1}\,dx \end{bmatrix}. \tag{A.11}$$

The Cotes numbers can thus be obtained by solving this system [5]. This method, however, is not the best method for obtaining the Cotes numbers; better methods will be discussed shortly.

The formula (A.6) is known today as the **Newton-Cotes** quadrature rule. Well-known special cases of it include the trapezoid rule ($n = 2$) and Simpson's rule ($n = 4$). The Newton-Cotes quadrature served as the cornerstone upon which Gauss, Jacobi, and many others would build the theory of mechanical quadrature.

A.2 Gauss-Jacobi Quadrature

As we discussed above, an arbitrary n-point Newton-Cotes quadrature Q_n has degree of exactness $n-1$. Can we, however, do any better? Can we choose the nodes ξ_j in such a fashion as to achieve a rule for which $d(Q_n) > n-1$? If so, how much better can we do? And how should we choose the nodes?

For starters, it is easy to see that since we have exactly $2n$ unknowns (nodes ξ_j and weights λ_j), we will need $2n$ conditions on the unknowns in general. These $2n$ conditions can be found by requiring the rule to be exact for polynomials of degrees $0, \ldots, 2n-1$, i.e., by requiring that $d(Q_n) = 2n-1$. Furthermore, we see that $2n-1$ is the maximum degree of exactness that we can require, as higher degrees will result in an overdetermined system that may not possess a solution.

It was Carl Friedrich Gauss who first pondered the question of how to choose the nodes optimally, and it was also Gauss who solved the problem. His nineteenth-century solution is not at all obvious—he uses continued fractions for ratios of hypergeometric functions, a tool that he also developed. A few years later, Jacobi would provide a more lucid proof.

Gauss began by examining a “generating function” for the remainders of monomials of the form

$$\sum_{k=0}^{\infty} \frac{R_n(x^k)}{z^{k+1}}.$$

He then manipulates this formal sum to obtain a closed-form formula for the sum:

$$\begin{aligned}\sum_{k=0}^{\infty} \frac{R_n(x^k)}{z^{k+1}} &= \sum_{k=0}^{\infty} \frac{1}{z^{k+1}} \left(I(x^k) - Q_n(x^k)\right)\\ &= \sum_{k=0}^{\infty} \frac{1}{z^{k+1}} \left(I(x^k) - \sum_{j=1}^{n} \lambda_j \xi_j^k\right)\\ &= I\left(\frac{1}{z}\sum_{k=0}^{\infty} \frac{x^k}{z^k}\right) - \sum_{k=0}^{\infty}\sum_{j=1}^{n} \lambda_j \frac{\xi_j^k}{z^{k+1}}\\ &= I\left(\frac{1}{z-x}\right) - \sum_{j=1}^{n} \lambda_j \frac{1}{z-\xi_j}\\ &= R_n\left(\frac{1}{z-x}\right).\end{aligned}$$

Under this interpretation, Q_n has degree of exactness $2n-1$ iff

$$R_n\left(\frac{1}{z-x}\right) = O\left(\frac{1}{z^{2n+1}}\right) \tag{A.12}$$

as $z \to \infty$.

The integral $\int_a^b dx/(z-x)$ was already familiar to Gauss in the context of continued fractions. By a suitable change of variables we may assume that the domain of integration is $[-1,1]$. An elementary calculation shows that

$$\int_{-1}^{1} \frac{1}{z-x}\, dx = \log \frac{1+1/z}{1-1/z}. \tag{A.13}$$

The right-hand side of this equation has a continued fraction expansion

$$\log \frac{1+1/z}{1-1/z} = \cfrac{2}{z - \cfrac{1/3}{z - \cfrac{3/5}{z - \cdots}}}. \tag{A.14}$$

This expansion comes from Gauss' more general expansion for the quotient of two hypergeometric functions [9].

Gauss considers the n-th convergent $R_{n-1,n}$ of the continued fraction (A.14), which is easily seen to be a rational function having a numerator N_{n-1} of degree $n-1$ and a denominator D_n of degree n. (These polynomials, it will turn out, are the Legendre polynomials of the first and second kind, respectively.)

By expanding $R_{n-1,n}$ in terms of powers of $1/z$, he shows that

$$I_n\left(\frac{1}{z-x}\right) = R_{n-1,n}(z) + O\left(\frac{1}{z^{2n+1}}\right) \tag{A.15}$$

for large z. He then decomposes the n-th convergent into partial fractions with the nodes ξ_j as poles and the weights λ_j as residues. We thus have the quadrature expression

$$Q_n\left(\frac{1}{z-x}\right) := R_{n-1,n}(z) = \sum_{j=1}^{n} \frac{\lambda_j}{z-\xi_j}. \tag{A.16}$$

Upon putting (A.15) and (A.16) together we obtain the desired asymptotic growth (A.12), completing the proof.

While Gauss's proof would later inspire Christoffel's generalization to weighted integrals (and Stieltjes measures), it was not, in the eyes of Carl Gustav Jacob Jacobi, the simplest proof. Jacobi rederived Gauss's result in a much clearer fashion, using arguments based upon orthogonal polynomials (though the notion of "orthogonal polynomials" was unknown at the time). Jacobi proved the following theorem.

Theorem A.1 (Jacobi)**.** *Given an integer k such that $0 \leq k < n$, the quadrature rule Q_n has degree of exactness $d(Q_n) = n - 1 + k$ iff it has degree of exactness $n - 1$ and the node polynomial ω_n is orthogonal to all polynomials of degree at most $k - 1$, i.e., $I(\omega_n p)$ for all polynomials p with $\deg p \leq k - 1$.*

From the orthogonality condition we see that $d(Q_n) \leq 2n - 1$, since the node polynomial ω_n cannot be orthogonal to itself.

Proof. Clearly, if Q_n is exact for polynomials of degree $2n - 1$ or less, it is exact for polynomials of degree $n - 1$ or less. Moreover, if p is a polynomial of degree at most $k - 1$, then $\omega_n p$ is a polynomial of degree at most $n + (k - 1)$. By hypothesis,

$R_n(\omega_n p) = 0$, so $I(\omega_n p) = Q_n(\omega_n p)$. However,

$$Q_n(\omega_n p) = \sum_{j=1}^{n} \lambda_j \omega_n(\xi_j) p(\xi_j) = 0, \tag{A.17}$$

since ω_n vanishes at each node ξ_j.

On the other hand, suppose p is an arbitrary polynomial of degree at most $n-1+k$. By the Division Algorithm there exist polynomials q of degree at most $k-1$ and r of degree at most $n-1$ such that $p = \omega_n q + r$. Then $I(p) = I(\omega_n q) + I(r) = I(r)$, since ω_n is orthogonal to polynomials of degree less than k. But Q_n is exact for r, so

$$I(p) = I(r) = Q_n(r) = Q_n(p) - Q_n(q\omega_n) = Q_n(p). \tag{A.18}$$

Thus $d(Q_n) = n - 1 + k$ as desired. □

For the case $k = n$, Jacobi's theorem tells us that the node polynomial ω_n is orthogonal to all polynomials of lower degree; if we take the interval of integration to be $[-1, 1]$ (which we can always do by an affine change of variables), we see that ω_n is a scalar multiple of the n-th Legendre polynomial P_n defined in Chapter 3. Thus, using the zeros of P_n as nodes for Q_n provides us with a quadrature rule of maximum degree of exactness.

Jacobi also managed to show (using the above line of reasoning and the Rodrigues formula for the Legendre polynomials) that the nodes ξ_j (i.e., the zeros of P_n) are real, simple, and contained in $(-1, 1)$ [9].

The quadrature method discussed above, particularly the case $k = n$, came to be known as **Gauss-Jacobi** quadrature. Over the next fifty years, several mathematicians devised variants on the Gauss-Jacobi method, in which the Legendre polynomials were replaced by other, newly discovered families of orthogonal polynomials (the Chebyshev polynomials, the Laguerre polynomials, etc.)

A.3 Generalizations to Weighted Integrals

Almost half a century later, the German mathematician Elwin Bruno Christoffel produced a more general version of the Gauss-Jacobi quadrature that superseded all these variants. Christoffel extended Jacobi's theorem and Gauss's continued fraction proof to weighted integrals over finite intervals. About fifteen years later, Thomas Stieltjes would extend these results to integrals with respect to Stieltjes measures on subsets of the real line.

In what follows, we will consider integrals with respect to (positive) Stieltjes measures:

$$I(f) = \int_a^b f(x)\, d\lambda(x).$$

Here $[a, b] \subseteq \mathbb{R}$ is a (possibly) infinite interval and $d\lambda(x)$ is a Stieltjes measure on $[a, b]$. It is assumed that $\lambda(x)$ has infinitely many points of increase, and that the measure $d\lambda(x)$ has finite moments of all orders; that is, $\int_a^b x^n \, d\lambda(x) < \infty$, for all $n \geq 0$.

The quadrature formula analogous to (A.6) and possessing degree of exactness $2n-1$ is called a **Gauss-Christoffel quadrature formula**; its weights λ_j are referred to as the **Christoffel numbers** for measure $d\lambda$.

Christoffel's theory of quadrature rests upon orthogonal polynomial theory. As we saw in Chapter 3, there is a unique family $\{p_n\}$ of monic orthogonal polynomials associated with the real inner product induced by $d\lambda$ ($\langle v, w \rangle = \int_a^b vw \, d\lambda$). This family satisfies a three-term recurrence (see Chapter 3) with real coefficients.

Jacobi's theorem (Theorem A.1) extends unchanged to Christoffel's theory. We again find that the nodes ξ_j of the quadrature rule Q_n are the zeros of the corresponding orthogonal polynomial p_n, and that the nodes are real, simple, and contained in the interval (a, b).

Finally, we again find that all Christoffel numbers are positive. Consider the integral

$$\int_a^b (l_j(x))^2 \, d\lambda(x). \tag{A.19}$$

Its value is clearly positive. The integrand $(l_j(x))^2$ is a polynomial of degree at most $2n-2$, so our quadrature formula computes (A.19) exactly:

$$\int_a^b (l_j(x))^2 \, d\lambda(x) = \sum_{k=1}^{n} \lambda_k (l_j(\xi_k))^2 = \lambda_j (l_j(\xi_j))^2 = \lambda_j. \tag{A.20}$$

Therefore the λ_j's are all positive [9]. Golub and Strakos [10] give the following explicit formula for the Christoffel numbers:

$$\lambda_j = \frac{\|p_{n-1}\|_w^2}{p_{n-1}(\xi_j) p_n'(\xi_j)}. \tag{A.21}$$

This formula follows from taking $f = p_{n-1}(x)p_n(x)/(x-\xi_j)$ in the Gauss-Christoffel quadrature and computing the integral in two different ways: from (A.27) we have the equality

$$\begin{aligned}\int_a^b p_{n-1}(x) \frac{p_n(x)}{x-\xi_j} \, dw(x) &= \sum_{\substack{k=1\\k\neq j}}^{n} \lambda_k p_{n-1}(\xi_k) \frac{p_n(\xi_k)}{\xi_k - \xi_j} + \lambda_j p_{n-1}(\xi_j) p_n'(\xi_j) \\ &= \sum_{\substack{k=1\\k\neq j}}^{n} \lambda_k p_{n-1}(\xi_k) p_n'(\xi_k) l_k(\xi_j) + \lambda_j p_{n-1}(\xi_j) p_n'(\xi_j) \\ &= \lambda_j p_{n-1}(\xi_j) p_n'(\xi_j).\end{aligned}$$

But by the orthogonality of the p_n's, we have

$$\int_a^b p_{n-1}(x) \frac{p_n(x)}{x-\xi_j} \, dw(x) = \int_a^b \left(p_{n-1}(x)\right)^2 \, dw(x) = \|p_{n-1}\|_w^2. \tag{A.22}$$

Thus (A.21) is established.

Gauss's idea of considering generating functions for the monomial errors can also be extended to weighted integrals. Define the three functions

$$L(z) = \int_a^b \frac{1}{z-x} \, d\lambda(x), \quad \rho_n(z) = \int_a^b \frac{p_n(z)}{z-x} \, d\lambda(x), \quad z \notin [a,b]$$
$$\sigma_n(z) = \int_a^b \frac{p_n(z) - p_n(x)}{z-x} \, d\lambda(x). \tag{A.23}$$

It is plain that we have the equality

$$p_n(z)L(z) = \sigma_n(z) + \rho_n(z). \tag{A.24}$$

Equation (A.24) represents p_nL as the formal sum of a polynomial (σ_n) and a generating series involving only negative powers of z:

$$\begin{aligned}
\rho_n(z) &= \int_a^b \frac{1}{z-x} p_n(z)\, d\lambda(x) \\
&= \int_a^b \sum_{k=0}^{\infty} \frac{x^k}{z^{k+1}} p_n(z)\, d\lambda(x) \\
&= \sum_{k=0}^{\infty} \frac{r_k}{z^{k+1}}, \quad r_k = \int_a^b x^k p_n(x)\, d\lambda(x). \qquad (A.25)
\end{aligned}$$

This expansion, combined with the orthogonality of the $p_n(z)$, shows that $\rho_n(z) = O(z^{-n-1})$. Since p_n has degree n, we conclude that

$$L(z) - \frac{\sigma_n(z)}{p_n(z)} = \frac{\rho_n(z)}{p_n(z)} = O\left(\frac{1}{z^{2n+1}}\right). \tag{A.26}$$

Following the example of Gauss, we decompose σ_n/p_n into partial fractions, using the nodes ξ_j as poles and the weights λ_j as residues, to obtain a formula for Q_n:

$$\frac{\sigma_n(z)}{p_n(z)} = \sum_{j=1}^{n} \frac{\lambda_n}{z-\xi_j} := Q_n\left(\frac{1}{z-x}\right). \tag{A.27}$$

We may obtain an exact formula for the Christoffel numbers by standard techniques of complex analysis:

$$\lambda_j = \lim_{z\to\xi_j} (z-\xi_j)\frac{\sigma_n(z)}{p_n(z)} = \frac{\sigma_n(\xi_j)}{p_n'(\xi_j)}. \tag{A.28}$$

From (A.27) we now conclude that

$$\begin{aligned}
\frac{\rho_n(z)}{p_n(z)} &= L(z) - \frac{\sigma_n(z)}{p_n(z)} = I\left(\frac{1}{z-x}\right) - Q_n\left(\frac{1}{z-x}\right) \qquad (A.29) \\
&:= R_n\left(\frac{1}{z-x}\right) \qquad (A.30) \\
&= \sum_{k=0}^{\infty} \frac{R_n(x^k)}{z^{k+1}}, \qquad (A.31)
\end{aligned}$$

as before. The last expression, when combined with (A.26), also shows that the monomial errors $R_n(t^k)$ vanish for $0 \leq k \leq 2n-1$, verifying the desired degree of exactness.

We can also form a continued fraction expansion for $L(z)$ as before by expanding $L - \sigma_n/p_n$ in powers of $1/z$:

$$L(z) = \cfrac{b_0}{(z-a_0) - \cfrac{b_1}{(z-a_1) - \cfrac{b_2}{(z-a_2) - \cdots}}}. \tag{A.32}$$

The n-th convergent of this continued fraction will be σ_n/p_n. As Gautschi notes [9], this characterization of orthogonal polynomials as denominators of convergents of continued fractions was very popular in the nineteenth century. Indeed, from standard three-term recurrences for the numerators and denominators of continued fractions one may easily derive the standard three-term recurrence for orthogonal polynomials [30]. (The numerator σ_n is merely the second solution of the recurrence for p_n.)

A.4 Gauss-Christoffel Quadrature with Preassigned Nodes

Finally, we discuss briefly the extension of Gauss-Christoffel quadrature to quadrature rules with preassigned nodes. In such formulae, the quadrature sum Q_n takes the form

$$Q_n(f) = \sum_{j=1}^{m} \omega_j f(\mu_j) + \sum_{j=1}^{n} \lambda_j f(\xi_j), \tag{A.33}$$

where the ω_j and λ_j's are weights, the μ_j's are nodes prescribed in advance, and the ξ_j's are nodes to be determined so that the rule has maximum degree of exactness. In the special case of one preassigned node, the endpoint of the interval, these formulae are referred to as **Gauss-Radau quadrature** (left endpoint) and **Gauss-Lobatto quadrature** (right endpoint).

Since there are $m + 2n$ unknowns (ω_j's, λ_j's and ξ_j's), we would surmise this maximum to be $m + 2n - 1$, and indeed that is the case. The following analogue of Jacobi's theorem holds for Gauss-Christoffel quadrature rules with preassigned nodes [5]:

Theorem A.2. *The quadrature rule given by* (A.33) *has degree of exactness* $m+2n-1$ *iff it has degree of exactness* $m + n - 1$ *and the node polynomial*

$$\omega(x) = \left(\prod_{j=1}^{n}(x - \xi_j)\right)\left(\prod_{j=1}^{m}(x - \mu_j)\right) := r(x)s(x) \tag{A.34}$$

is orthogonal (in the inner product induced by $\lambda(x)$*) to all polynomials of degree at most* $n - 1$.

The proof of this theorem is similar to that of Jacobi's Theorem (Theorem A.1).

The nodes ξ_j are once again the zeros of orthogonal polynomials, only now the polynomials are orthogonal with respect to the weight $s(x)\lambda(x)$ instead of $\lambda(x)$. Christoffel derived an expression for these polynomials in terms of (monic) polynomials orthogonal with respect to $\lambda(x)$:

Theorem A.3 (Christoffel's Theorem)**.** *Let* $\{p_n(x)\}$ *be a family of orthogonal polynomials on* $[a, b]$ *with respect to the weight* $\lambda(x)$*. Let* $s(x)$ *be defined as in* (A.34) *and suppose that the* μ_j*'s are distinct. Suppose that* $\{q_n(x)\}$ *is a family of orthogonal polynomials on* $[a, b]$ *with respect to the weight* $s(x)\lambda(x)$*. Then*

$$s(x)q_n(x) = \begin{vmatrix} p_n(x) & p_{n+1}(x) & \cdots & p_{n+m}(x) \\ p_n(\mu_1) & p_{n+1}(\mu_1) & \cdots & p_{n+m}(\mu_1) \\ \vdots & \vdots & \ddots & \vdots \\ p_n(\mu_m) & p_{n+1}(\mu_m) & \cdots & p_{n+m}(\mu_m) \end{vmatrix}. \tag{A.35}$$

For the proof of this theorem, see [5].

Appendix B

BASIC ARITHMETIC OPERATIONS IN FINITE-PRECISION

In this chapter we review some elementary facts about finite-precision arithmetic. We will remain faithful to the treatment of Wilkinson's text [32]. We will only examine the case of floating-point arithmetic; similar statements can be made for fixed-point arithmetic.

Throughout this section, $\mathbf{A}$ will denote an $m \times m$ matrix, $\mathbf{x}$ and $\mathbf{y}$ will denote $m \times 1$ vectors, and c will denote a scalar. Furthermore, ν will denote the maximum number of non-zero elements in any row of $\mathbf{A}$. We will adopt the notation $fl(\mathbf{x})$ to denote the floating-point representation of $\mathbf{x}$. We will also engage in a slight abuse of notation and use ϵ to refer to any quantity smaller than the machine precision. Finally, we will assume that the quantities

$$\sigma := \|\mathbf{A}\| \quad \text{and} \quad \beta\sigma := \||\mathbf{A}|\|\,, \tag{B.1}$$

where $|\mathbf{A}| = |a_{ij}|$, are known *a priori.*

First, we recall how floating-point arithmetic affects basic operations. Here we assume that intermediate results are stored in an "accumulator" of the same size as the operands. This assumption is valid for double-precision IEEE floating-point arithmetic, which is the standard for numerical calculations [19].

If x is the exact representation of a quantity, then $fl(x) = x(1+\epsilon)$, where ϵ is the machine precision. The floating-point sum/difference of two floating-point numbers x_1 and x_2 is given by

$$fl(x_1 \pm x_2) = (x_1 \pm x_2)(1+\epsilon).$$

For multiplication of two floating-point numbers x_1 and x_2 we have

$$fl(x_1 * x_2) = (x_1 * x_2)(1+\epsilon),$$

while for division, we have the analogous statement

$$fl(x_1/x_2) = (x_1/x_2)(1+\epsilon),$$

provided x_2 is not zero. Division by zero is typically defined in floating-point arithmetic either to return a value indicating infinity or to signal an error to the program. It is then the responsibility of the program to decide how the attempted division should be handled.

Our first three propositions establish relations between basic matrix and vector operations and their finite-precision counterparts.

Proposition B.1.

$$fl\left(fl(\mathbf{x}) - fl(c)\, fl(\mathbf{y})\right) = (\mathbf{x} - c\mathbf{y}) + \delta\mathbf{z}, \quad \textit{where } \|\delta\mathbf{z}\| \le \left(\|\mathbf{x}\| + 2|c|\,\|\mathbf{y}\|\right)\epsilon \tag{B.2}$$

Proof.

$$fl\left(fl(\mathbf{x}) - fl(c)\, fl(\mathbf{y})\right) = \left(\mathbf{x} - c\mathbf{y}(1+\epsilon)\right)(1+\epsilon) = (\mathbf{x} - c\mathbf{y}) + (\mathbf{x} - 2c\mathbf{y})\,\epsilon,$$

since finite-precision quantities are known only to $O(\epsilon)$ accuracy;

$$= (\mathbf{x} - c\mathbf{y}) + \delta\mathbf{z},$$

where

$$\delta\mathbf{z} \le \left(\|\mathbf{x}\| + 2|c|\,\|\mathbf{y}\|\right)\epsilon.$$

□

Proposition B.2.

$$fl\left(fl(\mathbf{y})^* fl(\mathbf{x})\right) = (\mathbf{y} + \delta\mathbf{y})^* x \quad \textit{where } \|\delta\mathbf{y}\| \le m\epsilon\,\|\mathbf{y}\|\,. \tag{B.3}$$

Proof. From the definition of the inner product, we have

$$fl\left(fl(\mathbf{y})^* fl(\mathbf{x})\right) = fl\left(fl(\overline{\mathbf{y}}_1)\, fl(\mathbf{x}_1) + \cdots + fl(\overline{\mathbf{y}}_m)\, fl(\mathbf{x}_m)\right).$$

We avail ourselves of the following trick [32]. Let $f_j = fl(fl(\overline{\mathbf{y}}_j)\, fl(\mathbf{x}_j))$ and define the partial sums s_j by $s_1 = f_1$ and $s_j = fl(s_{j-1} + f_j)$. For each j we have $f_j = \overline{\mathbf{y}}_j \mathbf{x}_j(1+\epsilon)$ and $s_j = (s_{j-1}+f_j)(1+\epsilon)$ in finite-precision. Combining these two facts and inducting on j we have

$$\begin{aligned}
s_1 &= \overline{\mathbf{y}}_1\mathbf{x}_1(1+\epsilon)\\
s_2 &= \left[\overline{\mathbf{y}}_1\mathbf{x}_1(1+\epsilon) + \overline{\mathbf{y}}_2\mathbf{x}_2(1+\epsilon)\right](1+\epsilon) = \left(\overline{\mathbf{y}}_1\mathbf{x}_1 + \overline{\mathbf{y}}_2\mathbf{x}_2\right)(1+\epsilon)^2\\
&\vdots\\
s_m &= \overline{\mathbf{y}}_1\mathbf{x}_1(1+\epsilon)^m + \overline{\mathbf{y}}_2\mathbf{x}_2(1+\epsilon)^m + \cdots + \overline{\mathbf{y}}_{m-1}\mathbf{x}_{m-1}(1+\epsilon)^3\\
&\qquad + \overline{\mathbf{y}}_m\mathbf{x}_m(1+\epsilon)^2\\
&= \mathbf{y}^*\mathbf{x} + \delta\mathbf{y}^*\mathbf{x},
\end{aligned}$$

where $\delta\mathbf{y}$ satisfies

$$\|\delta\mathbf{y}\| = \left\|\begin{bmatrix} m\mathbf{y}_1 & m\mathbf{y}_2 & \dots & 2\mathbf{y}_m\end{bmatrix}^* \epsilon\right\| \leq m\epsilon\,\|\mathbf{y}\|,$$

as claimed. □

Proposition B.3.

$$fl\left(fl(\mathbf{A})\, fl(\mathbf{x})\right) = (\mathbf{A} + \delta\mathbf{A})\,\mathbf{x}, \quad \textit{where } |\delta\mathbf{A}| \leq \nu\epsilon|\mathbf{A}|. \tag{B.4}$$

Proof. Let $\mathbf{A}_i$ denote the i-th row of $\mathbf{A}$. In exact arithmetic we have

$$\mathbf{A}\mathbf{x} = \begin{bmatrix}\mathbf{A}_1\mathbf{x} & \dots & \mathbf{A}_m\mathbf{x}\end{bmatrix}^T.$$

We have already seen how finite-precision arithmetic affects inner products: in the proof of Proposition B.2, we showed that

$$fl\left(fl(\mathbf{A}_i)\, fl(\mathbf{x})\right) = (\mathbf{A}_i + \delta\mathbf{A}_i)\,\mathbf{x},$$

where the error term $\delta\mathbf{A}_i\mathbf{x}$ is given by

$$\delta\mathbf{A}_i\mathbf{x} = [m\mathbf{A}_{i1}\mathbf{x}_1 + m\mathbf{A}_{i2}\mathbf{x}_2 + \cdots + 2\mathbf{A}_{im}\mathbf{x}_m]\,\epsilon.$$

Some of these entries, however, may be zero, so it is not necessary to compute them all. If ν is the maximum number of non-zero entries in any row, then we perform at most ν multiplies and $\nu - 1$ additions in the process of computing $\mathbf{A}_i\mathbf{x}$. Therefore, we can bound $|\delta\mathbf{A}_i|$ by $\nu\epsilon|\mathbf{A}_i|$ using the same argument as in the proof of the previous proposition. It follows that $|\delta\mathbf{A}| \leq \nu\epsilon|A|$. □

When (B.4) is combined with our assumption (B.1), we get the bound

$$\|\delta\mathbf{A}\| \leq \|\,|\delta\mathbf{A}|\,\| \leq \nu\epsilon\,\|\,|\mathbf{A}|\,\| = \nu\beta\epsilon\sigma. \tag{B.5}$$

Our next proposition details how finite-precision arithmetic affects calculations with norms.

Proposition B.4. *Assume that taking square roots introduces a relative error no greater than ϵ. Then*

$$c = +fl\left(\sqrt{fl(\mathbf{x})^*\,fl(\mathbf{x})}\right) = \left(1 + \frac{1}{2}(m+2)\epsilon\right)\|\mathbf{x}\| \tag{B.6}$$

$$y = fl\left(\frac{fl(\mathbf{x})}{c}\right) = \operatorname{diag}(1+\epsilon)\,\mathbf{x}/c \tag{B.7}$$

$$y^*y = 1 + (m+4)\,\epsilon. \tag{B.8}$$

Proof. From Proposition B.2 we have

$$fl\left(\sqrt{\mathbf{x}^*\mathbf{x}}\right) = \sqrt{fl\left(fl(\mathbf{x})^*\,fl(\mathbf{x})\right)} + \epsilon\,\|\mathbf{x}\| \leq \sqrt{1+m\epsilon}\,\|\mathbf{x}\| + \epsilon\,\|\mathbf{x}\|\,.$$

By Bernoulli's theorem (or a simple binomial formula estimate) we have the inequality $\sqrt{1+u} \leq 1 + \frac{1}{2}u$, from which (B.6) follows immediately. The second statement follows immediately from our earlier discussion of how floating-point arithmetic affects division. Finally,

$$\begin{aligned} y^* y &= \left(\mathbf{x}^* \operatorname{diag}\left(1+\epsilon\right)/c\right)\left(\operatorname{diag}\left(1+\epsilon\right)\mathbf{x}/c\right) \\ &= (1+2\epsilon)\mathbf{x}^*\mathbf{x}/\beta^2 = (1+2\epsilon)\left[1+\epsilon\left(\frac{m+2}{2}\right)\right]^{-2} \\ &= (1+2\epsilon)\left(1+(m+2)\epsilon\right)) = 1+(m+4)\,\epsilon. \end{aligned}$$

□

Appendix C

PROOF OF PROPOSITION 4.5

Proposition C.1. *Suppose that*

$$4n\left\{3\left(m+4\right)\epsilon+\left(7+\nu\beta\right)\epsilon\right\}\ll 1. \tag{C.1}$$

Then the bound

$$\|u_n\|\leq\sigma\left\{1+2n\left[3\left(m+4\right)\epsilon+\left(7+\nu\beta\right)\epsilon\right]\right\} \tag{C.2}$$

holds at each step of Algorithm 4.1.

Proof. The proof of this statement is seemingly complicated, but it boils down to repeated use of the results we have established so far.

First, we must establish some intermediate results. From (4.25) we have

$$\|u_n+\delta u_n\|^2=\|Aq_n-\beta_n q_{n-1}\|^2=\|Aq_n\|^2+\beta_n^2\,\|q_{n-1}\|^2-2\beta_n q_n^* Aq_{n-1} \tag{C.3}$$

and from (4.27)

$$\beta_n q_n^* Aq_{n-1}=\beta_n q_n^*\left(\beta_n q_n+\alpha_{n-1}q_{n-1}+\beta_{n-1}q_{n-2}+f_{n-1}\right)=\beta_n^2+\delta\beta_n.$$

The error term $\delta\beta_n$ is given explicitly by the formula

$$\delta\beta_n=\beta_n^2(q_n^* q_n-1)+\beta_n\alpha_{n-1}q_n^* q_{n-1}+\beta_n\beta_{n-1}q_n^* q_{n-2}+\beta_n q_n^* f_{n-1}. \tag{C.4}$$

It is clear from this equation that we will need a bound on quantities of the form $q_n^* q_{n-2}$. An easy way to do this is the following trick due to Paige [24]. First compare the $(j-1,j)$-elements of each side of (4.34) to obtain the identities

$$\alpha_1\rho_{12}-\alpha_2\rho_{12}-\beta_3\rho_{13}=\eta_{12} \tag{C.5}$$

$$\beta_{j-1}\rho_{j-2,j}+(\alpha_{j-1}-\alpha_j)\rho_{j-1,j}-\beta_{j+1}\rho_{j-1,j+1}=\eta_{j-1,j},\quad j=2,\ldots,n. \tag{C.6}$$

Notice the occurrence of the terms of the form $\rho_{j-2,j}$ in these identities. If we now multiply both sides of (C.5) and (C.6) by β_j and define

$$\zeta_j = (\alpha_{j-1} - \alpha_j)\, \beta_j \rho_{j-1,j} - \beta_j \eta_{j-1,j}, \quad j = 2, \ldots, n, \tag{C.7}$$

we obtain the recurrence

$$\beta_j \beta_{j+1} \rho_{j-1,j+1} = \beta_{j-1} \beta_j \rho_{j-2,j} + \zeta_j.$$

It follows immediately that $\beta_j \beta_{j+1} \rho_{j-1,j+1} = \zeta_j + \cdots + \zeta_2$.

Using several of the estimates we have developed in the previous subsections, we see that

$$|\zeta_j| \leq 2\left[(1 + \nu\beta)\sigma + (3n + 18)\mu_j\right] \mu_j \epsilon.$$

Hence we obtain the bound

$$|\beta_j||\beta_{j+1}||\rho_{j-1,j+1}| \leq 2(j-1)\left[(1 + \nu\beta)\sigma + (3n + 18)\mu_j\right] \mu_j \epsilon. \tag{C.8}$$

Using the above results with (4.25) we get

$$\begin{aligned} \|u_n\|^2 &= \|Aq_n - \beta_{n-1} q_{n-1} + \delta u_n + \delta A q_n\|^2 \\ &= \|Aq_n\|^2 + \beta_n^2 (\|q_{n-1}\|^2 - 2) + \|u_n\|^2 + 2\beta_n^2 \\ &\qquad - 2(\delta u_n)^* \left(Aq_n - \beta_n q_{n-1}\right) - 2\beta_n q_n^* A q_{n-1} \\ &= \|Aq_n\|^2 + \beta_n^2 (\|q_{n-1}\|^2 - 2) + \delta\beta_n', \end{aligned} \tag{C.9}$$

where the error term $\delta\beta_n'$ satisfies

$$\begin{aligned} |\delta\beta_n'| &= \left| \|u_n\|^2 + 2\beta_n^2 - 2\beta_n q_n^* A q_{n-1} - 2(\delta u_n)^* \left(Aq_n - \beta_n q_{n-1}\right) \right| \\ &\leq \left\{4(n-1)(1 + \nu\beta)\sigma + \left[(2n-3)6(m+6) + 4\right] \mu_n\right\} \mu_n \epsilon. \end{aligned} \tag{C.10}$$

We are now ready to prove Proposition 4.5. Let

$$\mu := \max(\mu_n, \sigma).$$

If $\mu = \sigma$, then $\|u_j\| \le \sigma$ for each $j \le n$, so (4.39) clearly holds. On the other hand, if $\mu = \mu_n$, then for $j = 1, \ldots, n$ we have

$$\begin{aligned}
\|u_j\|^2 &= \|Aq_j\|^2 + \beta_j^2(\|q_{j-1}\|^2 - 2) + \delta\beta_j' \\
&\le \sigma^2 \left(1 + (m+4)\epsilon\right) + \left(1 + (m+4)\epsilon\right)\left(1 + (4m+12)\epsilon\right)\mu_{n-1}^2 \\
&\qquad + \left\{4(n-1)(1+\nu\beta)\sigma + \left[(2n-3)6(m+6) + 4\right]\mu_n\right\}\mu_n\epsilon \\
&\le \sigma^2 + 4n\left[(7+\nu\beta) + 3(m+4)\right]\mu^2\epsilon.
\end{aligned}$$

Thus, since the above bound holds for $j = 1, \ldots, n$, we have

$$\mu^2 \le \sigma^2 + \left\{4n\left((7+\nu\beta) + 3(m+4)\right)\epsilon\right\}\mu^2,$$

which implies

$$\mu^2 \le \frac{\sigma^2}{1 - \left\{4n\left((7+\nu\beta) + 3(m+4)\right)\epsilon\right\}} = \sigma^2\left(1 + \left\{4n\left((7+\nu\beta) + 3(m+4)\right)\epsilon\right\}\right)$$

to first order, since we assumed (C.1). This proves (C.2). □

BIBLIOGRAPHY

[1] Dario Bini and Victor Pan. Computing matrix eigenvalues and polynomials zeros where the output is real. *SIAM J. on Computing*, 27(4):1099–1115, 1998.

[2] Mike Botchev. A. N. Krylov: A Short Biography. http://ta.twi.tudelft.nl/users/vuik/burgers/krylov.html (7 July 2001).

[3] T. S. Chihara. *An Introduction to Orthogonal Polynomials.* Gordon and Breach Science Publishers, New York, 1978.

[4] William Connett and Alan Schwartz. Lecture notes on orthogonal polynomials. Unpublished. From an REU at the University of Missouri, St. Louis, Summer 1997.

[5] Philip J. Davis and Philip Rabinowitz. *Numerical Integration.* Blaisedell Publishing Co., Waltham, MA-Toronto-London, 1967.

[6] Stanley C. Eisenstat and Ming Gu. A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem. *SIAM J. Matrix Analysis Appl.*, 16:172–191, 1995.

[7] Jean Favard. Sur les polynomes de Tchebicheff. *Comptes Rendus de l'Academie des sciences*, 200:2052–2053, 1935.

[8] Gerald Folland. *Real Analysis: Modern Techniques and Their Applications.* John Wiley & Sons, Inc., New York, NY, second edition, 1999.

[9] W. Gautschi. A survey of Gauß-Christoffel quadrature formulae. In P.L. Bultzer and F. Fehér, editors, *E. B. Christoffel – The Influence of His Work on Mathematics and the Physical Sciences*, pages 73–157. Birkhauser, Boston, MA, 1981.

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

[11] C. George Green. Connections Between Lanczos Iteration and Orthogonal Polynomials. Master's thesis, University of Washington, Seattle, Washington, August 2001.

[12] Anne Greenbaum. Conversations with author. Various dates.

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

[14] Anne Greenbaum. *Iterative Methods for Solving Linear Systems.* SIAM Press, Philadelphia, PA, 1997.

[15] Leslie F. Greengard. *The Rapid Evaluation of Potential Fields in Particle Systems.* MIT Press, Cambridge, MA, 1988.

[16] Magnus R. Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Res. Natural Bureau Standards*, 49:409–436, 1952.

[17] Harry Hochstadt. *Special Functions of Mathematical Physics.* Holt, Rinehart and Winston, New York, 1961.

[18] Roger A. Horn and Charles R. Johnson. *Matrix Analysis.* Cambridge University Press, Cambridge, UK, 1985.

[19] IEEE. IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Standard, Std 754-1984, New York, 1985.

[20] Donald L. Krieder et al. *An Introduction to Linear Analysis.* Addison-Wesley Publishing Co., Reading, MA, 1966.

[21] Alexei N. Krylov. On the numerical solution of the equation by which, in technical matters, frequencies of small oscillations of material systems are determined. *Izvestija AN S.S.S.R. (News of [the] Academy of Sciences of the U.S.S.R), Otdel. mat. i estest. nauk*, 7(4):491–539, 1931. In Russian.

[22] Cornelius Lanczos. Solution of systems of linear equations by minimized iterations. *Journal of Res. Natural Bureau of Standards*, 49:33–53, 1952.

[23] Chris C. Paige. *The Computation of Eigenvalues and Eigenvectors of Very Large Sparse Matrices.* PhD thesis, University of London, 1971.

[24] Chris C. Paige. Error analysis of the Lanczos algorithm for tridiagonalizing a symmetric matrix. *Journal of the Institute of Mathematical Applications*, 18:341–349, 1976.

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

[26] Beresford N. Parlett. *The Symmetric Eigenvalue Problem.* Prentice-Hall Series in Computational Mathematics. Prentice-Hall, Englewood Cliffs, NJ, 1980.

[27] Theodore J. Rivlin. *The Chebyshev Polynomials.* John Wiley & Sons, New York, 1974.

[28] V. Rokhlin. Rapid solution of integral equations of classical potential theory. *J. Comp. Phys.*, 60:187–207, 1983.

[29] Akbar Shetty. Better living through functional analysis. Manuscript, in progress.

[30] Gabor Szegö. *Orthogonal Polynomials*, volume 23 of *American Mathematical Society Colloquium Publishings.* American Mathematical Society, Providence, RI, fourth edition, 1939.

[31] Lloyd N. Trefethen and David Bau, III. *Numerical Linear Algebra.* SIAM Press, Philadelphia, PA, 1997.

[32] J. H. Wilkinson. *Rounding Errors in Algebraic Processes.* Prentice-Hall, Englewood Cliffs, NJ, 1963.