7

Iterative Methods for Eigenvalue Problems

7.1. Introduction

In this chapter we discuss iterative methods for finding eigenvalues of matrices that are too large to use the direct methods of Chapters 4 and 5. In other words, we seek algorithms that take far less than $O(n^2)$ storage and $O(n^3)$ flops. Since the eigenvectors of most n-by-n matrices would take n^2 storage to represent, this means that we seek algorithms that compute just a few user-selected eigenvalues and eigenvectors of a matrix.

We will depend on the material on Krylov subspace methods developed in section 6.6, the material on symmetric eigenvalue problems in section 5.2, and the material on the power method and inverse iteration in section 5.3. The reader is advised to review these sections.

The simplest eigenvalue problem is to compute just the largest eigenvalue in absolute value, along with its eigenvector. The power method (Algorithm 4.1) is the simplest algorithm suitable for this task: Recall that its inner loop is

$$\begin{aligned} y_{i+1} &= Ax_i, \\ x_{i+1} &= y_{i+1}/\|y_{i+1}\|_2, \end{aligned}$$

where x_i converges to the eigenvector corresponding to the desired eigenvector (provided that there is only one eigenvalue of largest absolute value, and x_1 does not lie in an invariant subspace not containing its eigenvector). Note that the algorithm uses A only to perform matrix-vector multiplication, so all that we need to run the algorithm is a "black-box" that takes x_i as input and returns Ax_i as output (see Example 6.13).

A closely related problem is to find the eigenvalue closest to a user-supplied value σ, along with its eigenvector. This is precisely the situation inverse iteration (Algorithm 4.2) was designed to handle. Recall that its inner loop is

$$\begin{aligned} y_{i+1} &= (A - \sigma I)^{-1} x_i, \\ x_{i+1} &= y_{i+1}/\|y_{i+1}\|_2, \end{aligned}$$

i.e., solving a linear system of equations with coefficient matrix $A-\sigma I$. Again x_i converges to the desired eigenvector, provided that there is just one eigenvalue closest to σ (and x_1 satisfies the same property as before). Any of the sparse matrix techniques in Chapter 6 or section 2.7.4 could be used to solve for y_{i+1}, although this is usually much more expensive than simply multiplying by A. When A is symmetric Rayleigh quotient iteration (Algorithm 5.1) can also be used to accelerate convergence (although it is not always guaranteed to converge to the eigenvalue of A closest to σ).

Starting with a given x_1, $k-1$ iterations of either the power method or inverse iteration produce a sequence of vectors $x_1, x_2, \ldots, x_k$. These vectors span a *Krylov subspace*, as defined in section 6.6.1. In the case of the power method, this Krylov subspace is $\mathcal{K}_k(A, x_1) = \text{span}[x_1, Ax_1, A^2x_1, \ldots, A^{k-1}x_1]$, and in the case of inverse iteration this Krylov subspace is $\mathcal{K}_k((A-\sigma I)^{-1}, x_1)$. Rather than taking x_k as our approximate eigenvector, it is natural to ask for the "best" approximate eigenvector in $\mathcal{K}_k$, i.e., the best linear combination $\sum_{i=1}^{k} \alpha_i x_i$. We took the same approach for solving $Ax = b$ in section 6.6.2, where we asked for the best approximate solution to $Ax = b$ from $\mathcal{K}_k$. We will see that the best eigenvector (and eigenvalue) approximations from $\mathcal{K}_k$ are much better than x_k alone. Since $\mathcal{K}_k$ has dimension k (in general), we can actually use it to compute k best approximate eigenvalues and eigenvectors. These best approximations are called the *Ritz values* and *Ritz vectors*.

We will concentrate on the symmetric case $A = A^T$. In the last section we will briefly describe the nonsymmetric case.

The rest of this chapter is organized as follows. Section 7.2 discusses the Rayleigh–Ritz method, our basic technique for extracting information about eigenvalues and eigenvectors from a Krylov subspace. Section 7.3 discusses our main algorithm, the Lanczos algorithm, in exact arithmetic. Section 7.4 analyzes the rather different behavior of the Lanczos algorithm in floating point arithmetic, and sections 7.5 and 7.6 describe practical implementations of Lanczos that compute reliable answers despite roundoff. Finally, section 7.7 briefly discusses algorithms for the nonsymmetric eigenproblem.

7.2. The Rayleigh–Ritz Method

Let $Q = [Q_k, Q_u]$ be any n-by-n orthogonal matrix, where Q_k is n-by-k and Q_u is n-by-$(n-k)$. In practice the columns of Q_k will be computed by the Lanczos algorithm (Algorithm 6.10 or Algorithm 7.1 below) and span a Krylov subspace $\mathcal{K}_k$, and the subscript u indicates that Q_u is (mostly) unknown. But for now we do not care where we get Q.

We will use the following notation (which was also used in equation (6.31)):

$$T = Q^T A Q = [Q_k, Q_u]^T A [Q_k, Q_u] = \begin{bmatrix} Q_k^T A Q_k & Q_k^T A Q_u \\ Q_u^T A Q_k & Q_u^T A Q_u \end{bmatrix}$$

$$\begin{aligned} & & k & & n-k \\ \equiv & k & \begin{pmatrix} T_k & & T_{uk} \end{pmatrix} \\ & n-k & \begin{pmatrix} T_{ku} & & T_u \end{pmatrix} \\ = & \begin{bmatrix} T_k & T_{ku}^T \\ T_{ku} & T_u \end{bmatrix}. \end{aligned} \tag{7.1}$$

When $k = 1$, T_k is just the Rayleigh quotient $T_1 = \rho(Q_1, A)$ (see Definition 5.1). So for $k > 1$, T_k is a natural generalization of the Rayleigh quotient.

DEFINITION 7.1. *The* Rayleigh–Ritz procedure *is to approximate the eigenvalues of A by the eigenvalues of $T_k = Q_k^T A Q_k$. These approximations are called* Ritz values. *Let $T_k = V \Lambda V^T$ be the eigendecomposition of T_k. The corresponding eigenvector approximations are the columns of $Q_k V$ and are called* Ritz vectors.

The Ritz values and Ritz vectors are considered *optimal* approximations to the eigenvalues and eigenvectors of A for several reasons. First, when Q_k and so T_k are known but Q_u and so T_{ku} and T_u are unknown, the Ritz values and vectors are the natural approximations from the known part of the matrix. Second, they satisfy the following generalization of Theorem 5.5. (Theorem 5.5 showed that the Rayleigh quotient was a "best approximation" to a single eigenvalue.) Recall that the columns of Q_k span an invariant subspace of A if and only if $AQ_k = Q_k R$ for some matrix R.

THEOREM 7.1. *The minimum of $\|AQ_k - Q_k R\|_2$ over all k-by-k symmetric matrices R is attained by $R = T_k$, in which case $\|AQ_k - Q_k R\|_2 = \|T_{ku}\|_2$. Let $T_k = V \Lambda V^T$ be the eigendecomposition of T_k. The minimum of $\|AP_k - P_k D\|_2$ over all n-by-k orthogonal matrices P_k where* $\text{span}(P_k) = \text{span}(Q_k)$ *and over diagonal D is also $\|T_{ku}\|_2$ and is attained by $P_k = Q_k V$ and $D = \Lambda$.*

In other words, the columns of $Q_k V$ (the Ritz vectors) are the "best" approximate eigenvectors and the diagonal entries of Λ (the Ritz values) are the "best" approximate eigenvalues in the sense of minimizing the residual $\|AP_k - P_k D\|_2$.

Proof. We temporarily drop the subscripts k on T_k and Q_k to simplify notation, so we can write the k-by-k matrix $T = Q^T A Q$. Let $R = T + Z$. We want to show $\|AQ - QR\|_2^2$ is minimized when $Z = 0$. We do this by using a disguised form of the Pythagorean theorem:

$$\begin{aligned} \|AQ - QR\|_2^2 &= \lambda_{\max}\left[(AQ - QR)^T (AQ - QR)\right] \\ &\qquad \text{by Part 7 of Lemma 1.7} \\ &= \lambda_{\max}\left[(AQ - Q(T+Z))^T (AQ - Q(T+Z))\right] \\ &= \lambda_{\max}\left[(AQ - QT)^T (AQ - QT) - (AQ - QT)^T (QZ)\right. \\ &\quad \left. -(QZ)^T (AQ - QT) + (QZ)^T (QZ)\right] \end{aligned}$$

$$
\begin{aligned}
&= \lambda_{\max}\left[(AQ-QT)^T(AQ-QT) - (Q^TAQ-T)Z\right. \\
&\quad \left. -Z^T(Q^TAQ-T) + Z^TZ\right] \\
&= \lambda_{\max}\left[(AQ-QT)^T(AQ-QT) + Z^TZ\right] \\
&\qquad \text{because } Q^TAQ = T \\
&\geq \lambda_{\max}\left[(AQ-QT)^T(AQ-QT)\right] \\
&\qquad \text{by Question 5.5, since } Z^TZ \text{ is} \\
&\qquad \text{symmetric positive semidefinite} \\
&= \|AQ-QT\|_2^2 \quad \text{by Part 7 of Lemma 1.7.}
\end{aligned}
$$

Restoring subscripts, it is easy to compute the minimum value

$$\|AQ_k - Q_kT_k\|_2 = \|(Q_kT_k + Q_uT_{ku}) - (Q_kT_k)\|_2 = \|Q_uT_{ku}\|_2 = \|T_{ku}\|_2.$$

If we replace Q_k with any product Q_kU, where U is another orthogonal matrix, then the columns of Q_k and Q_kU span the same space, and

$$\|AQ_k - Q_kR\|_2 = \|AQ_kU - Q_kRU\|_2 = \|A(Q_kU) - (Q_kU)(U^TRU)\|_2.$$

These quantities are still minimized when $R = T_k$, and by choosing $U = V$ so that U^TT_kU is diagonal, we solve the second minimization problem in the statement of the theorem. □

This theorem justifies using Ritz values as eigenvalue approximations. When Q_k is computed by the Lanczos algorithm, in which case (see equation (6.31))

$$
T = \left[\begin{array}{c|c} T_k & T_{ku}^T \\ \hline T_{ku} & T_u \end{array}\right] = \left[\begin{array}{ccccc|ccccc}
\alpha_1 & \beta_1 & & & & & & & & \\
\beta_1 & \ddots & \ddots & & & & & & & \\
 & \ddots & \ddots & \beta_{k-1} & & & & & & \\
 & & \beta_{k-1} & \alpha_k & \beta_k & & & & & \\ \hline
 & & & \beta_k & \alpha_{k+1} & \beta_{k+1} & & & & \\
 & & & & \beta_{k+1} & \ddots & \ddots & & & \\
 & & & & & \ddots & \ddots & \beta_{n-1} & & \\
 & & & & & & \beta_{n-1} & \alpha_n &
\end{array}\right],
$$

then it is easy to compute all the quantities in Theorem 7.1. This is because there are good algorithms for finding eigenvalues and eigenvectors of the symmetric tridiagonal matrix T_k (see section 5.3) and because the residual norm is simply $\|T_{ku}\|_2 = \beta_k$. (From the Lanczos algorithm we know that β_k is nonnegative.) This simplifies the error bounds on the approximate eigenvalues and eigenvectors in the following theorem.

THEOREM 7.2. *Let T_k, T_{ku}, and Q_k be as in equation* (7.1). *Let $T_k = V\Lambda V^T$ be the eigendecomposition of T_k, where $V = [v_1, \ldots, v_k]$ is orthogonal and $\Lambda = \operatorname{diag}(\theta_1, \ldots, \theta_k)$. Then*

1. *There are k eigenvalues $\alpha_1, \ldots, \alpha_k$ of A (not necessarily the largest k) such that $|\theta_i - \alpha_i| \leq \|T_{ku}\|_2$ for $i = 1, \ldots, k$. If Q_k is computed by the Lanczos algorithm, then $|\theta_i - \alpha_i| \leq \|T_{ku}\|_2 = \beta_k$, where β_k is the single (possibly) nonzero entry in the upper right corner of T_{ku}.*

2. *$\|A(Q_k v_i) - (Q_k v_i)\theta_i\|_2 = \|T_{ku} v_i\|_2$. Thus, the difference between the Ritz value θ_i and some eigenvalue α of A is at most $\|T_{ku} v_i\|_2$, which may be much smaller than $\|T_{ku}\|_2$. If Q_k is computed by the Lanczos algorithm, then $\|T_{ku} v_i\|_2 = \beta_k |v_i(k)|$, where $v_i(k)$ is the kth (bottom) entry of v_i. This formula lets us compute the residual $\|A(Q_k v_i) - (Q_k v_i)\theta_i\|_2$ cheaply, i.e., without multiplying any vector by Q_k or by A.*

3. *Without any further information about the spectrum of T_u, we cannot deduce any useful error bound on the Ritz vector $Q_k v_i$. If we know that the gap between θ_i and any other eigenvalue of T_k or T_u is at least g, then we can bound the angle θ between $Q_k v_i$ and a true eigenvector of A by*

$$\frac{1}{2} \sin 2\theta \leq \frac{\|T_{ku}\|_2}{g}. \tag{7.2}$$

If Q_k is computed by the Lanczos algorithm, then the bound simplifies to

$$\frac{1}{2} \sin 2\theta \leq \frac{\beta_k}{g}.$$

Proof.

1. The eigenvalues of $\hat{T} = \begin{bmatrix} T_k & 0 \\ 0 & T_u \end{bmatrix}$ include θ_1 through θ_k. Since

$$\|\hat{T} - T\|_2 = \left\| \begin{bmatrix} 0 & T_{ku}^T \\ T_{ku} & 0 \end{bmatrix} \right\|_2 = \|T_{ku}\|_2,$$

Weyl's theorem, Theorem 5.1, tells us that the eigenvalues of $\hat{T}$ and T differ by at most $\|T_{ku}\|_2$. But the eigenvalues of T and A are identical, proving the result.

2. We compute

$$\begin{aligned} \|A(Q_k v_i) - (Q_k v_i)\theta_i\|_2 &= \|Q^T A(Q_k v_i) - Q^T (Q_k v_i)\theta_i\|_2 \\ &= \left\| \begin{bmatrix} T_k v_i \\ T_{ku} v_i \end{bmatrix} - \begin{bmatrix} v_i \theta_i \\ 0 \end{bmatrix} \right\|_2 = \left\| \begin{bmatrix} 0 \\ T_{ku} v_i \end{bmatrix} \right\|_2 \\ &\quad \text{since } T_k v_i = \theta_i v_i \\ &= \|T_{ku} v_i\|_2. \end{aligned}$$

Then by Theorem 5.5, A has some eigenvalue α satisfying $|\alpha - \theta_i| \leq \|T_{ku} v_i\|_2$. If Q_k is computed by the Lanczos algorithm, then $\|T_{ku} v_i\|_2 = \beta_k |v_i(k)|$, because only the top right entry of T_{ku}, namely, β_k, is nonzero.

3. We reuse Example 5.4 to show that we cannot deduce a useful error bound on the Ritz vector without further information about the spectrum of T_u:

$$T = \begin{bmatrix} 1+g & \epsilon \\ \epsilon & 1 \end{bmatrix},$$

where $0 < \epsilon < g$. We let $k = 1$ and $Q_1 = [e_1]$, so $T_1 = 1 + g$ and the approximate eigenvector is simply e_1. But as shown in Example 5.4, the eigenvectors of T are close to $[1, \epsilon/g]^T$ and $[-\epsilon/g, 1]^T$. So without a lower bound on g, i.e., the gap between the eigenvalue of T_k and all the other eigenvalues, including those of T_u, we cannot bound the error in the computed eigenvector. If we do have such a lower bound, we can apply the second bound of Theorem 5.4 to T and $T + E = \text{diag}(T_k, T_u)$ to derive equation (7.2). $\diamond$

7.3. The Lanczos Algorithm in Exact Arithmetic

The Lanczos algorithm for finding eigenvalues of a symmetric matrix A combines the Lanczos algorithm for building a Krylov subspace (Algorithm 6.10) with the Rayleigh–Ritz procedure of the last section. In other words, it builds an orthogonal matrix $Q_k = [q_1, \ldots, q_k]$ of orthogonal Lanczos vectors and approximates the eigenvalues of A by the Ritz values (the eigenvalues of the symmetric tridiagonal matrix $T_k = Q_k^T A Q_k$), as in equation (7.1).

ALGORITHM 7.1. *Lanczos algorithm in exact arithmetic for finding eigenvalues and eigenvectors of* $A = A^T$*:*

$q_1 = b/\|b\|_2$, $\beta_0 = 0$, $q_0 = 0$
for $j = 1$ *to* k
  $z = Aq_j$
  $\alpha_j = q_j^T z$
  $z = z - \alpha_j q_j - \beta_{j-1} q_{j-1}$
  $\beta_j = \|z\|_2$
  if $\beta_j = 0$, *quit*
  $q_{j+1} = z/\beta_j$
  Compute eigenvalues, eigenvectors, and error bounds of T_j
end for

In this section we explore the convergence of the Lanczos algorithm by describing a numerical example in some detail. This example has been chosen to illustrate both typical convergence behavior, as well as some more problematic behavior, which we call *misconvergence*. Misconvergence can occur because the starting vector q_1 is nearly orthogonal to the eigenvector of the desired eigenvalue or when there are multiple (or very close) eigenvalues.

The title of this section indicates that we have (nearly) eliminated the effects of roundoff error on our example. Of course, the Matlab code (HOMEPAGE/Matlab/LanczosFullReorthog.m) used to produce the example below ran in floating point arithmetic, but we implemented the Lanczos algorithm (in particular the inner loop of Algorithm 7.1) in a particularly careful and expensive way in order to make it mimic the exact result as closely as possible. This careful implementation is called *Lanczos with full reorthogonalization*, as indicated in the titles of the figures below.

In the next section we will explore the same numerical example using the original, inexpensive implementation of Algorithm 7.1, which we call *Lanczos with no reorthogonalization* in order to contrast it with *Lanczos with full reorthogonalization.* (We will also explain the difference in the two implementations.) We will see that the original Lanczos algorithm can behave significantly differently from the more expensive "exact" algorithm. Nevertheless, we will show how to use the less expensive algorithm to compute eigenvalues reliably.

EXAMPLE 7.1. We illustrate the Lanczos algorithm and its error bounds by running a large example, a 1000-by-1000 diagonal matrix A, most of whose eigenvalues were chosen randomly from a normal Gaussian distribution. Figure 7.1 is a plot of the eigenvalues. To make later plots easy to understand, we have also sorted the diagonal entries of A from largest to smallest, so $\lambda_i(A) = a_{ii}$, with corresponding eigenvector e_i, the ith column of the identity matrix. There are a few extreme eigenvalues, and the rest cluster near the center of the spectrum. The starting Lanczos vector q_1 has all equal entries, except for one, as described below.

There is no loss in generality in experimenting with a diagonal matrix, since running the Lanczos algorithm on A with starting vector q_1 is equivalent to running the Lanczos algorithm on $Q^T A Q$ with starting vector $Q^T q_1$ (see Question 7.1).

To illustrate convergence, we will use several plots of the sort shown in Figure 7.2. In this figure the eigenvalues of each T_k are shown plotted in column k, for $k = 1$ to 9 on the top, and for $k = 1$ to 29 on the bottom, with the eigenvalues of A plotted in an extra column at the right. Thus, column k has k pluses, one marking each eigenvalue of T_k. We have also color-coded the eigenvalues as follows: The largest and smallest eigenvalues of each T_k are shown in black, the second largest and second smallest eigenvalues are red, the third largest and third smallest eigenvalues are green, and the fourth largest and fourth smallest eigenvalues are blue. Then these colors recycle into the interior of the spectrum.

To understand convergence, consider the largest eigenvalue of each T_k; these black pluses are on the top of each column. Note that they increase monotonically as k increases; this is a consequence of the Cauchy interlace theorem, since T_k is a submatrix of T_{k+1} (see Question 5.4). In fact, the Cauchy interlace theorem tells us more, that the eigenvalues of T_k *interlace* those of T_{k+1},

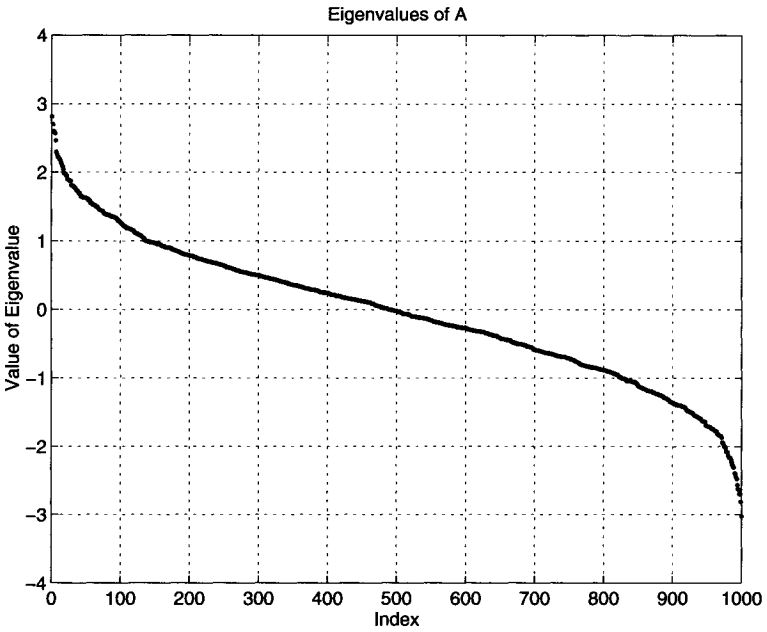

Fig. 7.1. *Eigenvalues of the diagonal matrix A.*

or that $\lambda_i(T_{k+1}) \geq \lambda_i(T_k) \geq \lambda_{i+1}(T_{k+1}) \geq \lambda_{i+1}(T_k)$. In other words, $\lambda_i(T_k)$ increases monotonically with k for any fixed i, not just $i = 1$ (the largest eigenvalue). This is illustrated by the colored sequences of pluses moving right and up in the figure.

A completely analogous phenomenon occurs with the smallest eigenvalues: The bottom black plus sign in each column of Figure 7.2 shows the smallest eigenvalue of each T_k, and these are monotonically decreasing as k increases. Similarly, the ith smallest eigenvalue is also monotonically decreasing. This is also a simple consequence of the Cauchy interlace theorem.

Now we can ask to which eigenvalue of A the eigenvalue $\lambda_i(T_k)$ can converge as k increases. Clearly the largest eigenvalue of T_k, $\lambda_1(T_k)$, ought to converge to the largest eigenvalue of A, $\lambda_1(A)$. Indeed, if the Lanczos algorithm proceeds to step $k = n$ (without quitting early because some $\beta_k = 0$), then T_n and A are similar, and so $\lambda_1(T_n) = \lambda_1(A)$. Similarly, the ith largest eigenvalue $\lambda_i(T_k)$ of T_k must increase monotonically and converge to the ith largest eigenvalue $\lambda_i(A)$ of A (provided that the Lanczos algorithm does not quit early). And the ith smallest eigenvalue $\lambda_{k+1-i}(T_k)$ of T_k must similarly decrease monotonically and converge to the ith smallest eigenvalue $\lambda_{n+1-i}(A)$ of A.

All these converging sequences are represented by sequences of pluses of a common color in Figure 7.2 and other figures in this section. Consider the bottom graph in Figure 7.2: For k larger than about 15, the topmost and bottom-most black pluses form horizontal rows next to the extreme eigenvalues of A, which are plotted in the rightmost column; this demonstrates conver-

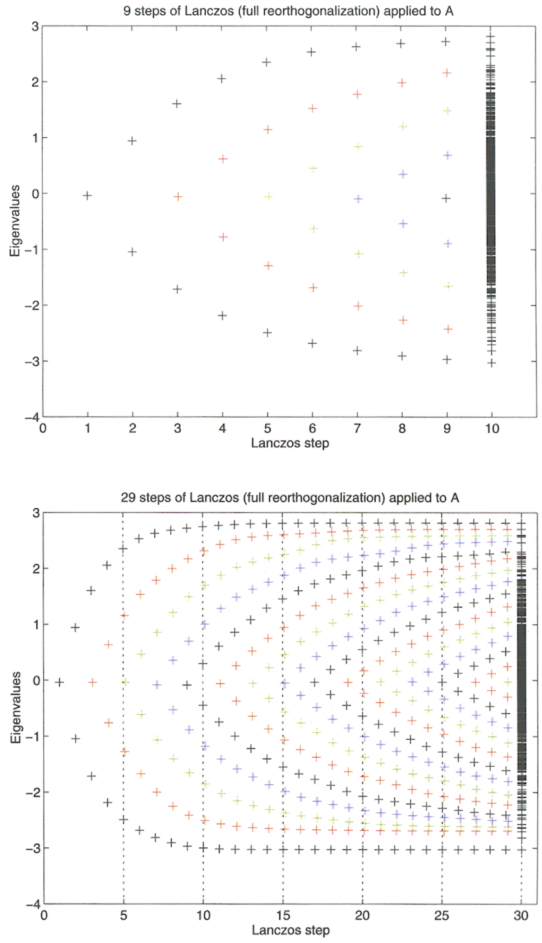

Fig. 7.2. *The Lanczos algorithm applied to A. The first* 9 *steps are shown on the top, and the first* 29 *steps are shown on the bottom. Column k shows the eigenvalues of* T_k*, except that the rightmost columns (column* 10 *on the top and column* 30 *on the bottom) show all the eigenvalues of A.*

gence. Similarly, the top sequence of red pluses forms a horizontal row next to the second largest eigenvalue of A in the rightmost column; they converge later than the outermost eigenvalues. A blow-up of this behavior for more Lanczos algorithm steps is shown in the top two graphs of Figure 7.3.

To summarize the above discussion, *extreme eigenvalues, i.e., the largest and smallest ones, converge first, and the interior eigenvalues converge last. Furthermore, convergence is monotonic, with the ith largest (smallest) eigenvalue of* T_k *increasing (decreasing) to the ith largest (smallest) eigenvalue of A, provided that the Lanczos algorithm does not stop prematurely with some* $\beta_k = 0$.

Now we examine the convergence behavior in more detail, compute the actual errors in the Ritz values, and compare these errors with the error bounds

in part 2 of Theorem 7.2. We run the Lanczos algorithm for 99 steps on the same matrix pictured in Figure 7.2 and display the results in Figure 7.3. The top left graph in Figure 7.3 shows only the largest eigenvalues, and the top right graph shows only the smallest eigenvalues.

The middle two graphs in Figure 7.3 show the errors in the four largest computed eigenvalues (on the left) and the four smallest computed eigenvalues (on the right). The colors in the middle graphs match the colors in the top graphs. We measure and plot the errors in three ways:

- The *global errors* (the solid lines) are given by $|\lambda_i(T_k) - \lambda_i(A)|/|\lambda_i(A)|$. We divide by $|\lambda_i(A)|$ in order to normalize all the errors to lie between 1 (no accuracy) and about 10^{-16} (machine epsilon, or full accuracy). As k increases, the global error decreases monotonically, and we expect it to decrease to machine epsilon, unless the Lanczos algorithm quits prematurely.

- The *local errors* (the dotted lines) are given by $\min_j |\lambda_i(T_k) - \lambda_j(A)|/|\lambda_i(A)|$. The local error measures the smallest distance between $\lambda_i(T_k)$ and the *nearest* eigenvalue $\lambda_j(A)$ of A, not just the ultimate value $\lambda_i(A)$. We plot this because sometimes the local error is much smaller than the global error.

- The *error bounds* (the dashed lines) are the quantities $|\beta_k v_i(k)|/|\lambda_i(A)|$ computed by the algorithm (except for the normalization by $|\lambda_i(A)|$, which of course the algorithm does not know!).

The bottom two graphs in Figure 7.3 show the eigenvector components of the Lanczos vectors q_k for the four eigenvectors corresponding to the four largest eigenvalues (on the left) and for the four eigenvectors corresponding to the four smallest eigenvalues (on the right). In other words, they plot $q_k^T e_j = q_k(j)$, where e_j is the jth eigenvector of the diagonal matrix A, for $k = 1$ to 99 and for $j = 1$ to 4 (on the left) and $j = 997$ to 1000 (on the right). The components are plotted on a logarithmic scale, with "+" and "o" to indicate whether the component is positive or negative, respectively. We use these plots to help explain convergence below.

Now we use Figure 7.3 to examine convergence in more detail. The largest eigenvalue of T_k (topmost black pluses in the top left graph of Figure 7.3) begins converging to its final value (about 2.81) right away, is correct to six decimal places after 25 Lanczos steps, and is correct to machine precision by step 50. The global error is shown by the solid black line in the middle left graph. The local error (the dotted black line) is the same as the global error after not too many steps, although it can be "accidentally" much smaller if an eigenvalue $\lambda_i(T_k)$ happens to fall close to some other $\lambda_j(A)$ on its way to $\lambda_i(A)$. The dashed black line in the same graph is the relative error bound computed by the algorithm, which overestimates the true error up to about

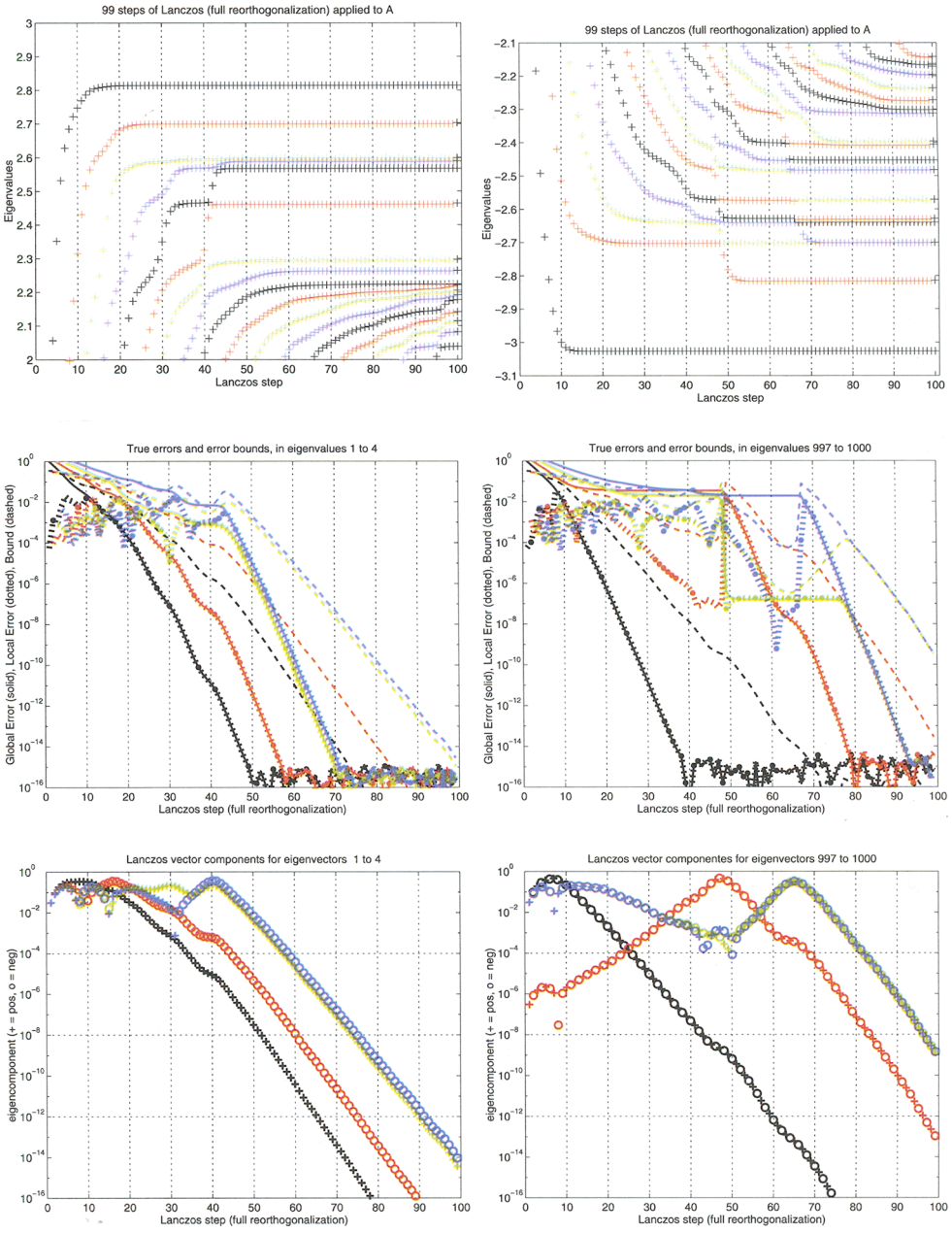

Fig. 7.3. *99 steps of the Lanczos algorithm applied to A. The largest eigenvalues are shown on the left, and the smallest on the right. The top two graphs show the eigenvalues themselves, the middle two graphs the errors (global = solid, local = dotted, bounds = dashed), and the bottom two graphs show eigencomponents of Lanczos vectors. The colors in a column of three graphs match.*

step 75. Still, the relative error bound correctly indicates that the largest eigenvalue is correct to several decimal digits.

The second through fourth largest eigenvalues (the topmost red, green and blue pluses in the top left graph of Figure 7.3) converge in a similar fashion, with eigenvalue i converging slightly faster than eigenvalue $i+1$. This is typical behavior of the Lanczos algorithm.

The bottom left graph of Figure 7.3 measures convergence in terms of the eigenvector components $q_k^T e_j$. To explain this graph, consider what happens to the Lanczos vectors q_k as the first eigenvalue converges. Convergence means that the corresponding eigenvector e_1 nearly lies in the Krylov subspace spanned by the Lanczos vectors. In particular, since the first eigenvalue has converged after $k = 50$ Lanczos steps, this means that e_1 must very nearly be a linear combination of q_1 through q_{50}. Since the q_k are mutually orthogonal, this means q_k must also be orthogonal to e_1 for $k > 50$. This is borne out by the black curve in the bottom left graph, which has decreased to less than 10^{-7} by step 50. The red curve is the component of e_2 in q_k, and this reaches 10^{-8} by step 60. The green curve (third eigencomponent) and blue curve (fourth eigencomponent) get comparably small a few steps later.

Now we discuss the smallest four eigenvalues, whose behavior is described by the three graphs on the right of Figure 7.3. We have chosen the matrix A and starting vector q_1 to illustrate certain difficulties that can arise in the convergence of the Lanczos algorithm to show that convergence is not always as straightforward as in the case of the four eigenvalues just examined.

In particular, we have chosen $q_1(999)$, the eigencomponent of q_1 in the direction of the second smallest eigenvalue (-2.81), to be about 10^{-7}, which is 10^5 times smaller than all the other components of q_1, which are equal. Also, we have chosen the third and fourth smallest eigenvalues (numbers 998 and 997) to be nearly the same: -2.700001 and -2.7.

The convergence of the smallest eigenvalue of T_k to $\lambda_{1000}(A) \approx -3.03$ is uneventful, similar to the largest eigenvalues. It is correct to 16 digits by step 40.

The *second* smallest eigenvalue of T_k, shown in red, begins by *misconverging* to the *third* smallest eigenvalue of A, near -2.7. Indeed, the dotted red line in the middle right graph of Figure 7.3 shows that $\lambda_{999}(T_k)$ agrees with $\lambda_{998}(A)$ to six decimal places for Lanczos steps $40 < k < 50$. The corresponding error bound (the red dashed line) tells us that $\lambda_{999}(T_k)$ equals *some* eigenvalue of A to three or four decimal places for the same values of k. The reason $\lambda_{999}(T_k)$ misconverges is that the Krylov subspace starts with a very small component of the corresponding Krylov subspace e_{999}, namely, 10^{-7}. This can be seen by the red curve in bottom right graph, which starts at 10^{-7} and takes until step 45 before a large component of e_{999} appears. Only at this point, when the Krylov subspace contains a sufficiently large component of the eigenvector e_{999}, can $\lambda_{999}(T_k)$ start converging again to its final value $\lambda_{999}(A) \approx -2.81$, as shown in the top and middle right graphs. Once this convergence has set in again,

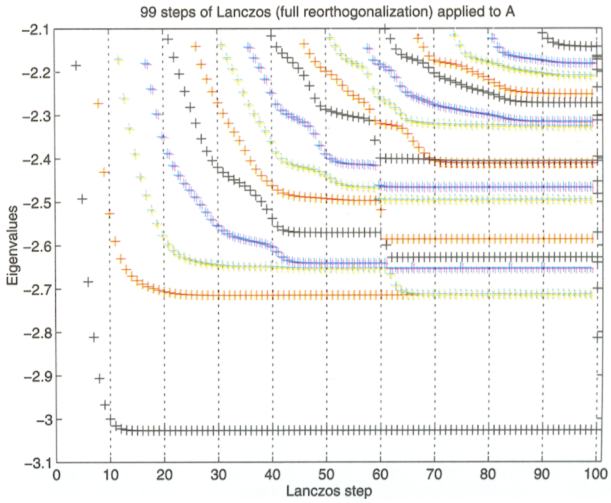

Fig. 7.4. *The Lanczos algorithm applied to A, where the starting vector q_1 is orthogonal to the eigenvector corresponding to the second smallest eigenvalue -2.81. No approximation to this eigenvalue is computed.*

the component of e_{999} starts decreasing again and becomes very small once $\lambda_{999}(T_k)$ has converged to $\lambda_{999}(A)$ sufficiently accurately. (For a quantitative relationship between the convergence rate and the eigencomponent $q_1^T e_{999}$, see the theorem of Kaniel and Saad discussed below.)

Indeed, if q_1 were *exactly* orthogonal to e_{999}, so $q_1^T e_{999} = 0$ rather than just $q_1^T e_{999} = 10^{-7}$, then all later Lanczos vectors would also be orthogonal to e_{999}. This means $\lambda_{999}(T_k)$ would never converge to $\lambda_{999}(A)$. (For a proof, see Question 7.3.) We illustrate this in Figure 7.4, where we have modified q_1 just slightly so that $q_1^T e_{999} = 0$. Note that no approximation to $\lambda_{999}(A) \approx -2.81$ ever appears.

Fortunately, if we choose q_1 at random, it is extremely unlikely to be orthogonal to an eigenvector. We can always rerun the Lanczos algorithm with a different random q_1 to provide more "statistical" evidence that we have not missed any eigenvalues.

Another source of "misconvergence" are (nearly) multiple eigenvalues, such as the the third smallest eigenvalue $\lambda_{998}(A) = -2.700001$ and the fourth smallest eigenvalue $\lambda_{997}(A) = -2.7$. By examining $\lambda_{998}(T_k)$, the bottommost green curve in the top right and middle right graphs of Figure 7.3, we see that during Lanczos steps $50 < k < 75$, $\lambda_{998}(T_k)$ *misconverges* to about -2.7000005, *halfway* between the two closest eigenvalues of A. This is not visible at the resolution provided by the top right graph but is evident from the horizontal segment of the solid green line in the middle right graph during Lanczos steps $50 < k < 75$. At step 76 rapid convergence to the final value $\lambda_{998}(A) = -2.700001$ sets in again.

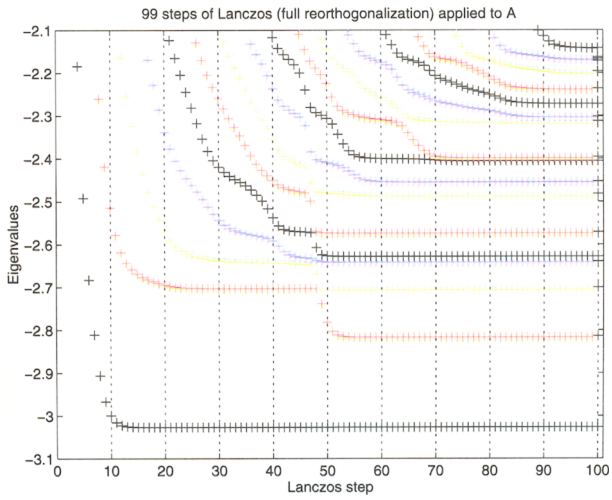

Fig. 7.5. *The Lanczos algorithm applied to A, where the third and fourth smallest eigenvalues are equal. Only one approximation to this double eigenvalue is computed.*

Meanwhile, the fourth smallest eigenvalue $\lambda_{997}(T_k)$, shown in blue, has misconverged to a value near $\lambda_{996}(A) \approx -2.64$; the blue dotted line in the middle right graph indicates that $\lambda_{997}(T_k)$ and $\lambda_{996}(A)$ agree to up to nine decimal places near step $k = 61$. At step $k = 65$ rapid convergence sets in again to the final value $\lambda_{997}(A) = -2.7$. This can also be seen in the bottom right graph, where the eigenvector components of e_{997} and e_{998} grow again during step $50 < k < 65$, after which rapid convergence sets in and they again decrease.

Indeed, if $\lambda_{997}(A)$ were *exactly* a double eigenvalue, we claim that T_k would never have two eigenvalues near that value but only one (in exact arithmetic). (For a proof, see Question 7.3.) We illustrate this in Figure 7.5, where we have modified A just slightly so that it has two eigenvalues exactly equal to -2.7. Note that only one approximation to $\lambda_{998}(A) = \lambda_{997}(A) = -2.7$ ever appears.

Fortunately, there are many applications where it is sufficient to find one copy of each eigenvalue rather than all multiple copies. Also, it is possible to use "block Lanczos" to recover multiple eigenvalues (see the algorithms cited in section 7.6).

Examining other eigenvalues in the top right graph of Figure 7.3, we see that misconvergence is quite common, as indicated by the frequent short horizontal segments of like-colored pluses, which then drop off to the right to the next smaller eigenvalue. For example, the seventh smallest eigenvalue is well-approximated by the fifth (black), sixth (red), and seventh (green) smallest eigenvalues of T_k at various Lanczos steps.

These misconvergence phenomena explain why the computable error bound provided by part 2 of Theorem 7.2 is essential to monitor convergence [198]. If the error bound is small, the computed eigenvalue is indeed a good approx-

imation to some eigenvalue, even if one is "missing." ◇

There is another error bound, due to Kaniel and Saad, that sheds light on why misconvergence occurs. This error bound depends on the angle between the starting vector q_1 and the desired eigenvectors, the Ritz values, and the desired eigenvalues. In other words, it depends on quantities unknown during the computation, so it is not of practical use. But it shows that if q_1 is nearly orthogonal to the desired eigenvector, or if the desired eigenvalue is nearly multiple, then we can expect slow convergence. See [197, sect. 12-4] for details.

7.4. The Lanczos Algorithm in Floating Point Arithmetic

The example in the last section described the behavior of the "ideal" Lanczos algorithm, essentially without roundoff. We call the corresponding careful but expensive implementation of Algorithm 6.10 *Lanczos with full reorthogonalization* to contrast it with the original inexpensive implementation, which we call *Lanczos with no reorthogonalization* (HOMEPAGE/Matlab/LanczosNoReorthog.m). Both algorithms are shown below.

ALGORITHM 7.2. *Lanczos algorithm with full or no reorthogonalization for finding eigenvalues and eigenvectors of* $A = A^T$*:*

$q_1 = b/\|b\|_2$, $\beta_0 = 0$, $q_0 = 0$
for $j = 1$ *to* k
  $z = Aq_j$
  $\alpha_j = q_j^T z$
  $\begin{cases} z = z - \sum_{i=1}^{j-1}(z^T q_i)q_i, & z = z - \sum_{i=1}^{j-1}(z^T q_i)q_i \quad \textit{full reorthogonalization} \\ z = z - \alpha_j q_j - \beta_{j-1}q_{j-1} & \textit{no reorthogonalization} \end{cases}$
  $\beta_j = \|z\|_2$
  if $\beta_j = 0$, *quit*
  $q_{j+1} = z/\beta_j$
  Compute eigenvalues, eigenvectors, and error bounds of T_k
end for

Full reorthogonalization corresponds to applying the Gram–Schmidt orthogonalization process "$z = z - \sum_{i=1}^{j-1}(z^T q_i)q_i$" *twice* in order to almost surely make z orthogonal to q_1 through q_{j-1}. (See Algorithm 3.1 as well as [197, sect. 6-9] and [171, chap. 7] for discussions of when "twice is enough.") In exact arithmetic, we showed in section 6.6.1 that z is orthogonal to q_1 through q_{j-1} without reorthogonalization. Unfortunately, we will see that roundoff destroys this orthogonality property, upon which all of our analysis has depended so far.

This loss of orthogonality does not cause the algorithm to behave completely unpredictably. Indeed, we will see that the price we pay is to get

multiple copies of converged Ritz values. In other words, instead of T_k having one eigenvalue nearly equal to $\lambda_i(A)$ for k large, it may have many eigenvalues nearly equal to $\lambda_i(A)$. This is not a disaster if one is not concerned about computing multiplicities of eigenvalues and does not mind the resulting delayed convergence of interior eigenvalues. See [57] for a detailed description of a Lanczos implementation that operates in this fashion, and NETLIB/lanczos for the software itself. (This software has heuristics for estimating multiplicities of eigenvalues.)

But if accurate multiplicities are important, then one needs to keep the Lanczos vectors (nearly) orthogonal. So one could use the Lanczos algorithm with full reorthogonalization, as we did in the last section. But one can easily confirm that this costs $O(k^2n)$ flops instead of $O(kn)$ flops for k steps, and $O(kn)$ space instead of $O(n)$ space, which may be too high a price to pay.

Fortunately, there is a middle ground between no reorthogonalization and full reorthogonalization, which nearly gets the best of both worlds. It turns out that the q_k lose their orthogonality in a very systematic way by developing large components in the directions of already converged Ritz vectors. (This is what leads to multiple copies of converged Ritz values.) This systematic loss of orthogonality is illustrated by the next example and explained by Paige's theorem below. We will see that by monitoring the computed error bounds, we can conservatively predict which q_k will have large components of which Ritz vectors. Then we can *selectively orthogonalize* q_k against just those few prior Ritz vectors, rather than against all the earlier q_is at each step, as with full reorthogonalization. This keeps the Lanczos vectors (nearly) orthogonal for very little extra work. The next section discusses selective orthogonalization in detail.

EXAMPLE 7.2. Figure 7.7 shows the convergence behavior of 149 steps of Lanczos on the matrix in Example 7.1. The graphs on the right are with full reorthogonalization, and the graphs on the left are with no reorthogonalization. These graphs are similar to those in Figure 7.3, except that the global error is omitted, since this clutters the middle graphs.

Figure 7.6 plots the smallest singular value $\sigma_{\min}(Q_k)$ versus Lanczos step k. In exact arithmetic, Q_k is orthogonal and so $\sigma_{\min}(Q_k) = 1$. With roundoff, Q_k loses orthogonality starting at around step $k = 70$, and $\sigma_{\min}(Q_k)$ drops to .01 by step $k = 80$, which is where the top two graphs in Figure 7.7 begin to diverge visually.

In particular, starting at step $k = 80$ in the top left graph of Figure 7.7, the second smallest (red) eigenvalue $\lambda_2(T_k)$, which had converged to $\lambda_2(A) \approx 2.7$ to almost 16 digits, leaps up to $\lambda_1(A) \approx 2.81$ in just a few steps, yielding a "second copy" of $\lambda_1(A)$ along with $\lambda_1(T_k)$ (in black). (This may be hard to see, since the red pluses overwrite and so obscure the black pluses.) This transition can be seen in the leap in the dashed red error bound in the middle left graph. Also, this transition was "foreshadowed" by the increasing component of e_1

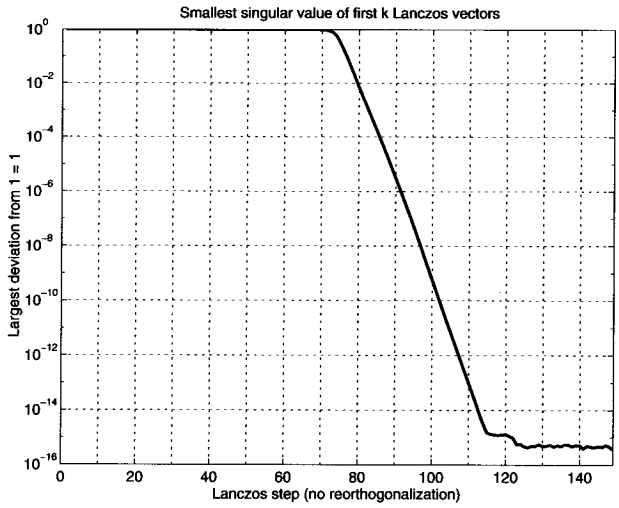

Fig. 7.6. *Lanczos algorithm without reorthogonalization applied to A. The smallest singular value* $\sigma_{\min}(Q_k)$ *of the Lanczos vector matrix* Q_k *is shown for* $k = 1$ *to* 149. *In the absence of roundoff,* Q_k *is orthogonal, and so all singular values should be one. With roundoff,* Q_k *becomes rank deficient.*

in the bottom left graph, where the black curve starts rising again at step $k = 50$ rather than continuing to decrease to machine epsilon, as it does with full reorthogonalization in the bottom right graph. Both of these indicate that the algorithm is diverging from its exact path (and that some selective orthogonalization is called for). After the second copy of $\lambda_1(A)$ has converged, the component of e_1 in the Lanczos vectors starts dropping again, starting a little after step $k = 80$.

Similarly, starting at about step $k = 95$, a second copy of $\lambda_2(A)$ appears when the blue curve ($\lambda_4(T_k)$) in the upper left graph moves from about $\lambda_3(A) \approx 2.6$ to $\lambda_2(A) \approx 2.7$. At this point we have two copies of $\lambda_1(A) \approx 2.81$ and two copies of $\lambda_2(A)$. This is a bit hard to see on the graphs, since the pluses of one color obscure the pluses of the other color (red overwrites black, and blue overwrites green). This transition is indicated by the dashed blue error bound for $\lambda_4(T_k)$ in the middle left graph rising sharply near $k = 95$ and is foreshadowed by the rising red curve in the bottom left graph, which indicates that the component of e_2 in the Lanczos vectors is rising. This component peaks near $k = 95$ and starts dropping again.

Finally, around step $k = 145$, a *third* copy of $\lambda_1(A)$ appears, again indicated and foreshadowed by changes in the two bottom left graphs. If we were to continue the Lanczos process, we would periodically get additional copies of many other converged Ritz values. ◇

The next theorem provides an explanation for the behavior seen in the above example, and hints at a practical criterion for selectively orthogonalizing Lanczos vectors. In order not to be overwhelmed by taking all possible roundoff

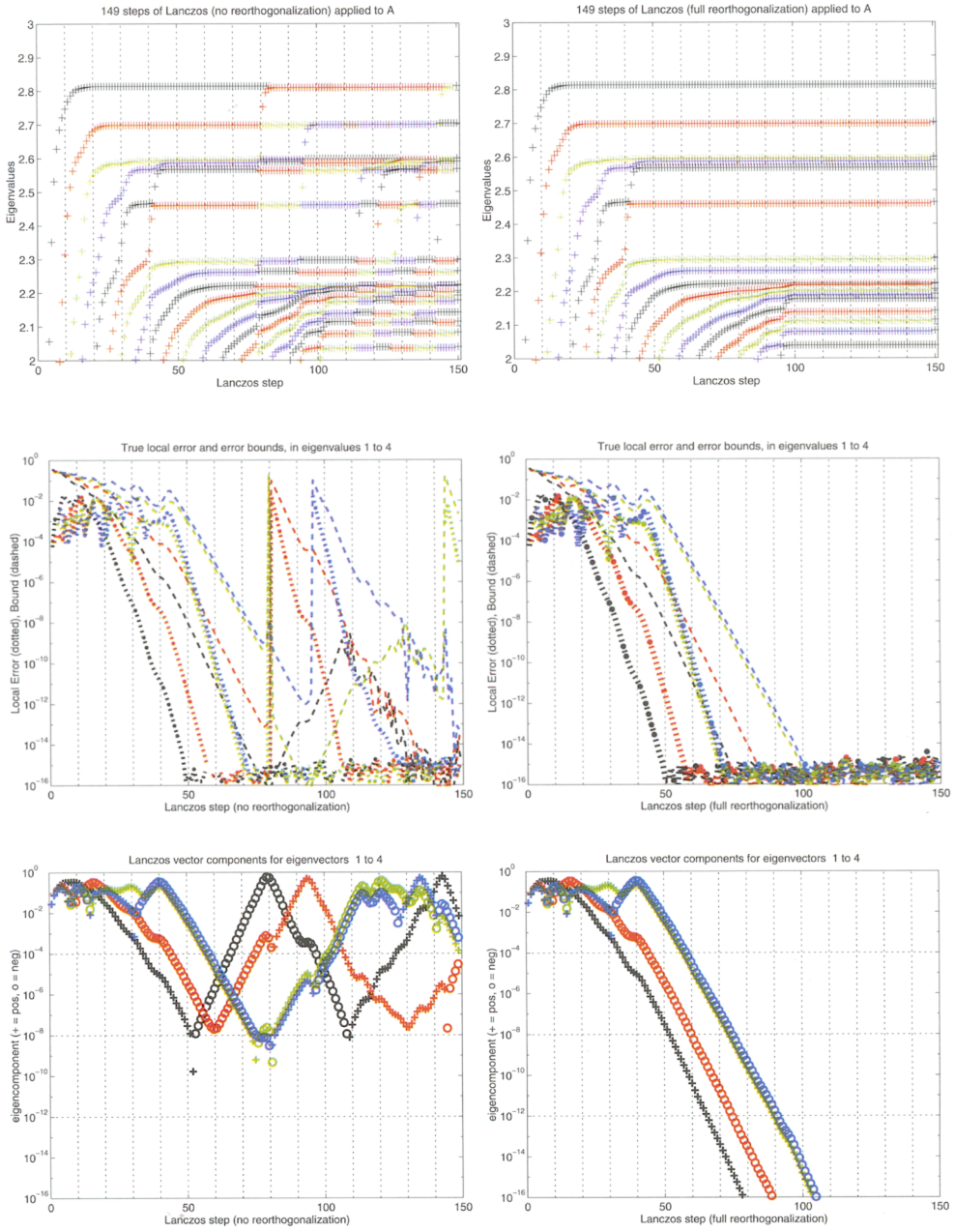

Fig. 7.7. 149 *steps of Lanczos algorithm applied to A. Column* 150 *(at the right of the top graphs) shows the eigenvalues of A. In the left graphs, no reorthogonalization is done. In the right graphs, full reorthogonalization is done.*

errors into account, we will draw on others' experience to identify those few rounding errors that are important, and simply ignore the rest [197, sect. 13-4]. This lets us summarize the Lanczos algorithm with no reorthogonalization in one line:

$$\beta_j q_{j+1} + f_j = Aq_j - \alpha_j q_j - \beta_{j-1} q_{j-1}. \tag{7.3}$$

In this equation the variables represent the values actually stored in the machine, except for f_j, which represents the roundoff error incurred by evaluating the right-hand side and then computing β_j and q_{j+1}. The norm $\|f_j\|_2$ is bounded by $O(\varepsilon\|A\|)$, where ε is machine epsilon, which is all we need to know about f_j. In addition, we will write $T_k = V\Lambda V^T$ *exactly*, since we know that the roundoff errors occurring in this eigendecomposition are not important. Thus, Q_k is not necessarily an orthogonal matrix, but V is.

THEOREM 7.3. Paige. *We use the notation and assumptions of the last paragraph. We also let* $Q_k = [q_1, \ldots, q_k]$, $V = [v_1, \ldots, v_k]$, *and* $\Lambda = \mathrm{diag}(\theta_1, \ldots, \theta_k)$. *We continue to call the columns* $y_{k,i} = Q_k v_i$ *of* $Q_k V$ *the Ritz vectors and the* θ_i *the Ritz values. Then*

$$y_{k,i}^T q_{k+1} = \frac{O(\varepsilon\|A\|)}{\beta_k |v_i(k)|}.$$

In other words the component $y_{k,i}^T q_{k+1}$ of the computed Lanczos vector q_{k+1} in the direction of the Ritz vector $y_{k,i} = Q_k v_i$ is proportional to the reciprocal of $\beta_k |v_i(k)|$, which is the error bound on the corresponding Ritz value θ_i (see Part 2 of Theorem 7.2). Thus, when the Ritz value θ_i converges and its error bound $\beta_k |v_i(k)|$ goes to zero, the Lanczos vector q_{k+1} acquires a large component in the direction of Ritz vector $y_{k,i}$. Thus, the Ritz vectors become linearly dependent, as seen in Example 7.2. Indeed, Figure 7.8 plots both the error bound $|\beta_k v_i(k)|/|\lambda_i(A)| \approx |\beta_k v_i(k)|/\|A\|$ and the Ritz vector component $y_{k,i}^T q_{k+1}$ for the largest Ritz value ($i = 1$, the top graph) and for the second largest Ritz value ($i = 2$, the bottom graph) of our 1000-by-1000 diagonal example. According to Paige's theorem, the product of these two quantities should be $O(\varepsilon)$. Indeed it is, as can be seen by the symmetry of the curves about the middle line $\sqrt{\varepsilon}$ of these semilogarithmic graphs.

Proof of Paige's theorem. We start with equation (7.3) for $j = 1$ to $j = k$, and write these k equations as the single equation

$$\begin{aligned} AQ_k &= Q_k T_k + [0, \ldots, 0, \beta_k q_{k+1}] + F_k \\ &= Q_k T_k + \beta_k q_{k+1} e_k^T + F_k, \end{aligned}$$

where e_k^T is the k-dimensional row vector $[0, \ldots, 0, 1]$ and $F_k = [f_1, \ldots, f_k]$ is the matrix of roundoff errors. We simplify notation by dropping the subscript k to get $AQ = QT + \beta q e^T + F$. Multiply on the left by Q^T to get $Q^T AQ =$

$Q^TQT + \beta Q^Tqe^T + Q^TF$. Since Q^TAQ is symmetric, we get that $Q^TQT + \beta Q^Tqe^T + Q^TF$ equals its transpose or, rearranging this equality,

$$0 = (Q^TQT - TQ^TQ) + \beta(Q^Tqe^T - eq^TQ) + (Q^TF - F^TQ). \tag{7.4}$$

If θ and v are a Ritz value and Ritz vector, respectively, so that $Tv = \theta v$, then note that

$$v^T\beta(eq^TQ)v = [\beta v(k)] \cdot [q^T(Qv)] \tag{7.5}$$

is the product of error bound $\beta v(k)$ and the Ritz vector component $q^T(Qv) = q^Ty$, which Paige's theorem says should be $O(\varepsilon\|A\|)$. Our goal is now to manipulate equation (7.4) to get an expression for eq^TQ alone, and then use equation (7.5).

To this end, we now invoke more simplifying assumptions about roundoff: Since each column of Q is gotten by dividing a vector z by its norm, the diagonal of Q^TQ is equal to 1 to full machine precision; we will suppose that it is exactly 1. Furthermore, the vector $z' = z - \alpha_j q_j = z - (q_j^Tz)q_j$ computed by the Lanczos algorithm is constructed to be orthogonal to q_j, so it is also true that q_{j+1} and q_j are orthogonal to nearly full machine precision. Thus $q_{j+1}^Tq_j = (Q^TQ)_{j+1,j} = O(\varepsilon)$; we will simply assume $(Q^TQ)_{j+1,j} = 0$. Now write $Q^TQ = I + C + C^T$, where C is lower triangular. Because of our assumptions about roundoff, C is in fact nonzero only on the second subdiagonal and below. This means

$$Q^TQT - TQ^TQ = (CT - TC) + (C^TT - TC^T),$$

where we can use the zero structures of C and T to easily show that $CT - TC$ is strictly lower triangular and $C^TT - TC^T$ is strictly upper triangular. Also, since e is nonzero only in its last entry, eq^TQ is nonzero only in the last row. Furthermore, the structure of Q^TQ just described implies that the last entry of the last row of eq^TQ is zero. So in particular, eq^TQ is also strictly lower triangular and Q^Tqe^T is strictly upper triangular. Applying the fact that eq^TQ and $CT - TC$ are both strictly lower triangular to equation (7.4) yields

$$0 = (CT - TC) - \beta eq^TQ + L, \tag{7.6}$$

where L is the strict lower triangle of $Q^TF - F^TQ$. Multiplying equation (7.6) on the left by v^T and on the right by v, using equation (7.5) and the fact that $v^T(CT - TC)v = v^TCv\theta - \theta v^TCv = 0$, yields

$$v^T\beta(eq^TQ)v = [\beta v(k)] \cdot [q^T(Qv)] = v^TLv.$$

Since $|v^TLv| \le \|L\| = O(\|Q^TF - F^TQ\|) = O(\|F\|) = O(\varepsilon\|A\|)$, we get

$$[\beta v(k)] \cdot [q^T(Qv)] = O(\varepsilon\|A\|),$$

which is equivalent to Paige's theorem. □

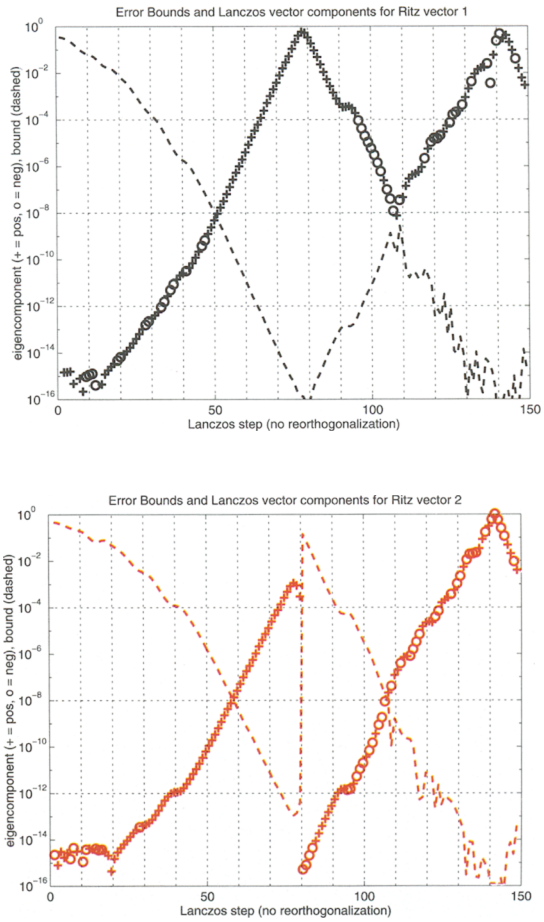

Fig. 7.8. *Lanczos with no reorthogonalization applied to A. The first* 149 *steps are shown for the largest eigenvalue (in black, at top) and for the second largest eigenvalue (in red, at bottom). The dashed lines are error bounds as before. The lines marked by pluses and o's show* $y_{k,i}^T q_{k+1}$, *the component of Lancos vector* $k+1$ *in the direction of the Ritz vector for the largest Ritz value* ($i = 1$, *at top) or for the second largest Ritz value* ($i = 2$, *at bottom).*

7.5. The Lanczos Algorithm with Selective Orthogonalization

We discuss a variation of the Lanczos algorithm which has (nearly) the high accuracy of the Lanczos algorithm with full reorthogonalization but (nearly) the low cost of the Lanczos algorithm with no reorthogonalization. This algorithm is called the Lanczos algorithm with *selective orthogonalization.* As discussed in the last section, our goal is to keep the computed Lanczos vectors q_k as nearly orthogonal as possible (for high accuracy) by orthogonalizing them against as few other vectors as possible at each step (for low cost). Paige's theorem (Theorem 7.3 in the last section) tells us that the q_k lose orthogonality because they acquire large components in the direction of Ritz vectors $y_{i,k} = Q_k v_i$ whose Ritz values θ_i have converged, as measured by the error bound $\beta_k |v_i(k)|$ becoming small. This phenomenon was illustrated in Example 7.2.

Thus, the simplest version of selective orthogonalization simply monitors the error bound $\beta_k |v_i(k)|$ at each step, and when it becomes small enough, the vector z in the inner loop of the Lanczos algorithm is orthogonalized against $y_{i,k}$: $z = z - (y_{i,k}^T z) y_{i,k}$. We consider $\beta_k |v_i(k)|$ to be small when it is less than $\sqrt{\varepsilon}\|A\|$, since Paige's theorem tells us that the vector component $|y_{i,k}^T q_{k+1}| = |y_{i,k}^T z / \|z\|_2|$ is then likely to exceed $\sqrt{\varepsilon}$. (In practice we may replace $\|A\|$ by $\|T_k\|$, since $\|T_k\|$ is known and $\|A\|$ may not be.) This leads to the following algorithm.

ALGORITHM 7.3. *The Lanczos algorithm with selective orthogonalization for finding eigenvalues and eigenvectors of* $A = A^T$*:*

$q_1 = b/\|b\|_2$, $\beta_0 = 0$, $q_0 = 0$
for $j = 1$ *to* k
  $z = Aq_j$
  $\alpha_j = q_j^T z$
  $z = z - \alpha_j q_j - \beta_{j-1} q_{j-1}$
  / Selectively orthogonalize against converged Ritz vectors */*
  for all $i \leq k$ *such that* $\beta_k |v_i(k)| \leq \sqrt{\varepsilon}\|T_k\|$
    $z = z - (y_{i,k}^T z) y_{i,k}$
  end for
  $\beta_j = \|z\|_2$
  if $\beta_j = 0$, *quit*
  $q_{j+1} = z/\beta_j$
  Compute eigenvalues, eigenvectors, and error bounds of T_k
end for

The following example shows what will happen to our earlier 1000-by-1000 diagonal matrix when this algorithm is used (HOMEPAGE/Matlab/LanczosSelectOrthog.m).

EXAMPLE 7.3. The behavior of the Lanczos algorithm with selective orthogonalization is visually indistinguishable from the behavior of the Lanczos algorithm with full orthogonalization shown in the three graphs on the right of Figure 7.7. In other words, selective orthogonalization provided as much accuracy as full orthogonalization.

The smallest singular values of all the Q_k were greater than $1-10^{-8}$, which means that selective orthogonalization did keep the Lanczos vectors orthogonal to about half precision, as desired.

Figure 7.9 shows the Ritz values corresponding to the Ritz vectors selected for reorthogonalization. Since the selected Ritz vectors correspond to converged Ritz values and the largest and smallest Ritz values converge first, there are two graphs: the large converged Ritz values are at the top, and the small converged Ritz values are at the bottom. The top graph matches the Ritz values shown in the upper right graph in Figure 7.7 that have converged to at least half precision. All together, 1485 Ritz vectors were selected for orthogonalization of a total possible 149*150/2 = 11175. Thus, selective orthogonalization did only 1485/11175 $\approx$ 13% as much work reorthogonalizing to keep the Lanczos vectors (nearly) orthogonal as full reorthogonalization.

Figure 7.10 shows how the Lanczos algorithm with selective reorthogonalization keeps the Lanczos vectors orthogonal just to the Ritz vectors for the largest two Ritz values. The graph at the top is a superposition of the two graphs in Figure 7.8, which show the error bounds and Ritz vectors components for the Lanczos algorithm with no reorthogonalization. The graph at the bottom is the corresponding graph for the Lanczos algorithm with selective orthogonalization. Note that at step $k = 50$, the error bound for the largest eigenvalue (the dashed black line) has reached the threshold of $\sqrt{\varepsilon}$. The Ritz vector is selected for orthogonalization (as shown by the top black pluses in the top of Figure 7.9), and the component in this Ritz vector direction disappears from the bottom graph of Figure 7.10. A few steps later, at $k = 58$, the error bound for the second largest Ritz value reaches $\sqrt{\varepsilon}$, and it too is selected for orthogonalization. The error bounds in the bottom graph continue to decrease to machine epsilon ε and stay there, whereas the error bounds in the top graph eventually grow again. $\diamond$

7.6. Beyond Selective Orthogonalization

Selective orthogonalization is not the end of the story, because the symmetric Lanczos algorithm can be made even less expensive. It turns out that once a Lanczos vector has been orthogonalized against a particular Ritz vector y, it takes many steps before the Lanczos vector again requires orthogonalization against y. So much of the orthogonalization work in Algorithm 7.3 can be eliminated. Indeed, there is a simple and inexpensive recurrence for deciding when to reorthogonalize [224, 192]. Another enhancement is to use the error bounds to efficiently distinguish between converged and "misconverged" eigen-

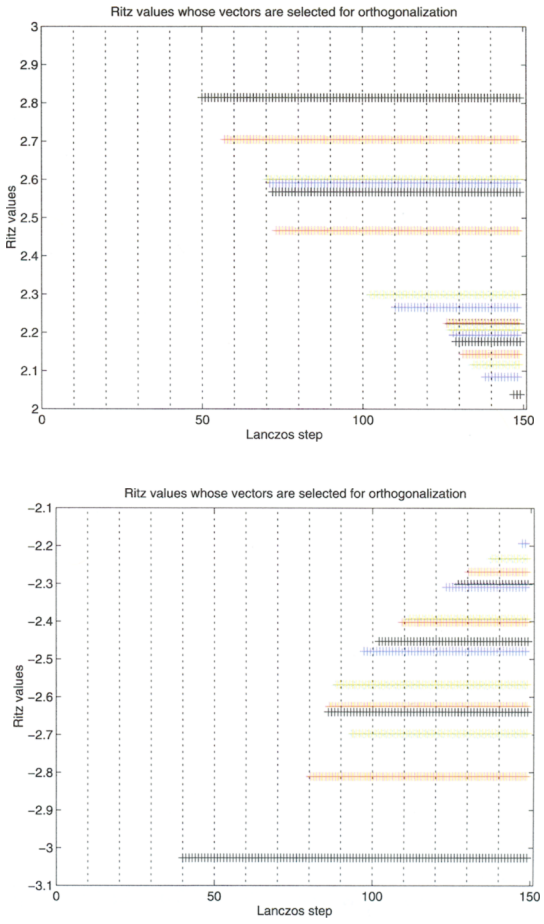

Fig. 7.9. *The Lanczos algorithm with selective orthogonalization applied to A. The Ritz values whose Ritz vectors are selected for orthogonalization are shown.*

values [198]. A state-of-the-art implementation of the Lanczos algorithm is described in [125]. A different software implementation is available in ARPACK (NETLIB/scalapack/readme.arpack [171, 233]).

If we apply the Lanczos algorithm to the shifted and inverted matrix $(A - \sigma I)^{-1}$, then we expect the eigenvalues closest to σ to converge first. There are other methods to "precondition" a matrix A to converge to certain eigenvalues more quickly. For example, Davidson's method [60] is used in quantum chemistry problems, where A is strongly diagonally dominant. It is also possible to combine Davidson's method with Jacobi's method [229].

7.7. Iterative Algorithms for the Nonsymmetric Eigenproblem

When A is nonsymmetric, the Lanczos algorithm described above is no longer applicable. There are two alternatives.

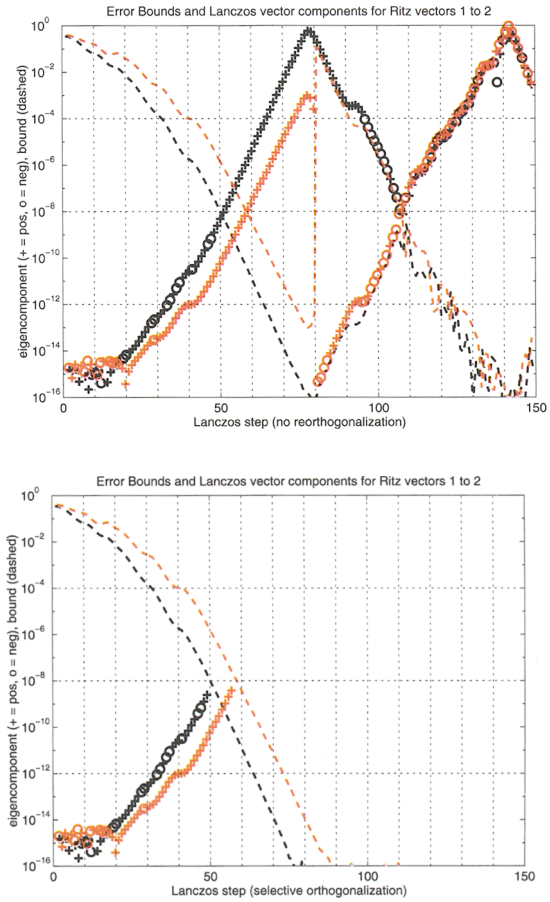

Fig. 7.10. *The Lanczos algorithm with selective orthogonalization applied to A. The top graph shows the first* 149 *steps of the Lanczos algorithm with no reorthogonalization, and the bottom graph shows the Lanczos algorithm with selective orthogonalization. The largest eigenvalue is shown in black, and the second largest eigenvalue is shown in red. The dashed lines are error bounds as before. The lines marked by pluses and o's show* $y_{k,i}^T q_{k+1}$, *the component of Lancos vector* $k+1$ *in the direction of the Ritz vector for the largest Ritz value* ($i = 1$, *in black) or for the second largest Ritz value* ($i = 2$, *in red). Note that selective orthogonalization eliminates these components after the first selective orthogonalizations at steps* 50 ($i = 1$) *and* 58 ($i = 2$).

The first alternative is to use the *Arnoldi algorithm* (Algorithm 6.9). Recall that the Arnoldi algorithm computes an orthogonal basis Q_k of a Krylov subspace $\mathcal{K}_k(A, q_1)$ such that $Q_k^T A Q_k = H_k$ is upper Hessenberg rather than symmetric tridiagonal. The analogue of the Rayleigh–Ritz procedure is again to approximate the eigenvalues of A by the eigenvalues of H_k. Since A is nonsymmetric, its eigenvalues may be complex and/or badly conditioned, so many of the attractive error bounds and monotonic convergence properties enjoyed by the Lanczos algorithm and described in section 7.3 no longer hold. Nonetheless, effective algorithms and implementations exist. Good references include [154, 171, 212, 216, 217, 233] and the book [213]. The latest software is described in [171, 233] and may be found in NETLIB/scalapack/readme.arpack. The Matlab command `eigs` (for "sparse eigenvalues") uses this software.

A second alternative is to use the *nonsymmetric Lanczos algorithm.* This algorithm attempts to reduce A to nonsymmetric tridiagonal form by a nonorthogonal similarity. The hope is that it will be easier to find the eigenvalues of a (sparse!) nonsymmetric tridiagonal matrix than the Hessenberg matrix produced by the Arnoldi algorithm. Unfortunately, the similarity transformations can be quite ill-conditioned, which means that the eigenvalues of the tridiagonal and of the original matrix may greatly differ. In fact, it is not always possible to find an appropriate similarity because of a phenomenon known as "breakdown" [42, 134, 135, 199]. Attempts to repair breakdown by a process called "look-ahead" have been proposed, implemented, and analyzed in [16, 18, 55, 56, 64, 108, 202, 265, 266].

Finally, it is possible to apply subspace iteration (Algorithm 4.3) [19], Davidson's algorithm [216], or the Jacobi–Davidson algorithm [230] to the sparse nonsymmetric eigenproblem.

7.8. References and Other Topics for Chapter 7

In addition to the references in sections 7.6 and 7.7, there are a number of good surveys available on algorithms for sparse eigenvalues problems: see [17, 51, 125, 163, 197, 213, 262]. Parallel implementations are also discussed in [76].

In section 6.2 we discussed the existence of on-line help to choose from among the variety of iterative methods available for solving $Ax = b$. A similar project is underway for eigenproblems and will be incorporated in a future edition of this book.

7.9. Questions for Chapter 7

QUESTION 7.1. *(Easy)* Confirm that running the Arnoldi algorithm (Algorithm 6.9) or the Lanczos algorithm (Algorithm 6.10) on A with starting vector q yields the identical tridiagonal matrices T_k (or Hessenberg matrices H_k) as running on $Q^T A Q$ with starting vector $Q^T q$.

QUESTION 7.2. *(Medium)* Let λ_i be a simple eigenvalue of A. Confirm that if q_1 is orthogonal to the corresponding eigenvector of A, then the eigenvalues of the tridiagonal matrices T_k computed by the Lanczos algorithm in exact arithmetic cannot converge to λ_i in the sense that the largest T_k computed cannot have λ_i as an eigenvalue. Show, by means of a 3-by-3 example, that an eigenvalue of some other T_k can equal λ_i "accidentally."

QUESTION 7.3. *(Medium)* Confirm that no symmetric tridiagonal matrix T_k computed by the Lanczos algorithm can have an exactly multiple eigenvalue. Show that if A has a multiple eigenvalue, then the Lanczos algorithm applied to A must break down before the last step.