MATHEMATICS OF COMPUTATION
VOLUME 42, NUMBER 165
JANUARY 1984, PAGES 115–142

The Lanczos Algorithm With Partial Reorthogonalization

By Horst D. Simon*

Abstract. The Lanczos algorithm is becoming accepted as a powerful tool for finding the eigenvalues and for solving linear systems of equations. Any practical implementation of the algorithm suffers however from roundoff errors, which usually cause the Lanczos vectors to lose their mutual orthogonality. In order to maintain some level of orthogonality, full reorthogonalization (FRO) and selective orthogonalization (SO) have been used in the past as a remedy. Here partial reorthogonalization (PRO) is proposed as a new method for maintaining semiorthogonality among the Lanczos vectors. PRO is based on a simple recurrence, which allows us to monitor the loss of orthogonality among the Lanczos vectors directly *without* computing the inner products. Based on the information from the recurrence, reorthogonalizations occur only when necessary. Thus substantial savings are made as compared to FRO. In some numerical examples we apply the Lanczos algorithm with PRO to the solution of large symmetric systems of linear equations and show that it is a robust and efficient algorithm for maintaining semiorthogonality among the Lanczos vectors. The results obtained compare favorably with the conjugate gradient method.

1. Introduction. In recent years there has been considerable interest in the Lanczos algorithm and its applications [1]–[4], [8]–[20], stimulated by the unusual behavior of the algorithm in finite precision arithmetic and by its great potential for sparse matrix problems. After Paige [10], [11] gave a thorough analysis of the algorithm, research among numerical analysts was stimulated along two directions. In one direction of research the simple ("Paige-style") Lanczos algorithm was considered without further modifications [1], [2], [14], [17]. A tridiagonal matrix is obtained which may be up to six times larger than the original matrix [17], yet contains approximations to all of the original matrix's eigenvalues. As of today there is no proof that all the eigenvalues will be found by this procedure.

On the other hand because of its so-called "instability" it was traditionally recommended to use the Lanczos algorithm only with full reorthogonalization [3], [22]. This was considered too expensive for large matrices. A second line of research attempts to cut down the number of orthogonalizations, yet obtain results from the practical Lanczos algorithm that are close to the ideal, roundoff-free algorithm (e.g. no appearance of duplicate copies of eigenvalues and termination after at most n steps). In order to achieve this goal Parlett and Scott [18] introduced selective orthogonalization (SO), which utilizes Paige's [10] theoretical explanation of the behavior of the algorithm, and performs reorthogonalizations only when necessary. Recently Grcar [4] presented a forward error analysis of the Lanczos algorithm and

Received September 8, 1982.
1980 *Mathematics Subject Classification*. Primary 65F10.
*The author gratefully acknowledges support from ONR Grant N00014-69-0200-1017.

©1984 American Mathematical Society
0025-5718/84 $1.00 + $.25 per page

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

in the light of his results proposed periodic reorthogonalization. Here we will follow this second line of research and propose a new orthogonalization method called partial reorthogonalization (PRO).

In order to present this new method we first introduce the Lanczos algorithm in Section 2, and then in Section 3 we discuss its behavior in the presence of roundoff. The loss of orthogonality among the Lanczos vectors is governed by a simple recurrence. This recurrence is the basis for PRO, which will be introduced in Section 4. Sections 5 to 8 deal with the computational details of PRO. In Section 9 we briefly compare PRO to the other reorthogonalization methods mentioned above. In Section 10 we present some numerical results.

In this paper we will follow the Householder convention and denote column vectors by small Roman letters, matrices by capital Roman letters, and scalars by small Greek letters. Symmetric matrices are indicated by symmetric letters (A, T), and $\| \; \|$ denotes the Euclidean norm for vectors, or the associated matrix norm. The conjugate transpose of v is denoted by v^*.

2. The Lanczos Algorithm in Exact Arithmetic. The simple Lanczos algorithm for a symmetric $n \times n$ matrix A computes a sequence of Lanczos vectors q_j and scalars α_j, β_j as follows:

```
1: choose a starting vector $r_1$, $r_1 \neq 0$, set $q_0 \equiv 0$, $\beta_1 \equiv \|r_1\|$,
2: for $j = 1,2,\ldots$ do
       $q_j = r_j/\beta_j$
       $u_j = Aq_j - \beta_j q_{j-1}$
       $\alpha_j = u_j^* q_j$
       $r_{j+1} = u_j - \alpha_j q_j$
       $\beta_{j+1} = \|r_{j+1}\|$.
```

One pass through step 2 is a Lanczos step. One Lanczos step is commonly derived from

$$\beta_{j+1} q_{j+1} = Aq_j - \alpha_j q_j - \beta_j q_{j-1}. \tag{2.1}$$

These equations can be condensed in matrix form as

$$AQ_j - Q_j T_j = \beta_{j+1} q_{j+1} e_j^*, \tag{2.2}$$

where $Q_j = (q_1, \ldots, q_j)$, $e_j^* = (0, 0, \ldots, 1)$ and

$$T_j \equiv \begin{bmatrix} \alpha_1 & \beta_2 & 0 & & \\ & & & \cdot & \cdot \\ \beta_2 & \alpha & \beta_3 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \beta_{j-1} & \alpha_{j-1} & \beta_j \\ \cdot & \cdot & 0 & \beta_j & \alpha_j \end{bmatrix}.$$

The Lanczos vectors q_j are orthonormal, i.e.

$$Q_j^* Q_j = I_j, \tag{2.3}$$

where I_j is the $j \times j$ identity matrix.

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

Paige [11] has shown that the above implementation is the best among several other possible ones.

The algorithm terminates if $\beta_{j+1} = 0$, and this will happen for some $j \leqslant n$ in exact arithmetic. The eigenvalues of the tridiagonal matrix T_j are called the Ritz values. If s_i, $i = 1,\ldots,j$, are the eigenvectors of T_j, the vectors $y_i = Q_j s_i$ are called the Ritz vectors. Ritz values and vectors are the Rayleigh-Ritz approximations to the eigenvalues and vectors of A from span(Q_j), the subspace spanned by the vectors $q_1,\ldots, q_j$. More details on the Lanczos algorithm for computing eigenvalues can be found in [15].

The algorithm can also be used for solving linear systems of equations $Ax = b$. Then b is chosen as starting vector, and at the jth step an approximate solution is given by $x_j \equiv Q_j T_j^{-1}\beta_1 e_1$. This is explained in detail in [16] and [20].

3. The Loss of Orthogonality. If the Lanczos algorithm is carried out in finite precision arithmetic, it behaves quite differently. The inevitable rounding errors in the computation affect the algorithm in a special way. Equation (2.1) becomes now

$$\beta_{j+1}q_{j+1} = Aq_j - \alpha_j q_j - \beta_j q_{j-1} - f_j, \tag{3.1}$$

where the n-vector f_j accounts for the rounding errors at the jth step, and α_j, β_j, and q_j denote from now on the corresponding computed quantities. Usually $\|f_j\|$ is small, so (2.1) still holds in an approximate sense. In contrast relation (2.3) fails completely after a while, i.e. the computed Lanczos vectors are no longer orthogonal, not even up to roundoff. This infamous loss of orthogonality is illustrated in Figure 3.1, where we have plotted $\log_{10}|q_j^* q_k/\varepsilon|$ rounded to the next integer for a sample run. Here ε denotes the roundoff unit. The integers in Figure 3.1 indicate by what power of 10 the inner products $q_j^* q_k$ have risen over the roundoff level.

Figure 3.1 shows that the orthogonality relation (2.3) starts failing already at an early stage during the algorithm. However a careful look at the numbers in Figure 3.1 also reveals that the growth of $|q_j^* q_k|$ is by no means irregular or jumpy, but appears to follow a certain rule. Indeed we have the following

THEOREM 1. *Let $\omega_{ik} = q_i^* q_k$. Then the ω_{ik} satisfy the following recurrence*:

$$\begin{aligned} \omega_{kk} &= 1 \qquad \textit{for } k = 1,\ldots,j, \\ \omega_{kk-1} &= q_k^* q_{k-1} \qquad \textit{for } k = 2,\ldots,j, \end{aligned} \tag{3.2}$$

$$\beta_{j+1}\omega_{j+1k} = \beta_{k+1}\omega_{jk+1} + (\alpha_k - \alpha_j)\omega_{jk} + \beta_k\omega_{jk-1} - \beta_j\omega_{j-1k} + q_j^* f_k - q_k^* f_j,$$

for $1 \leqslant k < j$, *and* $\omega_{jk+1} = \omega_{k+1j}$. *Here* $\omega_{k0} \equiv 0$.

Proof. Write (3.1) for j and for k:

$$\beta_{j+1}q_{j+1} = Aq_j - \alpha_j q_j - \beta_j q_{j-1} - f_j, \tag{3.3}$$

$$\beta_{k+1}q_{k+1} = Aq_k - \alpha_k q_k - \beta_k q_{k-1} - f_k. \tag{3.4}$$

Form $q_k^*(3.3) - q_j^*(3.4)$ and simplify to obtain the result. □

Theorem 1 was already known by Paige [10] and by Takahasi and Natori [21]. But here it is used for the first time as a computational tool. It is also of central importance for the understanding of the loss of orthogonality. Its statement can be visualized by considering Figure 3.2.

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

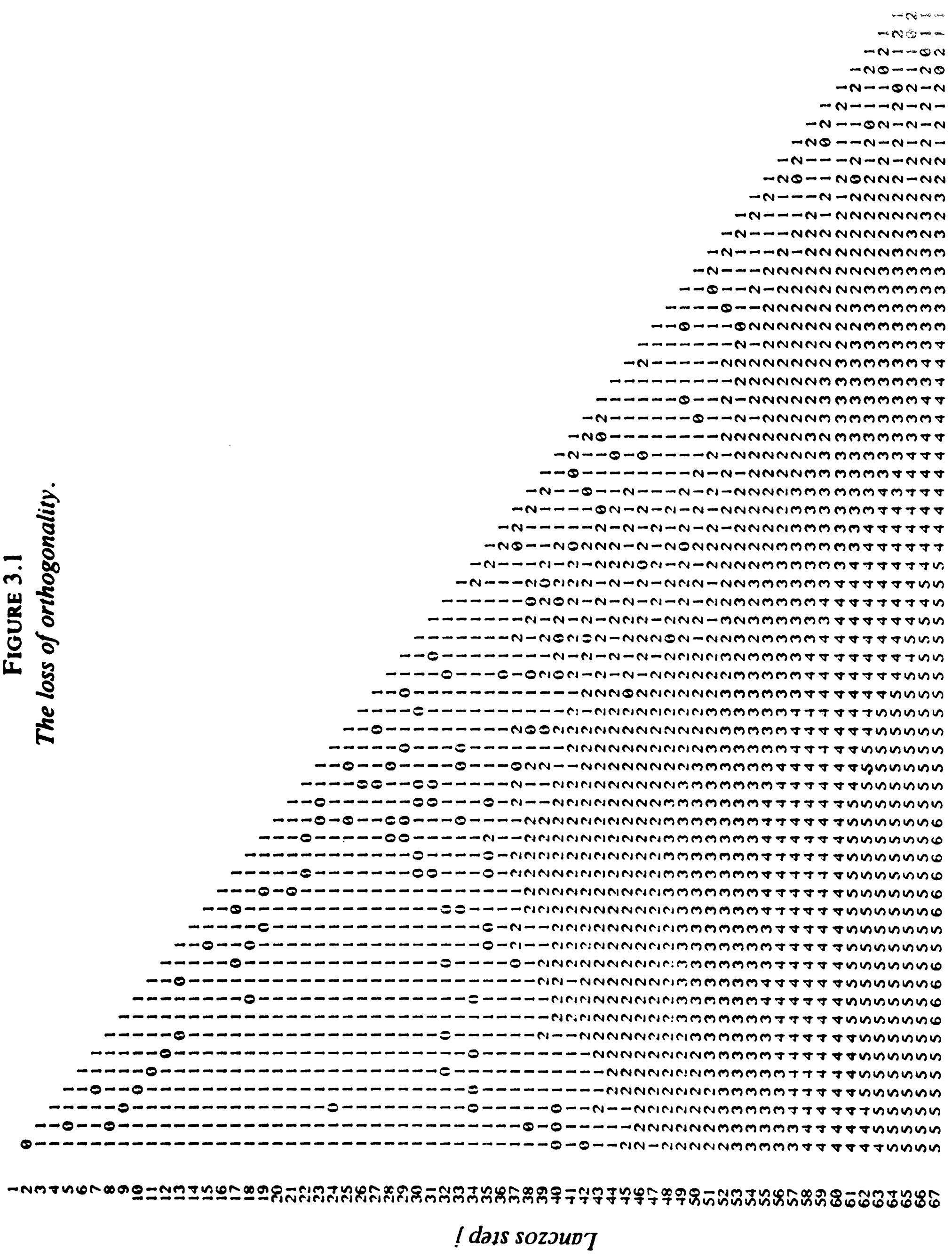

FIGURE 3.1
The loss of orthogonality.

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

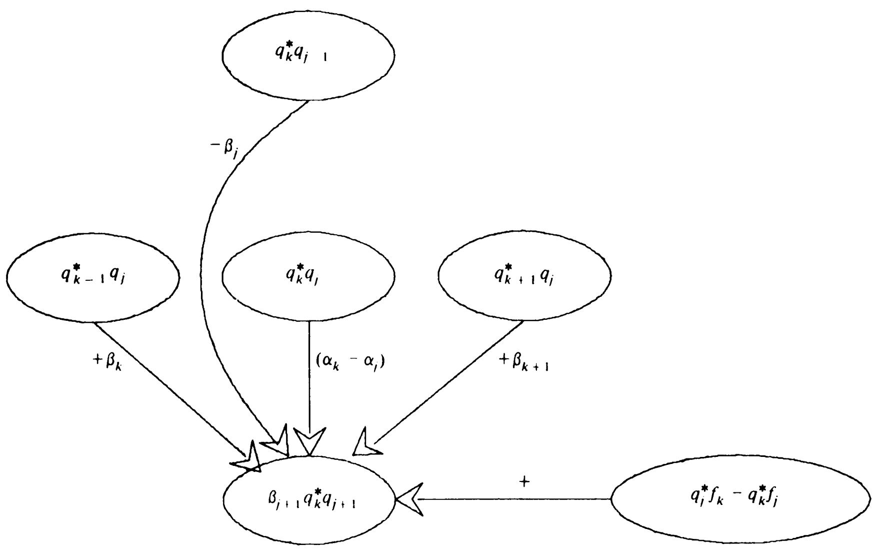

FIGURE 3.2
The statement of Theorem 1.

Theorem 1 says that the inner product $q_{j+1}^* q_k$ is a weighted combination of inner products from the previous two Lanczos steps, where the weights are the coefficients from the Lanczos recurrence. In addition the roundoff term $q_j^* f_k - q_k^* f_j$ enters the picture. The loss of orthogonality follows a second order inhomogeneous difference equation (3.2) with variable coefficients. In the ideal algorithm the roundoff terms f_j and the $q_k^* q_{k-1}$ are all zero. We then obtain a homogeneous difference equation with zero initial conditions. Hence all the ω_{jk} will be zero, i.e. the Lanczos vectors are orthogonal. So the loss of orthogonality can also be explained by the instability of the difference equation. An attempt to analyze (3.2) further [20] yields Paige's well-known theorem [10], [15, p. 264].

4. Partial Reorthogonalization. The goal of all reorthogonalization methods mentioned in Section 1 is to prevent the loss of orthogonality, i.e. to maintain a certain level of orthogonality among the Lanczos vectors. We define the level of orthogonality κ_j among the Lanczos vectors at the jth step as:

$$\kappa_j \equiv \max_{1 \leqslant k \leqslant j-1} |q_j^* q_k|. \tag{4.1}$$

Clearly full reorthogonalization, i.e. the explicit reorthogonalization of q_{j+1} against all previous Lanczos vectors, aims at keeping the level of orthogonality at roundoff level. However all that effort is not necessary. Numerical results (Scott [19]) and theoretical considerations in connection with the various reorthogonalization methods (Parlett and Scott [18], Parlett [15], Grcar [4]) have shown that semiorthogonality, i.e. $\kappa_j \equiv \sqrt{\varepsilon}$, among the Lanczos vectors is sufficient to permit the computation of eigenvalues without the appearance of spurious duplicate copies. The analysis of all these methods can be unified and we have the following theorem [20]:

THEOREM 2. *Let T_j be the tridiagonal matrix computed by the Lanczos algorithm, where by some means the Lanczos vectors are kept semiorthogonal. Then T_j is, up to roundoff, the orthogonal projection of A onto* span(Q_j).

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

Proof. See [20, Theorem 2.5].

Theorem 2 implies that the eigenvalues of T_j are (up to roundoff) Rayleigh-Ritz approximations to the eigenvalues of A, although from a slightly different subspace than the ideal one. Similarly it can be shown that if an approximate solution x_j to $Ax = b$ is computed by $x_j = Q_jT_j^{-1}Q_j^*b$, then $\|x_j - \bar{x}_j\| \leqslant \sqrt{2j\varepsilon}$, where $\bar{x}_j$ is the best approximate solution from span(Q_j) [20].

Semiorthogonality therefore appears to be all that is needed for the finite precision Lanczos algorithm in order to preserve most of the properties of the ideal algorithm. The method of partial reorthogonalization can now be described at an abstract level as follows: Using the recurrence (3.2), we compute estimates ω_{j+1k} for the inner products of the Lanczos vectors, and then judiciously perform reorthogonalizations based on the information from the recurrence in order to maintain semiorthogonality. In a more formal way the algorithm for PRO can be written as follows:

(1) Perform a regular Lanczos step:

$$r'_{j+1} \equiv Aq_j - \alpha_j q_j - \beta_j q_{j-1} - f'_j. \tag{4.2a}$$

(2) Update the estimates ω_{j+1k} for $q'^*_{j+1}q_k$, for $k = 1,\dots,j$ using the recurrence (3.2).

(3) Based on the information from the ω_{j+1k}, determine a set of indices $L(j) = \{k|1 \leqslant k \leqslant j\}$ and compute

$$r_{j+1} = r'_{j+1} - \sum_{k \in L(j)} q_k(r'^*_{j+1}q_k) - f_j. \tag{4.2b}$$

PRO has some obvious advantages. The computation of the estimates ω_{jk} involves only a simple updating procedure for two vectors of length j. No inner products of Lanczos vectors have to be formed, and yet the loss of orthogonality can be monitored except for the roundoff term. For many Lanczos steps no orthogonalization at all may be necessary, and this information can be gained quite cheaply.

But even when some $|\omega_{jk}| > \sqrt{\varepsilon}$ indicate that semiorthogonality has been lost, then orthogonalizations against some, but not all previous Lanczos vectors are necessary. Against which Lanczos vectors one should orthogonalize, and how the recurrence (3.2) is evaluated computationally, is discussed in the following sections.

5. Computing the Level of Orthogonality. An accurate evaluation of (3.2) would be advantageous in two respects: the loss of orthogonality (given by $\omega_{jk} = q_j^*q_k$) could be monitored directly and these inner products would be on hand in the event of a reorthogonalization. However (3.2) involves the local error vectors f_j, which are unknown unless one wants to compute them using double precision. But even this may be impossible if the matrix vector product is inaccessible and truly in black box form. The lack of f_j appears to make (3.2) useless, but there is a way to utilize (3.2) without computing f_j.

Once the ω_{jk}'s have risen to a level close to $\sqrt{\varepsilon}$ the $q_j^*f_k - q_k^*f_j$-terms, which are at roundoff level, do not contribute significantly to the value of ω_{j+1k}. These terms are only important as long as the ω_{jk} are small like $\varepsilon\|A\|$. We propose that the computation of the inner products $q_{j+1}^*q_k$ can be simulated by replacing the

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

unknown quantities by random values from appropriate ranges as follows:

$$(5.1)\qquad \begin{aligned} \omega_{kk} &= 1 \quad \text{for } k = 1,\dots,j,\\ \omega_{k\,k-1} &= \psi_k \quad \text{for } k = 2,\dots,j,\end{aligned}$$

$$\omega_{j+1\,k} = \frac{1}{\beta_{j+1}}\left[\beta_{k+1}\omega_{j\,k+1} + (\alpha_k - \alpha_j)\omega_{jk} + \beta_k\omega_{j\,k-1} - \beta_j\omega_{j-1\,k}\right] + \vartheta_{jk},$$

for $1 \leqslant k < j$, and $\omega_{j\,k+1} = \omega_{k+1\,j}$. Here $\omega_{k0} \equiv 0$, and ϑ_{jk} and ψ_k are certain random numbers, which have to be chosen appropriately. From now on we will refer to the ω_{jk}'s computed with (5.1) as the *computed* or *estimated orthogonality components*, in contrast to the true components which are given by the inner products $q_j^* q_k$.

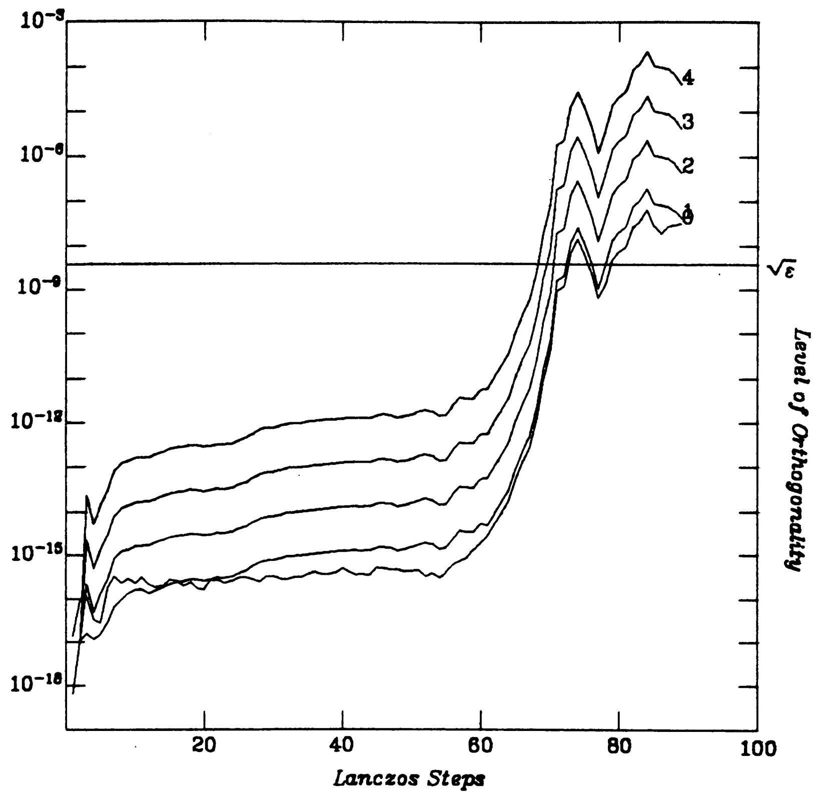

FIGURE 5.1
True and estimated level of orthogonality for various choices of ϑ_{jk}.

0—True level of orthogonality
1—Estimated level of orthogonality, $\kappa = 1.0$
2—Estimated level of orthogonality, $\kappa = 10.0$
3—Extimated level of orthogonality, $\kappa = 100.0$
4—Estimated level of orthogonality, $\kappa = 1000.0$

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

Formula (5.1) can be regarded as a simulation of how the loss of orthogonality would occur on a different machine which generated numbers ϑ_{jk} and ψ_k as actual roundoff errors. From Theorem 1 we can conclude that the loss of orthogonality mainly depends on the α_j and β_j, and from Theorem 2 we know that the computed α_j and β_j are exact up to roundoff. Therefore the computed loss of orthogonality from formula (5.1) will behave like the true loss of orthogonality as soon as the ω_{jk}'s exceed $n\varepsilon$.

This is illustrated by the following example, where we examined the dependence of formula (5.1) on the choice of ϑ_{jk} and ψ_k. For a matrix of order $n = 128$, which is part of the matrix in Example 1.2, Section 10, and with starting vector $q_1^* = (1, \ldots, 1)/\sqrt{128}$, we determined first the true loss of orthogonality. It turns out that for this matrix the Lanczos vectors remain semiorthogonal for 71 steps. Then we computed in two series of experiments the values for ω_{jk} with (5.1). First we chose $\psi_k \in N(0, \varepsilon)$ (i.e., we chose for the ψ_k's a sequence of normally distributed random numbers with mean 0 and standard deviation ε), and $\vartheta_{jk} \in N(0, \kappa\varepsilon)$, with $\kappa = 1.0$, 10.0, 100.0, 1000.0. Then we kept ϑ_{jk} fixed and varied ψ_k. The true and the estimated loss of orthogonality are plotted in Figures 5.1 and 5.2.

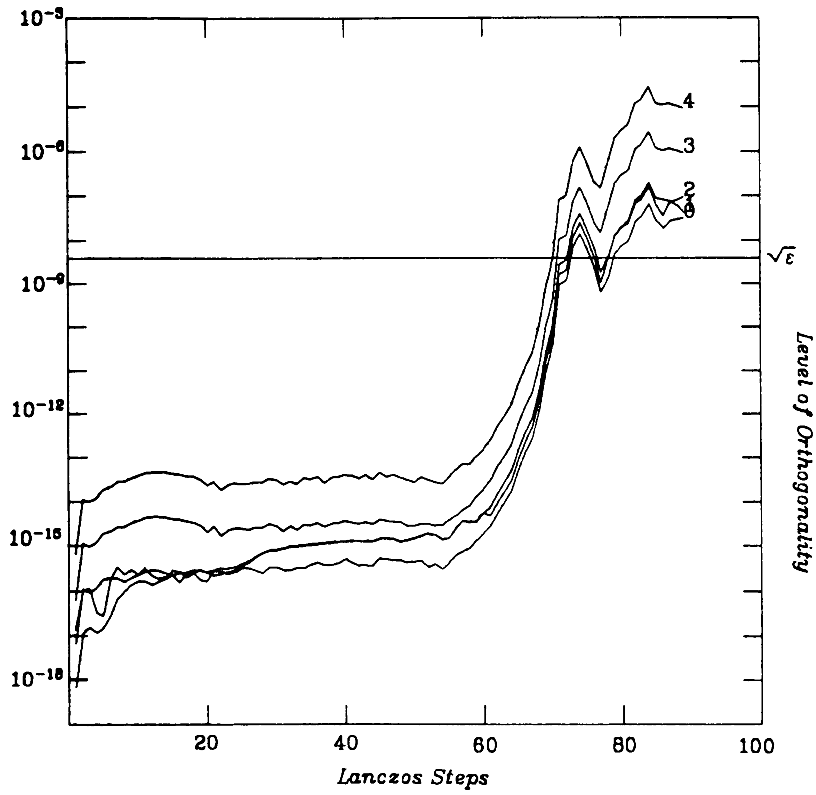

FIGURE 5.2
True and estimated level of orthogonality for various choices of ψ_k.
(Graphs labeled as in Figure 5.1)

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

Figures 5.1 and 5.2 show that the estimated level of orthogonality with formula (5.1) reflects quite well the qualitative behavior of the true level of orthogonality. It is important to see that although, due to an overestimate of the error terms the computed level of orthogonality lies initially above the true level of orthogonality, the curves move very close together when they reach the critical $\sqrt{\varepsilon}$ region. Even the curve with the largest overestimate reaches the $\sqrt{\varepsilon}$ threshold only *three* steps too early, at step 68. In spite of the dependence on the random terms, (5.1) appears to produce an accurate estimate for the step at which orthogonality is lost.

These tests were repeated with different examples in [20] and similar results were obtained. In all the cases (5.1) signals at about the right Lanczos step that the $\sqrt{\varepsilon}$-level has been reached. These tests also show that the recurrence is relatively insensitive to moderate overestimates in the error terms. For example, as Figure 5.1 shows, an increase in the estimate for the $q_j^*f_k - q_k^*f_j$-term by a factor 1000, resulted in ω_{jk}'s which reached the threshold only 3 steps too early. For a practical computation of the level of orthogonality with (5.1) in connection with PRO it is therefore advisable to overestimate these terms somewhat.

At this point we could content ourselves with the analysis of these error terms, since their direct influence on the loss of orthogonality is not too strong. However, there is one incentive, which may make a further study of these terms rewarding. It may be possible to compute (5.1) so accurately that the direct computation of $q_{j+1}^*q_k$ can be saved and the values $\omega_{j+1\,k}$ can be used instead in the usual reorthogonalization process.

In order to obtain more information about the behavior of the $q_j^*f_k - q_k^*f_j$-terms and $q_{j+1}^*q_k$, a detailed statistical study of the roundoff quantities has been performed**. The results of this study are reported in [20]. Based on this study we decided to choose

$$\vartheta_{jk} = \varepsilon(\beta_{k+1} + \beta_{j+1})\Theta, \tag{5.2}$$

where $\Theta \in N(0, 0.3)$, and

$$\psi_k = \varepsilon n \frac{\beta_2}{\beta_{j+1}} \Psi, \tag{5.3}$$

where $\Psi \in N(0, 0.6)$.

There is one more error term to be considered. After a reorthogonalization has been performed, the terms $q_{j+1}^*q_k$ have to be reset. Ideally, of course, these inner products should be zero, but here we expect them to be at roundoff level. Again we performed a statistical study and decided to choose $\omega_{j+1\,k} \in N(0, 1.5)\varepsilon$ after a reorthogonalization has been performed.

6. The Behavior of the Computed Level of Orthogonality.

After the roundoff quantities were chosen as described in the previous section, we tested (5.1) with several examples where a full reorthogonalization was performed whenever one ω_{jk} became larger than the threshold of $\sqrt{\varepsilon}$. In Figures 6.1 and 6.2 the true level of orthogonality and the computed estimate are plotted for two of the sample runs.

**All computations were carried out on the VAX 11/780 of the EECS Department, Computer Science Division at the University of California, Berkeley. For single-precision computations the roundoff unit $\varepsilon = 2^{-24}$, for double precision $\varepsilon = 2^{-56}$.

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

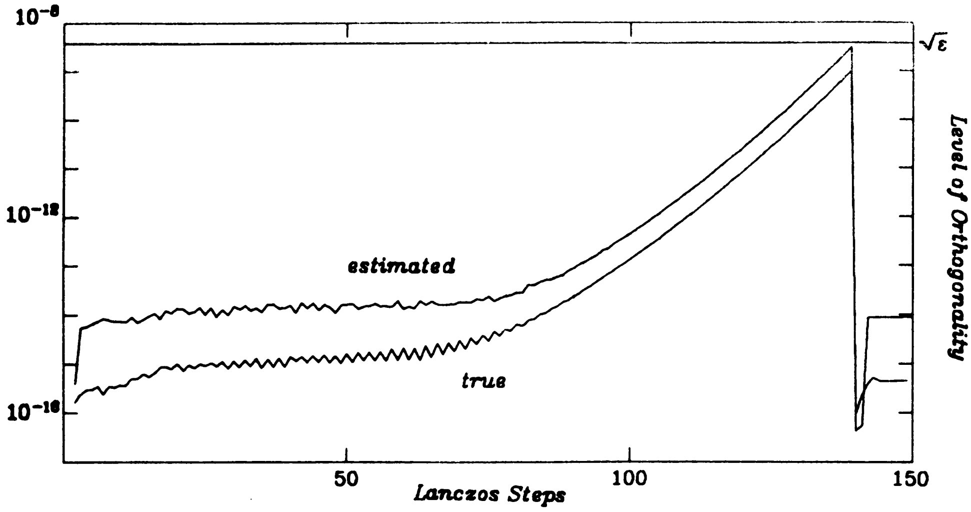

FIGURE 6.1
True and computed level of orthogonality for A = diag$(1^2, 2^2, \dots, 1000^2)$
and q_1^* = $(1, 1, \dots, 1)/\sqrt{1000}$.
(For the effect of having A diagonal see next section)

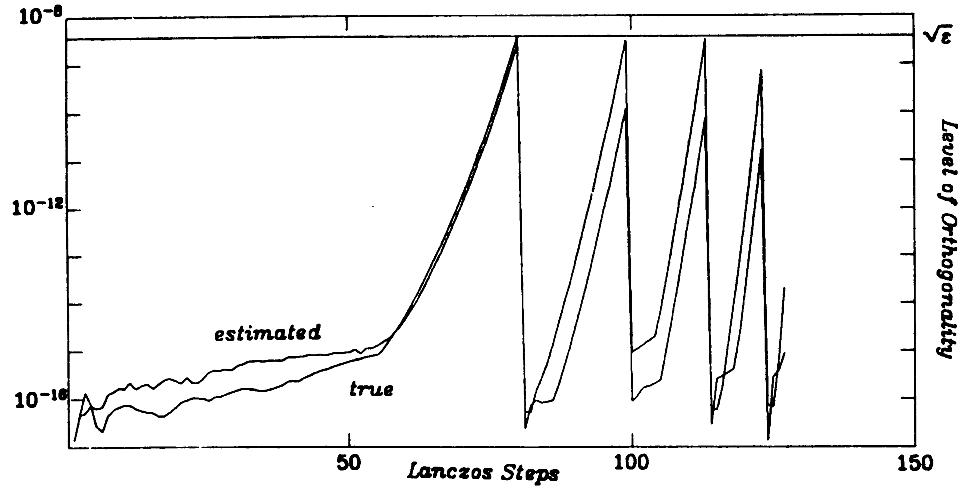

FIGURE 6.2
True and computed level of orthogonality for the matrix
used in Figures 5.1 *and* 5.2.

These figures show that the computed level of orthogonality behaves as expected. The overestimates for the error terms cause an overestimate for the computed level of orthogonality as long as it is about $\varepsilon^{3/4}$. If the level of orthogonality increases further the error terms are relatively unimportant and the computed level of orthogonality approximates the true level of orthogonality quite closely.

In Figure 6.1 we used a diagonal matrix for test purposes. This seems to be artificial and a trivial example. The Lanczos algorithm is however invariant (in exact arithmetic) under similarity transformations and a diagonal matrix as good as any other for a *theoretical* study of the Lanczos algorithm. Since it is not obvious that

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

this is also true in a finite precision environment, we repeated the sample run from Figure 6.1 with a similarity transformation of the diagonal matrix A. The starting vector was changed accordingly. We obtained:

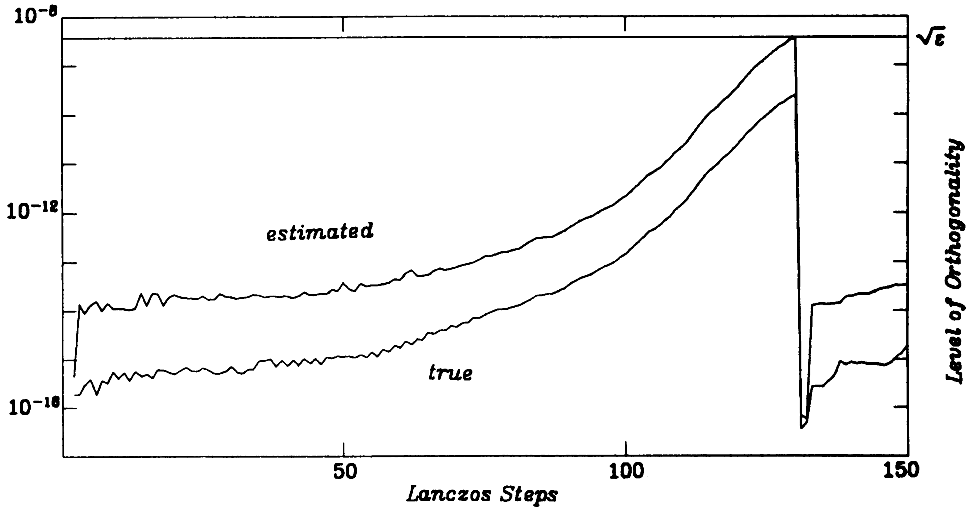

FIGURE 6.3

True and computed level of orthogonality for a matrix similar to A from Figure 6.1.

The level of orthogonality is different from Figure 6.1. Here the threshold is reached about 10 steps earlier. This different behavior is due to the fact that the tridiagonal matrix is changed slightly, and the change in α_j and β_j in turn produces different orthogonality components. This is not surprising, and consistent with the results from Section 3. However what is more important for our analysis here is the fact that in both cases computed and true level of orthogonality agree well with each other in the sense that the reaching of the threshold is signalized at about the right time. Their mutual relation is not affected by whether a diagonal matrix is used or not. So although diagonal matrices are of course trivial examples for solving linear systems, it is quite legitimate (and cheaper) to use them for the purpose of studying the loss of orthogonality and related questions. In the following sections we will therefore repeatedly use diagonal matrices as test examples.

The properties of (5.1) discussed above turn the formula into a useful tool for predicting the level of orthogonality. It would be even more convenient if the ω_{jk} would be so accurate that the inner products $q_j^* q_k$ would not have to be recomputed. Let us recall that by Paige's Theorem ([10], cf. [15, p. 264]) the vector $u_j \equiv Q_j^* q_{j+1} = (q_1^* q_{j+1}, q_2^* q_{j+1}, \ldots, q_j^* q_{j+1})^*$ tilts towards an eigenvector of T_j, when the corresponding Ritz value is about to converge to an eigenvalue of A. Let us consider now the vector $w_j \equiv (\omega_{j+1\,1}, \omega_{j+1\,2}, \ldots, \omega_{j+1\,j})^*$ computed by (5.1). Earlier we expressed the view that the computation of (5.1) can be considered as a simulation of the level of orthogonality that would occur on a different machine, where the random numbers chosen for ϑ_{jk} and ψ_k would be equal to the corresponding actual roundoff error terms. Therefore Paige's Theorem will also hold for w_j, i.e., w_j will have large components in direction of those eigenvectors of T_j for which the corresponding Ritz values are about to converge.

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

In numerical tests [20] we observed a behavior of u_j and w_j consistent with Paige's Theorem. However we only know *that* u_j and w_j will form a small angle with the subspace spanned by the eigenvectors corresponding to converging Ritz values, but we do not know *how* u_j and w_j will behave in relation to the individual eigenvectors of T_j. Since in general at a given Lanczos step we do not even know how many Ritz values are about to converge (unless we want to do a spectral analysis of T_j comparable to selective orthogonalization), there seems to be no easy way to relate u_j and w_j either in terms of eigenvectors of T_j or directly.

Nevertheless we tried to use the computed ω_{jk} instead of the exact inner products $q_j^*q_k$, when performing a reorthogonalization. It turned out that the level of orthogonality is indeed reduced to a value below the threshold level, however not to roundoff level. This had the negative effect that the next reorthogonalization occurred much earlier. So although we saved the computation of the inner products, reorthogonalizations were needed more frequently, and no overall savings in computations were made. Therefore the $\omega_{j+1\,k}$ are used in PRO only for estimating the level of orthogonality, but not for the computation of the $q_{j+1}^*q_k$.

7. Choosing Reorthogonalizations. In Sections 5 and 6 we saw how to compute the level of orthogonality from (5.1), and what information from the computed level of orthogonality can be inferred. In this section we will discuss how this information is used in order to decide when and against which past Lanczos vectors the current Lanczos vector has to be orthogonalized.

From the remarks in Section 4 it follows that it is always necessary to orthogonalize q_{j+1} against some previous Lanczos vectors, if $|q_{j+1}^*q_k| > \sqrt{\varepsilon}$ for some k. $\sqrt{\varepsilon}$ is the optimal threshold here, since it is the largest loss of orthogonality among the Lanczos vectors which we can tolerate and still obtain accurate α_j's and β_j's. A smaller threshold would result only in more orthogonalizations without any gain in accuracy. This is confirmed by numerical tests (Scott [19, p. 82]) in relation with the analysis of selective orthogonalization.

There is another important idea concerning reorthogonalization, which we can borrow from the method of selective orthogonalization [18]. Suppose at step j we decided to reorthogonalize q_{j+1} against all previous q_k, then we will also reorthogonalize at step $j+1$ the new Lanczos vector q_{j+2} against all previous q_k, no matter what the $q_{j+1}^*q_k$ are. There is a direct justification of this additional reorthogonalization through formula (3.2). By reorthogonalizing at step j we make $q_{j+1}^*q_k = O(\varepsilon)$ for all $k \leqslant j$. Then

$$\beta_{j+2}q_{j+2}^*q_k = -\beta_{j+1}q_j^*q_k + O(\varepsilon). \tag{7.1}$$

But if for some k, $|q_{j+1}^*q_k| \geqslant \sqrt{\varepsilon}$ before the reorthogonalization, then also $q_j^*q_k$ must have been comparatively large, i.e. almost as big as $\sqrt{\varepsilon}$. One reorthogonalization by itself therefore does not help very much to reduce the size of $q_{j+2}^*q_k$. If, however, two reorthogonalizations are performed in a row, then formula (3.2) yields

$$\beta_{j+2}q_{j+2}^*q_k = O(\varepsilon), \tag{7.2}$$

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

and we can be sure that at least for the next couple of steps the level of orthogonality will remain small.

So far we have assumed that during one reorthogonalization the current Lanczos vector was orthogonalized against *all* previous Lanczos vectors. But this is not necessary if our aim is to maintain only semiorthogonality. An important observation concerning the loss of orthogonality can be drawn from Figure 7.1. Here we plotted on a logarithmic scale $q_{43}^* q_k$, $k = 1, \ldots, 42$ for a run of the Lanczos algorithm with $A = \text{diag}(1, 4, 9, \ldots, 100^2)$ and $q_1^* = (1, 1, \ldots, 1)/10$.

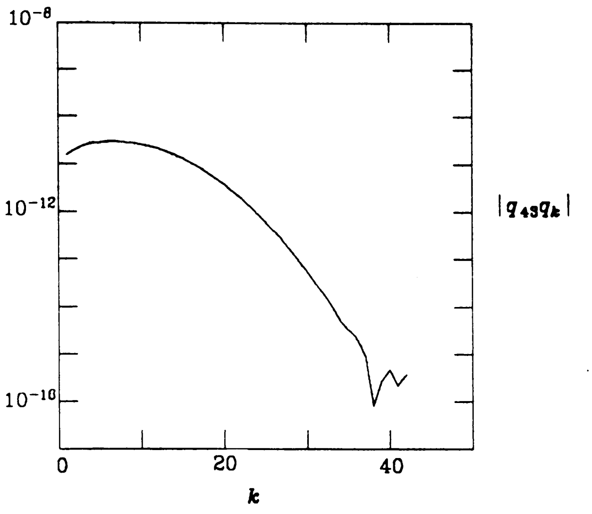

FIGURE 7.1

$|q_{j+1}^* q_k|$ *for fixed* j *and* $k \leq j$.

Figure 7.1 shows the typical pattern in the loss of orthogonality. Usually only some neighboring $q_{j+1}^* q_k$ have grown to about the $\sqrt{\varepsilon}$ level, whereas most other inner products remain quite small. In order to maintain semiorthogonality it is therefore only necessary to orthogonalize against selected Lanczos vectors. In the example given in the table it could be the first ten. Since (5.1) gives a reliable prediction of the loss of orthogonality it can indicate the old Lanczos vectors against which q_{j+1} has to be orthogonalized. It is clear that an orthogonalization against only those q_k with $|q_{j+1}^* q_k| > \sqrt{\varepsilon}$ alone is not sensible. The same argument which was used to introduce two successive orthogonalizations at consecutive Lanczos steps can be applied again. Formula (7.1) says that $q_{j+1}^* q_k$ depends on $q_j^* q_{k+1}, q_j^* q_k, q_j^* q_{k-1}$, and $q_{j-1}^* q_k$. Therefore it does not help to make only $q_j^* q_k$ and $q_{j+1}^* q_k$ small. The neighboring $q_j^* q_{k+1}$ and $q_j^* q_{k-1}$ have to be reduced in order to make the orthogonalization useful, i.e., not to allow $q_{j+2}^* q_k$ to become large again. However, in order to keep these small for some more Lanczos steps their neighbors in turn have to be small.

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

This situation can be expressed best in the following figure (similar to the domain of dependence/domain of influence argument in numerical PDE):

$k-3$	□	□	□	□	□
$k-2$	■	□	□	□	□
$k-1$	■	■	□	□	□
k	■	■	■	□	□
$k+1$	■	■	□	□	□
$k+2$	■	□	□	□	□
$k+3$	□	□	□	□	□
	j	$j+1$	$j+2$	$j+3$	$j+4$

■–*small* ω_{kj}

□–*large* ω_{kj}

FIGURE 7.2
Propagation of the loss of orthogonality.

Figure 7.2 shows that reorthogonalizations against single Lanczos vectors are useless, since their effect is immediately wiped out by the neighboring large terms. The best strategy for choosing Lanczos vectors to reorthogonalize against therefore seems to be to group them into "batches". One batch contains all the offending Lanczos vectors, i.e., all the q_k with $|q_{j+1}^* q_k| > \sqrt{\varepsilon}$, and in addition to that a certain number of neighboring vectors. The next obvious question is then: how many neighboring vectors should be included in those batches?

A first approach to this problem could be as follows: Suppose $q_j^* q_k$ has grown in l steps from the roundoff level to $\sqrt{\varepsilon}$. Then we should not only orthogonalize against q_k, but also against all vectors from q_{k-l} to q_{k+l}, in order to assure that $q_{j+m}^* q_k$ stays small for another l steps ($m \leqslant l$). But some thought indicates that this is too much work. The terms $q_j^* q_{k+p}$ for $p \leqslant l$, p "far" away from k, may be already quite small (cf. Figure 7.1) by themselves and their influence is only felt in $q_{j+p}^* q_k$, i.e., after p more steps. But then $q_{j+p}^* q_k$ may have grown already by the dynamics of formula (3.2) to a magnitude where the $q_j^* q_{k+p}$ because of its small size plays no role any more.

The question of how many neighboring Lanczos vectors should be used for orthogonalizing apparently cannot be answered a priori. Therefore the following numerical experiment was carried out. At each Lanczos step the recurrence (5.1) was updated. If any of the $|\omega_{j+1\,k}|$ was larger than $\sqrt{\varepsilon}$, then the neighboring $\omega_{j+1\,k-1}$, $\omega_{j+1\,k-2}, \ldots$ and $\omega_{j+1\,k+1}$, $\omega_{j+1\,k+2}, \ldots$ were checked until $\omega_{j+1\,k-s}$ and $\omega_{j+1\,k+r}$ were found with $|\omega_{j+1\,k-s}| \leqslant \eta$ and $|\omega_{j+1\,k+r}| \leqslant \eta$. Then q_{j+1} was orthogonalized against all q's from q_{k-s} to q_{k+r} inclusive. At the following Lanczos step q_{j+2} was orthogonalized against $q_{k-s+1}, \ldots, q_{k+r-1}$. Orthogonalizations against q_{k-s} and q_{k+r} are not necessary any more at the $j+1$st step (except for $k-s=1$), because the inner products $q_{j+2}^* q_{k-s}$ and $q_{j+2}^* q_{k+r}$ will deteriorate anyway due to the influence of unorthogonalized neighbors (cf. Figure 7.2). These runs were repeated for different values of η. Table 7.1 summarizes the results for two examples. For each example we list in the first column the number of orthogonalizations (one orthogonalization = two inner products) and in the second column the number of recalls, i.e., the number of steps at which reorthogonalizations occurred.

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

TABLE 7.1
Influence of the lower bound η on the reorthogonalizations.

	Example I		Example II	
η	Orthogonalizations	Recalls	Orthogonalizations	Recalls
$\sqrt{\varepsilon}$	624	26	1518	40
$10^{-1}\sqrt{\varepsilon}$	520	21	1178	29
$10^{-2}\sqrt{\varepsilon}$	526	17	781	22
$10^{-3}\sqrt{\varepsilon}$	507	15	675	15
$10^{-4}\sqrt{\varepsilon}$	478	12	617	11
$10^{-5}\sqrt{\varepsilon}$	504	10	672	9
$10^{-6}\sqrt{\varepsilon}$	576	10	705	8
$10^{-7}\sqrt{\varepsilon}$	620	10	843	8
$10^{-8}\sqrt{\varepsilon}$	756	10	925	8

Here Example I is the matrix $A = 10^4 \cdot \mathrm{diag}(1, 1/2, 1/3, \ldots, 1/1000)$ and Example II is the matrix $A = \mathrm{diag}(100, 49.5, 48.5, \ldots, -49.5)$ both with $q_1^* = (1, 1, \ldots, 1)/10$ as starting vector. Although the figures in the table look rather similar, the two examples are quite different. Example II has a uniform and equally spaced eigenvalue distribution, whereas the eigenvalues in Example I have a large relative separation at one end of the spectrum and are clustered at the other. The minimum number of orthogonalizations occurs in both cases for $\eta = 10^{-4}\varepsilon$.

There is however a second cost factor which we have ignored so far. For large examples it will not be possible to keep the Lanczos vectors in fast storage. They have to be written into secondary storage, and every time some of them are needed one has to scan through all the Lanczos vectors. The cost of the recall operation will depend strongly on the system which is used and it is therefore difficult to compare it to savings in the orthogonalizations. The numbers in Table 7.1 suggest that the number of recall operations or rewinds of the tape with the Lanczos vectors is constant as long as $\eta \leqslant 10^{-5}\sqrt{\varepsilon}$ and then increases only slowly. Therefore the optimal choice for η regarding both cost factors lies somewhere between $10^{-5}\sqrt{\varepsilon}$ and $10^{-4}\sqrt{\varepsilon}$, regardless of the precise relation between both cost factors. In order to determine an η independent from the machine used, we suggest $\eta = \varepsilon^{3/4}$. On the VAX 11/780 this choice yields $\eta \approx 0.2274*10^{-12}$, which is slightly smaller than $10^{-4}\sqrt{\varepsilon} \approx 0.3725*10^{-12}$. This also seems to be a satisfactory choice in the sense that

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

$\eta = \varepsilon^{3/4}$ is "halfway" between $\sqrt{\varepsilon}$ (semiorthogonality) and ε (orthogonality to working precision) on a logarithmic scale. The examples from Table 7.1 were run again with this η, and the following results were obtained.

TABLE 7.2
Results with $\eta = \varepsilon^{3/4}$.

	No. of Orthogonalizations	No. of Recalls
Example I	501	11
Example II	607	10

The figures in Table 7.2 indicate that $\eta = \varepsilon^{3/4}$ yields almost the minimum number of both orthogonalizations and recalls. However these data have been gathered only for one fixed value of ε. Thus $\eta = \varepsilon^{3/4}$ and $\eta = 1000\varepsilon$ have about the same numerical value but a totally different dependence of ε. The numerical test described above was therefore repeated on the UNIVAC 1100 of the Computing Center of SUNY, Stony Brook using single and double precision. Here the values for the roundoff unit are $\varepsilon_s \approx 0.2980{*}10^{-7}$ and $\varepsilon_d \approx 0.3469{*}10^{-17}$. For these different values of ε and the matrix of Example I the following table was obtained.

TABLE 7.3
Influence of the lower bound η *on the reorthogonalizations.*
(Different Roundoff Units)

	Example I tested with $\varepsilon \approx 0.2980 \times 10^{-7}$, $\varepsilon^{\frac{3}{4}} \approx 0.2268 \times 10^{-5}$		Example I tested with $\varepsilon \approx 0.3469 \times 10^{-17}$, $\varepsilon^{\frac{3}{4}} \approx 0.8039 \times 10^{-13}$	
η	Orthogonalizations	Recalls	Orthogonalizations	Recalls
$\sqrt{\varepsilon}$	783	26	604	25
$10^{-1}\sqrt{\varepsilon}$	668	24	462	21
$10^{-2}\sqrt{\varepsilon}$	622	20	421	16
$10^{-3}\sqrt{\varepsilon}$	755	17	410	12
$10^{-4}\sqrt{\varepsilon}$	862	16	383	10
$10^{-5}\sqrt{\varepsilon}$			412	9
$10^{-6}\sqrt{\varepsilon}$			408	8
$10^{-7}\sqrt{\varepsilon}$			430	8
$10^{-8}\sqrt{\varepsilon}$			516	8
$10^{-9}\sqrt{\varepsilon}$			554	8
$\varepsilon^{\frac{3}{4}}$	668	18	429	9

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

The results reported in this table confirm that also in a different computing environment $\eta = \varepsilon^{3/4}$ is optimal in the sense discussed above. This choice of η finally determines against which previous Lanczos vectors the current Lanczos vector has to be orthogonalized, and thus completes the definition of partial reorthogonalization.

A good insight into the mechanism of PRO can be gained from Figures 7.3 and 7.4. Horizontal bars indicate the "batches" of Lanczos vectors against which the current Lanczos vector is orthogonalized. The double appearance of the bars corresponds to the fact that orthogonalizations are always carried out for two consecutive steps.

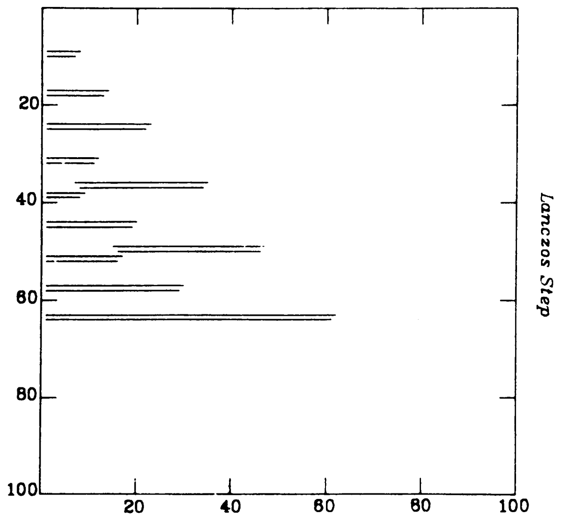

FIGURE 7.3
Range of reorthogonalizations for Example I, $\eta = \varepsilon^{3/4}$.

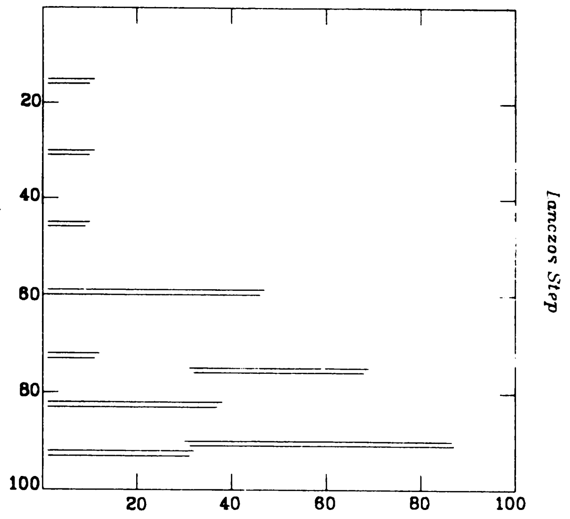

FIGURE 7.4
Range of reorthogonalizations for Example II, $\eta = \varepsilon^{3/4}$.

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

Let us finally summarize the results of the discussion of PRO in the form of the following algorithm:

TABLE 7.4
Algorithm for partial reorthogonalization.

```
Parameters:

  $r_i$= index, which indicates the first vector in batch i
  $s_i$= index, which indicates the last vector in batch i

Initialize:

  first step ← true
  $r_i$ ← 0
  $s_i$ ← 0

Subroutine PRO at the j-th Lanczos step:

  1.) for $k=1, \cdots j$ do
        update the recurrence for $\omega_{j+1\,k}$

  2.) if (first step) then
        begin for $k = 1, \cdots j$ do
          if ($|\omega_{j+1\,k}| \geq \sqrt{\varepsilon}$) then
            determine $r_i$ and $s_i$, such that $|\omega_{j+1\,l}| > \eta$,
            where $l=r_i,r_i+1, \cdots k-1,k,k+1, \cdots s_i-1,s_i$
          if (all $r_i$ and $s_i$ are 0) then return
        end

  3.) for $l=r_1,r_1+1, \cdots s_1-1,s_1,r_2,r_2+1, \cdots ,s_2-1,s_2,r_3,....$ do
        orthogonalize $\beta'_{j+1}q'_{j+1}$ against $q_l$

  4.) if (firststep) then
        first step ← false; $r_i$ ← $r_{i+1}$; $s_i$ ← $s_{i-1}$
        else first step ←true; $r_i$ ← 0; $s_i$ ← 0
```

8. Some More Details on PRO. There are two more topics to be discussed in relation with PRO. One concerns the question of the effect of PRO on the inner products of q_{j+1} with those previous Lanczos vectors against which the current Lanczos vector is *not* orthogonalized. Let q'_{j+1} be the current Lanczos vector before reorthogonalization and (compare 4.2)

$$q_{j+1} = q'_{j+1} - \sum_{k \in L(j)} \left(q'^*_{j+1} q_k\right) q_k. \tag{8.1}$$

Then for q_l, $l \notin L(j)$:

$$q^*_{j+1} q_l = q'^*_{j+1} q_l - \sum_{k \in L(j)} \left(q'^*_{j+1} q_k\right)\left(q^*_k q_l\right). \tag{8.2}$$

Since semiorthogonality is maintained, we know that $|q^*_k q_l| \leqslant \sqrt{\varepsilon}$, and also that $|q'^*_{j+1} q_k| \approx \sqrt{\varepsilon}$. Hence

$$q^*_{j+1} q_l = q'^*_{j+1} q_l + O\left(|L(j)| \varepsilon \|A\|\right), \tag{8.3}$$

and we do not have to worry that the level of orthogonality between the Lanczos vectors unaffected by PRO may deteriorate. A similar argument was used for SO and the corresponding Ritz vectors (Parlett [15, p. 281]).

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

Finally we want to mention that there is an easy way of avoiding the second of the two consecutive recalls of the Lanczos vectors by utilizing (7.1). Suppose at the jth step we orthogonalized q_{j+1} against q_k. Then at the $(j+1)$st step by (7.1)

$$\beta'_{j+2}q'^{*}_{j+2}q_k = -\beta_{j+1}q^{*}_{j}q_k + O(\varepsilon\|A\|). \tag{8.4}$$

Since we orthogonalize in batches the inner products $q_j^*q_{k+1}$ and $q_j^*q_{k-1}$ are also at roundoff level, and we obtain (8.4) for *all* vectors in the interior of the batches. We do not have to be concerned about the two vectors, which border the batch, because we do not orthogonalize against them at the $(j+1)$st step anyway. Therefore it is possible to compute *at the jth step* the vector

$$y_j = -\beta_{j+1}\sum_k (q_j^*q_k)q_k, \tag{8.5}$$

where we sum over all $k \in L(j)$ which are not on the edge of the batch. Then at the $(j+1)$st step the second orthogonalization simply becomes

$$\beta_{j+2}q_{j+2} = \beta'_{j+2}q'_{j+2} - y_j. \tag{8.6}$$

Thus at the cost of one extra n-vector the second recall of the Lanczos vectors is saved. There are however no savings in terms of arithmetic operations. This device is therefore only useful if the recall of the Lanczos vectors is expensive.

9. Comparison with Other Orthogonalization Methods.

9.1. *Periodic Reorthogonalization.* Grcar's [4] periodic reorthogonalization and PRO are quite closely related, however there are two main differences: in periodic reorthogonalization a full N-vector has to be updated versus only a j-vector for PRO, and whenever the threshold $\sqrt{\varepsilon}$ is reached the current Lanczos vector is made orthogonal to *all* previous q's versus only some previous ones in PRO. PRO is therefore more economical than partial reorthogonalization. On average PRO needs about $\frac{2}{3}$ of the number of orthogonalizations of periodic reorthogonalization (see Figures 7.3 and 7.4). For a comparison of PRO with FRO see Section 10.

9.2. *Selective Orthogonalization.* Selective orthogonalization (SO) was briefly discussed in the introduction as an alternative method for maintaining semiorthogonality. We assume here that the reader is familiar with the method of selective orthogonalization ([18], for an exposition see [15]). Practical numerical experience with SO for eigenvalue problems (Nour-Omid, Parlett, and Taylor [9]) and for the solution of linear systems (Nour-Omid [8]) shows that SO works very efficiently. Since SO maintains orthogonality with respect to the Ritz vectors rather than with respect to the Lanczos vectors a direct theoretical comparison of how both methods go about maintaining semiorthogonality is difficult. Numerical tests reported in [20] show that PRO cannot be easily explained in terms of the Ritz vectors. Reorthogonalizations in PRO mainly occurred in the direction of the dominant Ritz vector, but there was also a not insignificant component in direction of the other Ritz vectors. PRO reduces these relatively small components together with the needed orthogonalization in direction of the dominant Ritz vector. It therefore prevents the growth of the level of orthogonality in the direction of those Ritz vectors already at an early stage, and orthogonalizations against the second or third Ritz vector (as one would expect in SO) occur only in a hidden way in PRO.

The cost of PRO and SO are in general comparable, and it appears that neither method has a clear edge over the other one. There are however examples [20] where

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

PRO is much more efficient than SO and vice versa. These examples seem to indicate that PRO is more advantageous for solving linear systems of equations, whereas SO is more appropriate for the eigenvalue problem. This conclusion is preliminary and has to be fortified by more numerical evidence.

10. Numerical Examples. The Lanczos algorithm with partial reorthogonalization (LANPRO) as described in the previous sections was used for solving several large sparse symmetric systems of linear equations. The solution algorithm is mentioned in Section 2 and discussed in detail in [16] and [20]. The first three examples arise from finite element approximations to problems in structural engineering. The corresponding stiffness matrices were computed using the finite element approximation program FEAP [24, Chapter 23]. The other two examples are derived from finite difference approximations to elliptic partial differential equations. The examples and the characteristics of the resulting matrix problems are described in more detail in the following tables.

TABLE 10.1. *Examples.*

Example No.	Problem/Right Hand Side
1.1	Biharmonic operator on a beam with one end free and one end fixed. Finite element approximation using 80 elements with 3 degrees of freedom per node. Unit load at about the middle of the beam.
1.2	Same as 1.1, but using 240 elements.
2	Biharmonic operator on a rectangular plate with one side fixed and the others free. Unit load at one of the free corners.
3	Building modelled by the structure in Figure 10.1; each beam is approximated as in Example 1. Unit load vector.
4	Poisson's equation in an L-shaped region with mixed boundary conditions.The same problem as Example 4 in [5],only using less grid points.
5	$-\nabla(a\nabla u) = f$ in the unit cube in R^3 with Dirichlet boundary conditions. a is varying from 10^{-2} to 10^{6}. Finite difference discretization using 7-point difference star and 9x9x9 grid points. Random right hand side.

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

TABLE 10.2. *Properties of the Test Matrices.*

Example	Order	Nonzeros	Half Bandwidth	Fill-In	Cond.Number
1.1	237	627	4	1175	$2.0 * 10^7$
1.2	957	2547	4	4775	$1.3 * 10^9$
2	960	8402	43	66612	$3.5 * 10^3$
3	468	2820	178	38232	$1.1 * 10^4$
4	675	1965	30	9929	$7.1 * 10^6$
5	729	2673	81	40961	$1.8 * 10^9$

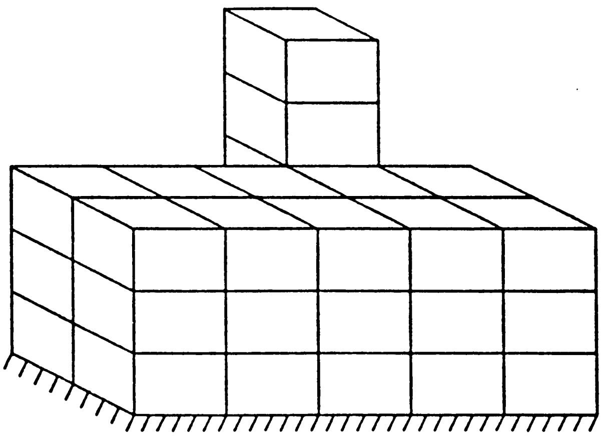

FIGURE 10.1. *Building.*

In the second column of Table 10.2 we list the number of nonzeros in the upper triangular part of the matrices, and in the third column the half bandwidth, where no attempt was made to reduce the bandwidth by reordering the matrix. In the fourth column we list the number of nonzeros in the Choleski factor obtained after a minimum degree ordering of the original matrix. All matrices are positive definite.

In a first series of tests we applied our algorithm to the six systems of linear equations as specified above, and compared its cost to the Lanczos algorithm with full reorthogonalization (FRO). The results are listed in the following Table 10.3, where we indicate the number of inner products used for the different parts of the algorithm. In each case the algorithm was stopped when the initial residual was reduced by a factor of 10^{-8}.

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

TABLE 10.3. *Cost of PRO versus FRO.*

Example	Number of inner products for						$\frac{PRO}{FRO}$
	Lanczos Step	Matrix Mult.	PRO	FRO	tot. PRO	tot. FRO	
1.1	959	420	7354	25172	8735	26502	0.33
1.2	3839	2762	205113	407682	211715	414283	0.51
2	1487	2170	4943	61256	8607	64913	0.13
3	2082	2084	13746	119370	17914	123537	0.15
4	1253	1007	17004	43472	19266	445733	0.24
5	2608	2748	124246	187922	129603	193279	0.67

The figures in Table 10.3 show that there are considerable savings in applying PRO as compared to FRO. There is apparently a direct correlation between the condition number of the matrix and the number of orthogonalizations which are necessary in order to maintain semiorthogonality, as a comparison of the corresponding columns in Tables 10.2 and 10.3 shows. Since several of our examples are very ill-conditioned the numbers in Table 10.3 can be considered as a worst case. In general for moderately well-conditioned matrices as in Examples 2, 3, and 4 we can expect that total cost of LANPRO is only about 20% of the cost of the Lanczos algorithm with full reorthogonalization.

In order to appreciate LANPRO it is also important to note that in spite of the ill-conditioned matrices, LANPRO showed a very robust behavior in all test cases. This is best illustrated in Figure 10.2, where we compared the residual norms for runs of LANPRO and conjugate gradients for Example 1.2. According to [23] this is one of the most difficult problems for an iterative solver. LANPRO terminates here after 638 steps, whereas the conjugate gradient algorithm needs 14,169 steps.

The maintenance of semiorthogonality among the Lanczos vectors yields, as expected, a large reduction in the number of necessary steps. It also guarantees termination of the Lanczos algorithm after at most n steps. In contrast to that the finite precision CG algorithm may not terminate at all.

Another advantage of the Lanczos algorithm is that it can be applied equally well to indefinite problems. We ran LANPRO with the matrix from Example 1.1, and introduced a shift of -2000 in order to make the system indefinite. Because CG in general is not applicable to indefinite problems, we compared our algorithm here with the algorithm SYMMLQ (cf. [14]). The indefiniteness caused no problem for LANPRO, and it compared very favorably in cost with SYMMLQ which is more expensive per step than conjugate gradients and in this case needed 2003 steps for convergence. The reduction in residual norm for this test run is shown in Figure 10.3.

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

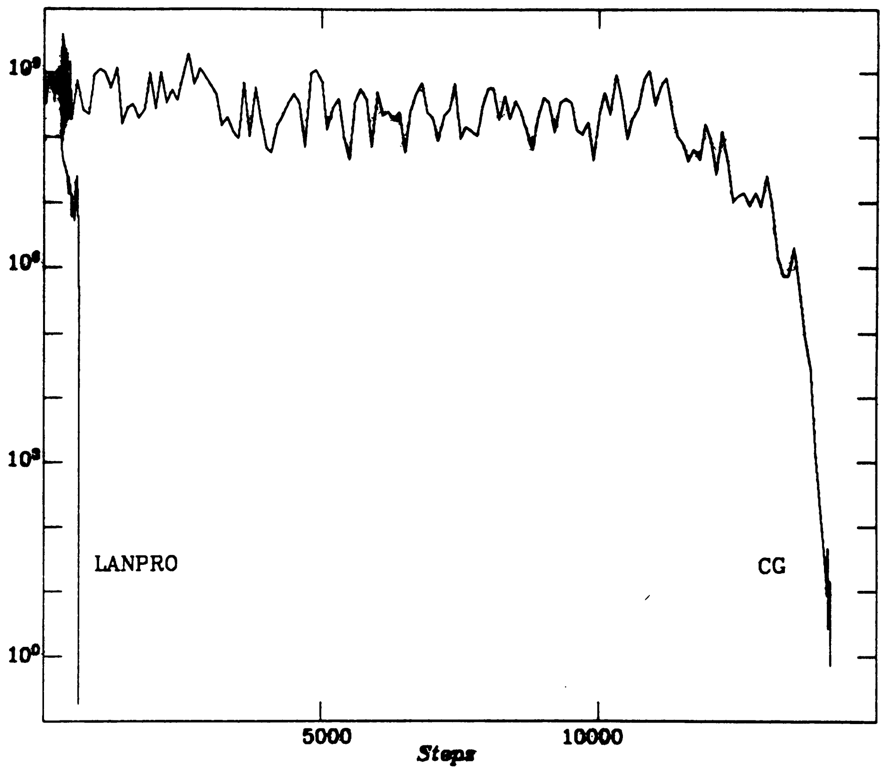

FIGURE 10.2
Residual Norms for Example 1.2.

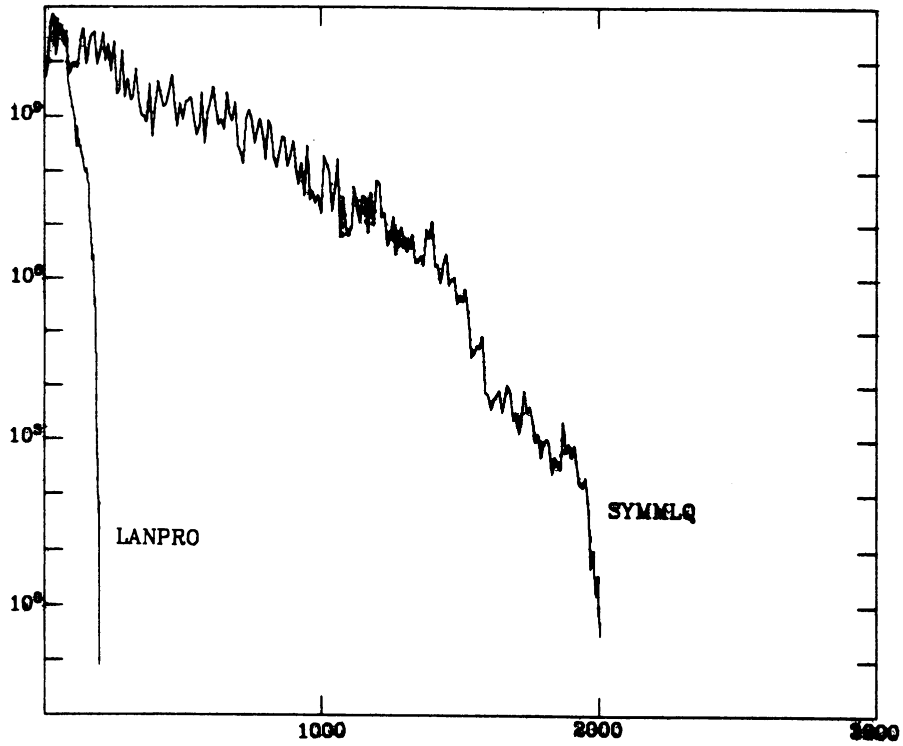

FIGURE 10.3
Residual Norms for Example 1.1 *shifted by* −2000.

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

These comparisons of LANPRO and other iterative methods stress the robustness and range of applications of LANPRO. In order to discuss the cost effectiveness of LANPRO we have to compare it to one of the most effective methods now in use: the preconditioned conjugate gradient algorithm. Since on an ideal level both conjugate gradients and Lanczos algorithm are identical, we can apply the same type of preconditioning to both and then compare their behavior. Here we choose as preconditioning the incomplete Choleski factorization routine MA31 from the Harwell Library by Munksgaard [7]. For our set of test matrices we obtained the following results:

TABLE 10.4

Comparison of Preconditioned LANPRO with Preconditioned CG.

		CG		LANPRO			
Example	NZL	Steps	Cost	Steps	Cost	Cost of Orth.	Ratio of Cost
1.1	1175	13	195	13	289	46	0.67
1.2	4775	32	468	32	670	114	0.70
2	23976	24	1204	22	1546	148	0.78
3	7610	42	1410	29	1517	260	0.93
4	3817	7	119	7	177	22	0.67
5	7407	14	339	13	420	40	0.81

The first column in Table 10.4 lists the number of nonzeros in the incomplete Choleski factor. We set the parameter C in MA31 to 10^{-2}, i.e. during the factorization all fill-in which was less than a hundredth of the corresponding diagonal elements was dropped. Since the cost of the incomplete factorization is the same for both algorithms it is not included in Table 10.4. For both LANPRO and CG we listed the number of steps which were necessary to reduce the residual norm by a factor of 10^{-8}, and the total cost given by the number of inner products. Note that the number of nonzeros in the incomplete Choleski factor strongly affects the total cost of both algorithms, as Example 2 shows. For LANPRO we also listed the part of the total number of operations that was spent for orthogonalizations. LANPRO performed orthogonalizations in all examples, however these orthogonalizations did not lead to a significantly faster convergence of the method. Hence LANPRO turned out to be slightly more expensive in all examples considered. The ratio of costs is best for LANPRO in the two- and three-dimensional examples. One can expect that for larger and more structured examples, the incomplete Choleski factor will have a larger number of nonzeros. For these cases LANPRO will be more advantageous than CG.

In order to illustrate the behavior of both algorithms we show below plots of the residual norm versus the number of steps and versus the number of operations for Examples 2 and 3.

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

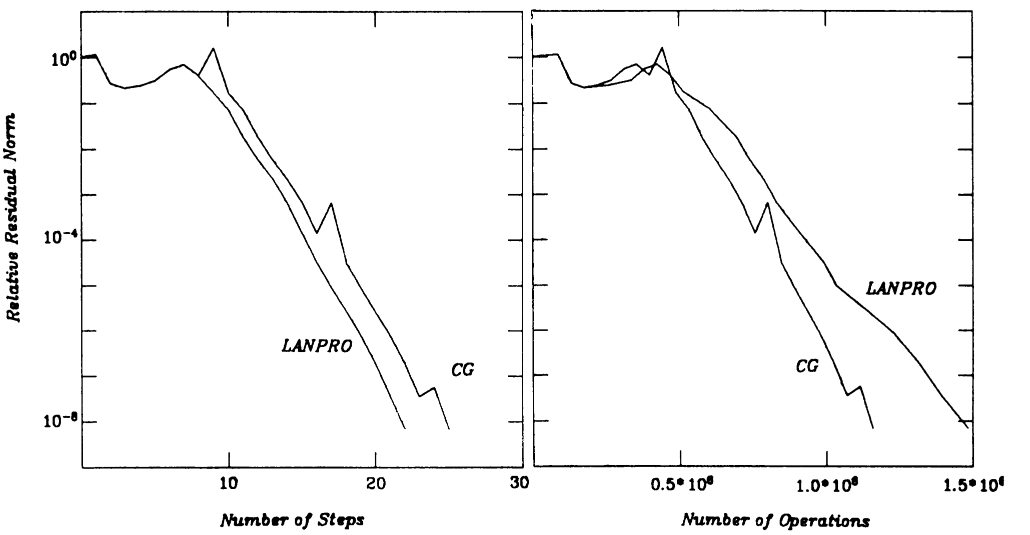

FIGURE 10.4
Residual norms for Example 2 with preconditioning.

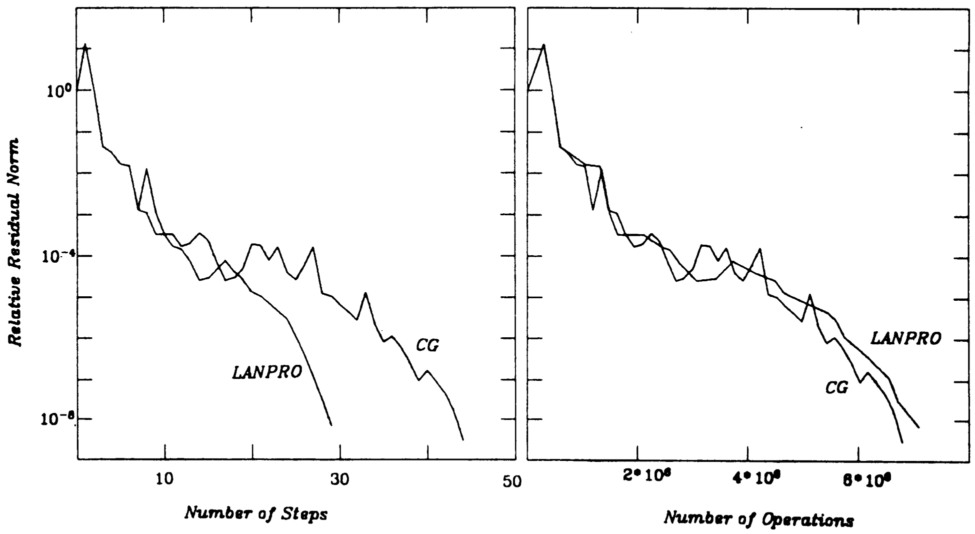

FIGURE 10.5
Residual norms for Example 3 with preconditioning.

In many applications the same problem has to be solved for several right-hand sides. Since LANPRO keeps the semiorthogonal Lanczos vectors, they can be easily used to compute first approximations to the solution for consecutive right-hand sides. CG on the other hand does not have this information directly available, and thus has to start anew with each problem. We consider here two common situations. As a typical structural engineering problem we solved the equations from Example 1.2 for 20 consecutive right-hand sides, which were just unit loads applied to 20 neighboring elements. In LANPRO we computed an initial approximation from the subspace spanned by the Lanczos vectors from the first run and used it as a starting guess. It then took only four iterations to obtain a reduction of the residual norm by 10^{-8}, and no more orthogonalizations were necessary. For CG we used the solution

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

from the previous right-hand side as a starting guess. Here it took between 11 and 13 iterations to obtain the same reduction in the residual norm. If the number of operations is plotted against the number of right-hand sides for both methods the following graph is obtained.

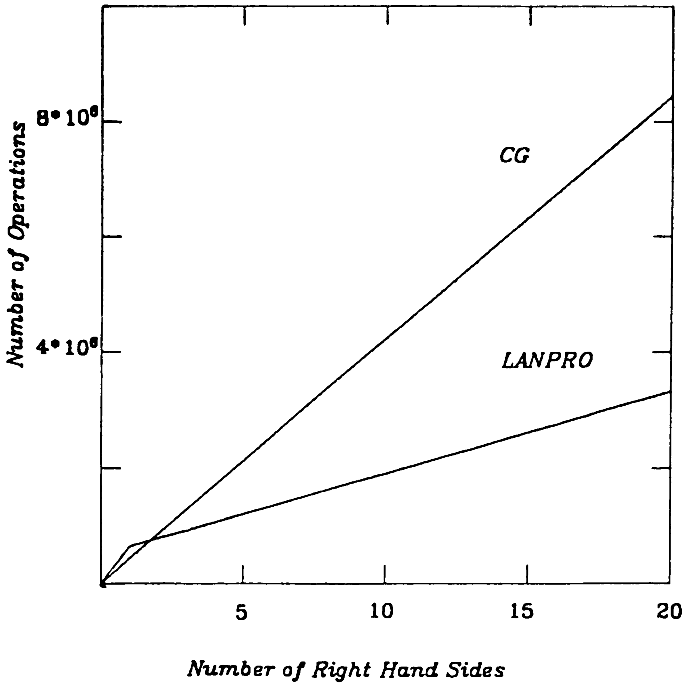

FIGURE 10.6
Comparison for consecutive right-hand sides, Example 1.2.

Figure 10.6 shows the clear advantage of LANPRO if the system has to be solved for consecutive right-hand sides. When two right-hand sides are present, the performances of LANPRO and CG are roughly equal. Since the cost for all further right-hand sides is constant in both algorithms, LANPRO's relative efficiency increases with the number of right-hand sides to be treated.

A similar situation occurs if parabolic partial differential equations are solved for several time steps using an implicit method in time direction. We modeled this by solving the linear system arising in Example 5 with the identity matrix added for 20 consecutive right-hand sides. The new right-hand side was chosen as the solution from the previous run. We proceed as above for Example 1.2. In this case, LANPRO has the additional advantage that the projection of the new right-hand side on the subspace spanned by the Lanczos vectors does not have to be computed, since it is already known from the previous run.

This numerical experiment showed results similar to those above. For all consecutive right-hand sides, LANPRO needed 5 iteration steps, whereas CG needed 8. The corresponding graph is given in Figure 10.7.

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

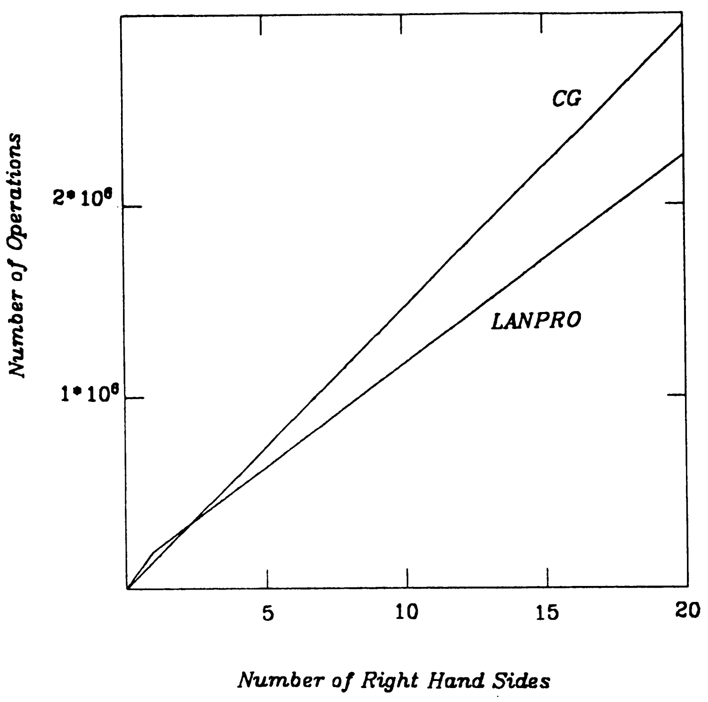

FIGURE 10.7
Comparison for consecutive right-hand sides, Example 5.

Figure 10.7 shows qualitatively the same behavior of LANPRO as Figure 10.6. Again, for two consecutive right-hand sides the cost of LANPRO and CG are about the same. The nice feature about LANPRO now becomes very clear. For a little extra cost we are able to produce a semiorthogonal basis for the Krylov subspace, which can be exploited for consecutive right-hand sides. The updating and monitoring of the recurrence and the occurrence of a few reorthogonalizations are a small price to pay for it and, yet, LANPRO compares favorably with CG in terms of cost.

Let us draw some conclusions from the observed behavior of the Lanczos algorithm with partial reorthogonalization. The above examples show that LANPRO

- □ finds a solution in $\leqslant n$ steps,
- □ is very economical for the treatment of several right-hand sides,
- □ can handle definite and indefinite problems equally well.

Acknowledgement. I wish to thank Professor B. N. Parlett for his support of this work. His suggestions and criticism improved the contents and presentation of this paper considerably.

Boeing Computer Services Company, MS 9C-01
565 Andover Park West
Tukwila, Washington 98188

1. J. CULLUM & R. WILLOUGHBY, "Lanczos and the computation in specified intervals of the spectrum of large sparse real symmetric matrices," in *Sparse Matrix Proceedings* (I. Duff and G. W. Stewart, eds.), SIAM, Philadelphia, Pa., 1979.

2. J. CULLUM & R. WILLOUGHBY, "Computing eigenvectors and eigenvalues of large sparse symmetric matrices using Lanczos tridiagonalization," in *Numerical Analysis Proceedings, Dundee* 1979 (G. A. Watson, ed.), Springer-Verlag, Berlin, 1980.

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

3. G. Golub, R. Underwood & J. H. Wilkinson, *The Lanczos Algorithm for the Symmetric $Ax = \lambda Bx$ Problem*, Tech. Rep. STAN-CS-72-720, Comp. Sci. Dept., Stanford University, 1972.

4. J. Grcar, *Analyses of the Lanczos Algorithm and of the Approximation Problem in Richardson's Method*, Ph.D. thesis, University of Illinois at Urbana-Champaign, 1981.

5. D. S. Kershaw, "The incomplete Choleski-conjugate gradient method for the iterative solution of systems of linear equations," *J. Comput. Phys.*, v. 24, 1978, pp. 43–65.

6. C. Lanczos, "Solution of systems of linear equations by minimized iterations," *J. Res. Nat. Bur. Standards*, v. 49, 1952, pp. 33–53.

7. N. Munksgaard, "Solving sparse symmetric sets of linear equations by preconditioned conjugate gradients," *ACM TOMS*, v. 6, 1980, pp. 206–219.

8. B. Nour-Omid, *A Newton-Lanczos Method for Solution of Nonlinear Finite Element Equations*, Report UCB/SESM-81/04, Dept. of Civil Engineering, University of California, Berkeley, 1981.

9. B. Nour-Omid, B. N. Parlett & R. Taylor, "Lanczos versus subspace iteration for the solution of eigenvalue problems," *Internat. J. Numer. Methods Engrg.*, v. 19, 1983, pp. 859–871.

10. C. Paige, *The Computation of Eigenvalues and Eigenvectors of Very Large Sparse Matrices*, Ph.D. Thesis, Univ. of London, 1971.

11. C. Paige, "Computational variants of the Lanczos method for the eigenproblem," *J. Inst. Math. Appl.*, v. 10, 1972, pp. 373–381.

12. C. Paige, "Error analysis of the Lanczos algorithm for tridiagonalizing a symmetric matrix," *J. Inst. Math. Appl.*, v. 18, 1976, pp. 341–349.

13. C. Paige, "Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem," *Linear Algebra Appl.*, v. 34, 1980, pp. 235–258.

14. C. Paige & M. Saunders, "Solution of sparse indefinite systems of linear equations," *SIAM J. Numer. Anal.*, v. 12, 1975, pp. 617–629.

15. B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, N. J., 1980.

16. B. N. Parlett, "A new look at the Lanczos algorithm for solving symmetric systems of linear equations," *Linear Algebra Appl.*, v. 29, 1980, pp. 323–346.

17. B. N. Parlett & J. K. Reid, "Tracking the progress of the Lanczos algorithm for large symmetric eigenproblems," *IMA J. Numer Anal.*, v. 1, 1981, pp. 135–155.

18. B. N. Parlett & D. Scott, "The Lanczos algorithm with selective orthogonalization," *Math. Comp.*, v. 33, 1979, pp. 217–238.

19. D. S. Scott, *Analysis of the Symmetric Lanczos Process*, Ph.D. thesis, Dept. of Math., University of California, Berkeley, 1978.

20. H. D. Simon, *The Lanczos Algorithm for Solving Symmetric Linear Systems*, Report PAM-74, Center for Pure and Appl. Math., Univ. of California, Berkeley, 1982.

21. H. Takahasi & M. Natori, *Eigenvalue Problem of Large Sparse Matrices*, Rep. Comp. Center, Univ. Tokyo, No. 4, 1971-1972, pp. 129–148.

22. J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.

23. E. Wilson, private communication.

24. O. C. Zienkiewicz, *The Finite Element Method*, 3rd ed., McGraw-Hill, London, 1977.

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use