

AD-A118 014

CALIFORNIA UNIV BERKELEY CENTER FOR PURE AND APPLIED--ETC F/G 12/1
THE LANCZOS ALGORITHM FOR SOLVING SYMMETRIC LINEAR SYSTEMS.(U)
APR 82 H D SIMON

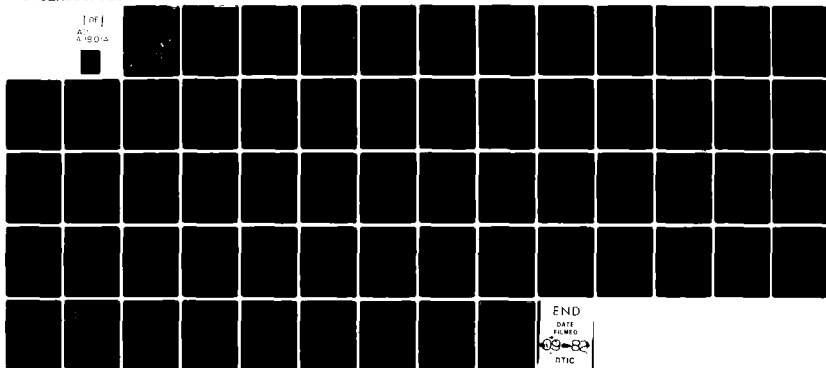
N00014-76-C-0013

UNCLASSIFIED

PAM-74

NL

[OF]
A-118 014



END
DATE
FILMED
69-82
NTIC

CENTER FOR PURE AND APPLIED MATHEMATICS
UNIVERSITY OF CALIFORNIA, BERKELEY

PAM-74

The Lanczos Algorithm for
Solving Symmetric Linear Systems

Horst D. Simon

DTIC
ELECTE
AUG 9 1982
S H

DTIC FILE COPY

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

20 April 1982

82 08 09 008

Legal Notice

This report was prepared as an account of work sponsored by the Center for Pure and Applied Mathematics. Neither the Center nor the Department of Mathematics, makes any warranty expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information or process disclosed.

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)

Mathematics Department U.C. Berkeley

2a. REPORT SECURITY CLASSIFICATION

UNCLASSIFIED

2b. GROUP

3. REPORT TITLE

The Lanczos Algorithm for Solving Symmetric Linear Systems

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Scientific Final

5. AUTHOR(S) (First name, middle initial, last name)

Horst D. Simon

6. REPORT DATE

April 20, 1982

7a. TOTAL NO. OF PAGES

109

7b. NO. OF REFS

42

8a. CONTRACT OR GRANT NO.

N 00014-76-C-0013

8b. PROJECT NO.

9a. ORIGINATOR'S REPORT NUMBER(S)

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited

11. SUPPLEMENTARY NOTES

TECH OTHER

12. SPONSORING MILITARY ACTIVITY

Office of Naval Research

13. ABSTRACT

The Lanczos Algorithm is becoming accepted as a powerful tool for finding the eigenvalues and eigenvectors of large sparse matrices. This dissertation considers the application of the Lanczos algorithm to the solution of large sparse symmetric systems of linear equations. We analyze the symmetric Lanczos process with various reorthogonalization methods, and present a new implementation of the algorithm, which efficiently maintains orthogonality among the Lanczos vectors. This new algorithm is discussed in detail, compared to other methods, and tested with some numerical examples.

DD FORM 1473
1 NOV 83

Security Classification

THE LANCZOS ALGORITHM FOR SOLVING SYMMETRIC LINEAR SYSTEMS *

Herst D. Simon
Dept. of Mathematics
University of California, Berkeley
Ph.D. Dissertation

April 1982

ABSTRACT

The Lanczos Algorithm is becoming accepted as a powerful tool for finding the eigenvalues and eigenvectors of large sparse matrices. This dissertation considers the application of the Lanczos algorithm to the solution of large sparse symmetric systems of linear equations. We analyze the symmetric Lanczos process with various reorthogonalization methods, and present a new implementation of the algorithm, which efficiently maintains orthogonality among the Lanczos vectors. This new algorithm is discussed in detail, compared to other methods, and tested with some numerical examples.

B.N. Parlett

Professor B.N. Parlett
Chairman of Thesis Committee

* Research sponsored by Office of Naval Research Contract N00014-78-C-0013

ACKNOWLEDGEMENTS

I would like to express my gratitude to Professor B.N. Parlett, who supervised this dissertation. He provided support and much valuable advice on this project. The many discussions I held with him were very instrumental in suggesting the directions of my research. I especially appreciate his generosity with his valuable time. I am also grateful to Professors F.A. Grünbaum and E. Wilson for reading this dissertation.

I would like to thank Babram Nour-Omid for many helpful discussions we had about the Lanczos algorithm and for providing the interesting numerical examples in Chapter 4.

Finally I wish to thank my wife Candice for her constant patience and support throughout my graduate studies.

DISTRIBUTION STATEMENT A
Approved for public release:
Distribution Unlimited

DTIC
UNCLASSIFIED
APR 9 1982

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	ii
INTRODUCTION	i
1. THE LANCZOS ALGORITHM IN EXACT ARITHMETIC	iv
1.1 Derivation and Properties	1
1.2 Conjugate Gradients and Related Methods for Solving Symmetric Systems	8
1.3 Convergence Properties	14
1.4 Updating the Residual Norm and Solving the Tridiagonal System	17
1.5 Initial Guess, Starting Vector, and Treatment of Several Right Hand Sides	21
2. ANALYSIS OF THE SYMMETRIC LANCZOS ALGORITHM IN FINITE PRECISION	24
2.1 A Mathematical Model of the Lanczos Algorithm in the Presence of Roundoff	24
2.2 The Loss of Orthogonality	29
2.3 Lemmas	33
2.4 Analysis of the Simple Lanczos Algorithm	38
2.5 Semiorthogonalization Strategies	46
2.5.1 Partial Reorthogonalization	50
2.5.2 Selective Orthogonalization	52
2.6 Applications	58

3. PARTIAL REORTHOGONALIZATION	iii
3.1 Computing the Level of Orthogonality	80
3.2 The Behavior of the Computed Level of Orthogonality	80
3.3 Choosing Reorthogonalizations	71
3.4 Some more Details on PRO	75
3.5 Comparison with Selective Orthogonalization	87
3.5.1 Maintenance of Semiorthogonality	89
3.5.2 Comparison of Costs	92
4. NUMERICAL EXAMPLES	95
REFERENCES	105

INTRODUCTION

In many applications one encounters the intermediate task of computing a solution vector x to the system of linear equations

$$Ax = b \quad (0.1)$$

where A is a symmetric, nonsingular $n \times n$ matrix and b is an n -vector. If A is large and sparse, there is an elegant way to exploit the sparsity by employing A only as a linear operator which computes Av for any given vector v . There are several methods known which produce an approximate solution vector based only on repeated computation of matrix vector products, e.g., the method of conjugate gradients (called hereafter CG) by Hestenes and Stiefel [12], Lanczos' [18] method of minimized iterations (called hereafter LAN), and the algorithm SYMMLQ by Paige and Saunders [28].

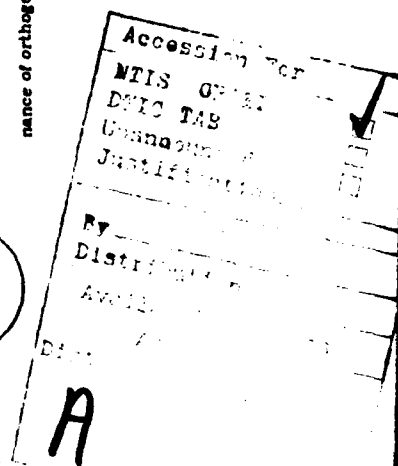
These methods have several attractive features in common. There are no special properties needed for A (except positive definiteness for CG), no acceleration parameters have to be estimated, and the fast storage requirements are only a few n -vectors in addition to the demands of the operator A .

Since Reid [33] pointed out these advantages CG has been widely used for solving sparse positive definite systems. By rethinking CG so that it could be applied to indefinite (i.e. neither positive nor negative definite) systems, Paige and Saunders created SYMMLQ. Both methods are iterative in nature, i.e. at each step a current approximate solution vector is updated until an estimate for the corresponding residual norm is smaller than a prescribed tolerance. In contrast to this updating feature the Lanczos algorithm computes a set of orthonormal vectors, the Lanczos vectors. Only at the end of a run is an approximate solution to (0.1) computed from the Lanczos vectors.

When λ is positive definite, it turns out that in exact arithmetic all these methods mentioned above produce exactly the same approximate solution. So why reintroduce a variant of an established algorithm, which - as it appears - has the disadvantage of requiring the storage of a large number of vectors? There are two arguments in favor of the Lanczos algorithm:

- 1) The availability of the Lanczos vectors makes it possible to compute approximate solutions for subsequent right hand sides at little cost, whereas for CG the iteration has to be carried out for each right hand side from the beginning.
- 2) Because of the influence of roundoff errors the actual implementations of these methods differ considerably from their ideal counterparts. The Lanczos vectors and the corresponding residual vectors in CG lose their orthogonality and may even become linearly dependent. One might expect that under these circumstances the algorithm is unstable and breaks down. But the loss of orthogonality does not prevent convergence, it only delays it. CG, which would terminate in exact arithmetic after at most n steps, may in practice well take many more than n steps for ill conditioned systems, but still produce a good solution. If orthogonality among the Lanczos vectors can be maintained at some reasonable cost, then LAN will minimize the number of calls on A and thus reduce the overall cost. Finally the recent idea of preconditioning can cut down significantly the number of steps needed (Mejerink and van der Vorst [21]; Kershaw [16]; Jennings and Malik [14]; Manteuffel [20]).

An implementation of the Lanczos algorithm therefore faces two crucial tasks: the storage of a certain number of Lanczos vectors and the maintenance of orthogonality among them. The problem of storing the Lanczos vec-



tors can be solved by using secondary storage. This should be fairly easy since the vectors are only needed from time to time and are always accessed sequentially.

The maintenance of orthogonality is more difficult. Traditionally (Wilkinson [41], Golub, Underwood, and Wilkinson [8]) it has been suggested to use full reorthogonalization of the Lanczos vectors at each step. This is very expensive for the size of problem considered here. Recently Parlett and Scott [32] introduced selective orthogonalization (SO) for the eigenvalue problem as an economical way of maintaining orthogonality among the Lanczos vectors. In [30] Parlett shows how the Lanczos algorithm with SO can be used for the solution of symmetric linear systems.

This thesis follows the program outlined in Parlett [30] and discusses in detail various aspects of the application of the Lanczos algorithm for solving (0.1) for large sparse systems. Its main contributions are a new understanding of the loss of orthogonality in finite precision arithmetic and a new reorthogonalization method which we call partial reorthogonalization (PRO) to distinguish it from Parlett and Scott's selective orthogonalization (SO).

In Chapter 1 the Lanczos algorithm in exact arithmetic is introduced and its relation to the method of conjugate gradients and the algorithm SYMMLQ is exhibited. The connections with various other methods for indefinite systems are discussed. An a priori error bound for this family of methods is derived, based on results obtained by Kaniel [15]. Some details of the algorithm are presented, and finally it is shown how the Lanczos algorithm lends itself to the treatment of several right hand sides.

Chapter 2 presents a readable error analysis of the symmetric Lanczos algorithm in finite precision arithmetic. The loss of orthogonality among the

computed Lanczos vectors can now be explained satisfactorily with the help of a recurrence formula, and Paige's Theorem [24] can be derived from this recurrence. A backward error analysis then shows that semiorthogonality among the Lanczos vectors is enough to guarantee the accuracy of the computed quantities up to machine precision. This is an improvement over results by Gear [9]. Finally various orthogonalization strategies are analyzed, and it is shown that selective orthogonalization as introduced by Parlett and Scott [32] indeed maintains semiorthogonality among the Lanczos vectors.

The results of Chapter 2 give rise to the already mentioned new reorthogonalization scheme for the Lanczos algorithm called PRO. It is based on a recurrence which allows one to monitor the loss of orthogonality among the Lanczos vectors directly without computing any inner products. Based on the information from the recurrence, reorthogonalizations are made only when necessary. Thus substantial savings are gained as compared to full reorthogonalization. Details of the practical implementation of PRO and a comparison with SO are discussed in Chapter 3.

The Lanczos algorithm with PRO is tested in Chapter 4. Several large systems (up to order 1000) of linear equations derived from finite element approximations to structural engineering problems are solved. The numerical results show that the Lanczos algorithm is especially useful when the matrix vector product dominates other costs, or when the system has to be solved for several right hand sides. In addition positive definite and indefinite systems can be solved with equal ease.

The results of Chapter 2 and 3 are also applicable to the eigenvalue problem. Therefore, whenever it is possible without distraction from the

main topic, immediate consequences of our results for the eigenvalue problem are stated.

Throughout this thesis the notation will follow Householder's convention: small Greek letters for scalars, small Roman letters for column vectors, capital Roman letters for matrices. Symmetric letters (A, M, V, W) will be reserved for symmetric matrices. All quantities are real unless otherwise noted. $\| \cdot \|$ denotes the Euclidean norm for vectors and the associated spectral norm for matrices.

1. THE LANCZOS ALGORITHM IN EXACT ARITHMETIC.

1.1. Derivation and Properties.

Many numerical methods are based on the idea of replacing a complicated or large problem by an easier and/or smaller one which is an approximation to the original problem in some subspace. This is also the basic idea behind the Lanczos algorithm applied to the solution of linear systems of equations. The subspace which will be considered here is a particular convenient and important subspace for methods in numerical linear algebra, the Krylov subspace K^j of R^n . Let A be a symmetric $n \times n$ matrix and b be a nonzero column vector then the Krylov subspace $K^j(b; A)$ is defined by

$$K^j(b; A) = \text{span} (b, Ab, A^2b, \dots, A^{j-1}b). \quad (1.1.1)$$

If no confusion is likely the arguments of K^j will be omitted. Unless b is specially related to A or $j > n$, K^j will have dimension j . On this assumption the Krylov vectors $b, Ab, A^2b, \dots, A^{j-1}b$ will form a basis for the space K^j . These vectors are easy to compute but for practical work it would be more desirable to have an orthonormal basis for K^j .

The Lanczos algorithm may now be described simply as the application of Gram-Schmidt orthogonalization to the Krylov vectors $b, Ab, A^2b, \dots$. At first glance this appears to involve many orthogonalizations, but there are two facts which make this task easy, and the Lanczos algorithm efficient.

Suppose that at the j -th step an orthogonal basis of Lanczos vectors $q_1, q_2, \dots, q_j$ for K^j has been computed. Then the next task would be to orthogonalize $A^j b$ against $q_1, \dots, q_j$. It turns out that it is equivalent to orthogonalize Aq_j against $q_1, \dots, q_j$. Furthermore Aq_j happens already to be orthogonal to $q_1, q_2, \dots, q_{j-1}$. This is where the symmetry of A is essential. So Aq_j has to be

orthogonalized only against q_{j-1} and q_j , i.e.

$$u_j = Aq_j - q_j \alpha_j - q_{j-1} \beta_j \quad (1.1.2)$$

where $\alpha_j = q_j^* A q_j$, $\beta_j = q_{j-1}^* A q_j$. u_j still has to be normalized to become q_{j+1} . It turns out that $\|u_j\| = \beta_{j+1}$. The core of a Lanczos step is therefore surprisingly simple. The algorithm will be started by setting $q_0 = 0$, $\beta_1 = \|b\|$, and $q_1 = b/\beta_1$. Note that $q_1, q_2, \dots, q_{j-2}$ are not needed in (1.1.2).

Because of its importance for the understanding of the behavior of the algorithm in the presence of roundoff errors we will show here that the Lanczos vectors $q_1, q_2, \dots, q_j$ computed by (1.1.2) indeed form an orthonormal basis for K^j , i.e. that

$$\text{span}(q_1, \dots, q_j) = K^j(b, A) \quad (1.1.3)$$

and

$$q_k^* q_l = \delta_{kl} \quad \text{for } k, l \leq j, \quad (1.1.4)$$

where δ_{kl} is the Kronecker delta. First note that this is true for $j=1$ and $j=2$, because Lanczos reduces to Gram-Schmidt. Suppose now (1.1.3) and (1.1.4) hold for general $j \geq 2$. Since by assumption $q_j \in K^j$, it can be written as a linear combination of the Krylov vectors, $q_j = \sum_{k=0}^{j-1} \gamma_k A^k b$, where the γ_k 's are some well determined coefficients. It follows that

$$\begin{aligned} \text{span}(q_1, \dots, q_j, q_{j+1}) &= \text{span}(q_1, \dots, q_j, (Aq_j - \alpha_j q_j - \beta_j q_{j-1})) \\ &= \text{span}(q_1, \dots, q_j, Aq_j) \\ &= \text{span}(q_1, \dots, q_j, A(\sum_{k=0}^{j-1} \gamma_k A^k b)) \\ &= \text{span}(q_1, \dots, q_j, A^j b) \\ &= K^{j+1}(b, A). \end{aligned} \quad (1.1.5)$$

In order to show that also (1.1.4) holds for $j+1$ it suffices to prove that

$$q_k^* q_{j+1} = 0 \quad \text{for } k = 1, \dots, j-2.$$

since $q_{j-1}^* q_{j+1} = 1$, $q_j^* q_{j+1} = 0$, $q_j^* q_{j-1} = 0$ by construction. For $1 \leq k \leq j-2$ we use (1.1.2) for the k -th Lanczos step to verify that

$$\begin{aligned} \beta_{j+1} q_k^* q_{j+1} &= q_k^* A q_j - \alpha_j - \beta_j q_{j-1}^* q_j = q_k^* A q_j \\ &= q_j^* A q_k = q_j^* (\beta_{k+1} q_{k+1} + \alpha_k q_k + \beta_k q_{k-1}) = 0. \end{aligned} \quad (1.1.6)$$

Hence the Lanczos vectors form an orthonormal basis.

The relationships defining the Lanczos algorithm can now be summarized in matrix form. Let the Lanczos vectors $q_1, \dots, q_j$ form the columns of the matrix $Q_j = (q_1, \dots, q_j)$ and denote by T_j the symmetric tridiagonal matrix

$$T_j = \begin{bmatrix} \alpha_1 & \beta_2 & 0 & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{j-1} & \alpha_{j-1} & \beta_j \\ & & 0 & \beta_j & \alpha_j \end{bmatrix} \quad (1.1.7)$$

then (1.1.2), (1.1.4), and (1.1.6) become (in this order):

$$A Q_j - Q_j T_j = \beta_{j+1} q_{j+1} q_j^* \quad (1.1.8a)$$

$$Q_j^* Q_j = I_j \quad (1.1.8b)$$

$$Q_j^* q_{j+1} = 0 \quad (1.1.8c)$$

Here q_j (or e_j) is the j -th column of the $j \times j$ identity matrix I_j .

The special choice of the Krylov subspace K^j gives yet another reward. Premultiplying (1.1.8a) by Q_j^* and simplifying one obtains

$$Q_j^* A Q_j = T_j. \quad (1.1.9)$$

Since $(Q_j Q_j^*)^* = Q_j Q_j^*$ and $(Q_j Q_j^*)^2 = Q_j Q_j^*$, the matrix $Q_j Q_j^*$ is an orthogonal

projector with the range $\text{span}(Q_j) = K^j$. Therefore we obtain the orthogonal projection of A onto K^j by restricting $(Q_j Q_j^T A Q_j Q_j^T)$ to K^j , i.e. by $Q_j^T A Q_j$. Hence the orthogonal projection of A onto $\text{span}(Q_j) = K^j$ is given by the triangular matrix T_j .

The next question is of course how to compute an approximate solution vector z_j from the Krylov subspace K^j . In terms of Q_j and T_j we can characterize z_j as follows

$$z_j = Q_j T_j^{-1} Q_j^T b. \quad (1.1.10)$$

Equation (1.1.10) says that z_j is obtained by projecting the right hand side b onto K^j , then solving in K^j the system $T_j f_j = Q_j^T b$, where T_j is the projection of A , and finally expressing f_j in terms of the original space by forming $z_j = Q_j f_j$. Note that $Q_j^T b = \alpha^{(j)} \beta_1$, since $g_1 = b / \beta_1$.

The approximate solution vector z_j can be characterized in another way. Consider the corresponding residual vector $r_j = b - A z_j$. Using (1.1.8a) one obtains

$$\begin{aligned} r_j &= b - A Q_j T_j^{-1} Q_j^T b \\ &= b - (Q_j T_j + \beta_{j+1} q_{j+1} e_j^T) T_j^{-1} Q_j^T b \\ &= -\beta_{j+1} q_{j+1} e_j^T T_j^{-1} \alpha^{(j)} \beta_1 \\ &= -\beta_{j+1} q_{j+1}. \end{aligned} \quad (1.1.11)$$

where q_j denotes the j -th component of the vector f_j . Thus the residual r_j is a scalar multiple of the vector q_{j+1} and by (1.1.8c) orthogonal to K^j . The approximate solution vector at the k -th step is therefore the unique vector $z_j \in K^j$ whose residual r_j is orthogonal to K^j . This suggests that z_j solves $\min_{z \in K^j} \|b - Az\|$. This is however not the case as will be shown in section 1.2.

By taking norms in (1.1.11) another key formula is obtained:

$$\|r_j\| = \beta_{j+1} \|\varphi_j\| \quad (1.1.12)$$

This shows that the norm of the j -th residual can be found without forming explicitly z_j or the residual. It is not even necessary to compute f_j at each step in order to find φ_j . There is a simple updating mechanism, which will be discussed in section 1.4. In order to check the progress of the algorithm only the scalar $\beta_{j+1} \|\varphi_j\|$ has to be monitored. Only when it becomes smaller than some given tolerance tol does the system $T_j f_j = \alpha^{(j)} \beta_1$ have to be solved. At this point it should be mentioned that if A is indefinite then for some j T_j may be close to singular and special precaution must be taken in solving this system. The details will be explained in section 1.4.

Finally the approximate solution z_j has to be computed. It is only here that the Lanczos vectors $q_1, q_2, \dots, q_{j+1}$ are needed again, because z_j has to be assembled by forming $z_j = Q_j f_j$. The Lanczos vectors will be written into secondary storage as soon as they are computed and, at the end, recalled one by one in order to form z_j . If a machine with virtual memory is used then the operating system takes care of the storage of unwanted Lanczos vectors instead of the Lanczos program.

We are now ready to give a condensed formulation of the Lanczos algorithm which needs only two n -vectors q and r for storage:

1. Initialization
 $q \leftarrow 0, r \leftarrow b, \beta_1 \leftarrow \|b\|$
2. Loop: for $j \leq \text{LMAX}$ do
 2.1 Perform Lanczos step
 2.2 Update residual norm $\|r_j\| = \beta_{j+1} \|y_j\|$
 2.3 Check: if $\|r_j\| \leq \text{tol}$ go to 3.
3. Solve $T_j f_j = e_1 \beta_1$
4. Assemble solution: $x_j = Q_j f_j$

Table 1.1. The Basic Lanczos Algorithm (LAN).

A Lanczos step will be performed as suggested by Paige[25]:

r	$\leftarrow r / \beta_j$
q	$\leftarrow -q \beta_j / \beta_j$
q	$\leftarrow q + \beta_j r$
	swap q and r
α_j	$\leftarrow q \cdot r$
r	$\leftarrow r - q \alpha_j$
β_{j+1}	$\leftarrow \ r\ $
	store q

Table 1.2. Lanczos Step.

An operation count in Table 1.2. shows that at each Lanczos step 5n operations (=multiplications/divisions) and one matrix-vector product are needed.

In addition to the Krylov and Lanczos vectors there is a third distinguished basis for the Krylov subspace K^j , the basis of Ritz vectors $y^{(j)}, y^{(j)}, \dots, y^{(j)}$. Consider the spectral factorization

$$T_j = S_j \Theta_j S_j^T \quad (1.1.13)$$

with $S_j^T = S_j^{-1}$, $\Theta_j = \text{diag}(\phi_1^{(j)}, \phi_2^{(j)}, \dots, \phi_j^{(j)})$. Then the Ritz vectors $y^{(j)}$ are defined as the columns of Y_j .

$$Y_j = Q_j S_j \quad (1.1.14)$$

and the corresponding $\phi^{(j)}$ are called Ritz values. Because of the orthonormality of Q_j and S_j the $y^{(j)}$ form an orthonormal basis for K^j . All Ritz vectors and values change from one Lanczos step to the next.

The Ritz values and vectors are approximations from K^j to the eigenvalues and vectors of A , which are optimal in the following sense: Let $Y = (y_1, y_2, \dots, y_j)$ be any $n \times j$ orthonormal matrix with $\text{span}(Y) = K^j$, and let $\Theta = \text{diag}(\theta_1, \theta_2, \dots, \theta_j)$. Then Y_j and Θ_j minimize $\|AY - Y\Theta\|$ over all such matrices Θ and Y , whose columns span K^j . This means that Ritz values and vectors are in this sense collectively the best approximation from K^j to the eigenvalues and vectors of A . A proof of this fact and a detailed description of the application of the Lanczos algorithm for eigenvalue problems are given in [28]. For our purposes the Ritz vectors are important only for the analysis of the finite precision Lanczos algorithm and for selective orthogonalization, both to be discussed in Chapter 2.

1.2. Conjugate Gradients and Related Methods for Solving Symmetric Systems.

Before the method of conjugate gradients is discussed in more detail, it is important to know that for positive definite A there is a third way of characterizing the approximate solution x_j from (1.1.10). In this case $\|s\|_A = \sqrt{s^T A s}$ defines a norm for all $s \in R^n$. x_j is then the unique vector, which minimizes the error in this norm over all vectors $s \in R^j$, i.e.

$$\|x - x_j\|_A^2 = \min_{s \in R^j} \|x - s\|_A^2 = \min_{s \in R^j} \{s^T A s - 2b^T s + b^T x\} \quad (1.2.1)$$

Since any vector $s \in R^j$ can be written $s = Q_j w$, with $w \in R^j$, one can also say that $x_j = Q_j f_j$, where $f_j \in R^j$ minimizes

$$\begin{aligned} \|x - Q_j w\|_A^2 &= w^T Q_j^T A Q_j w - 2b^T Q_j w + b^T x \\ &= w^T T_j w - 2w^T e_j \beta_j + b^T x \end{aligned} \quad (1.2.2)$$

over all $w \in R^j$. f_j then can be found by annihilating the gradient: $f_j = T_j^{-1} e_j \beta_j$, and this is a minimum since A is assumed to be positive definite. Hence $x_j = Q_j T_j^{-1} e_j \beta_j$ and this is precisely (1.1.10).

Similarly it is possible to derive a different method which computes a vector x_j^f which minimizes $\|b - A s\|$ over all $s \in R^j$.

$$\begin{aligned} \min_{s \in R^j} \|b - A s\|^2 &= \min_{s \in R^j} \{b^T b - 2b^T A s + s^T A^2 s\} \\ &= \min_{w \in R^j} \{\beta_j^2 - 2\beta_j e_j^T T_j w + w^T Q_j^T A^2 Q_j w\}. \end{aligned}$$

As before the minimum h_j can now be found by solving $(Q_j^T A^2 Q_j) h_j = \beta_j T_j e_j$, and x_j^f is given by $x_j^f = Q_j h_j$. Using (1.1.6) it turns out that $Q_j^T A^2 Q_j = T_j^2 + \beta_j^2 e_j e_j^T$. Hence x_j^f will in general be different from x_j .

The method of conjugate gradients is usually derived as a method for minimizing $\|x - s\|_A$. Suppose that, at the $(j-1)$ -st step, an approximate

solution vector x_{j-1}^{CG} and a search direction p_{j-1} are given. Then the next iterate is

$$x_j^{CG} = x_{j-1}^{CG} + \gamma_j p_{j-1} \quad (1.2.3a)$$

where γ_j is determined such that x_j^{CG} minimizes $\|x - (x_{j-1}^{CG} + \gamma p_{j-1})\|$ over all $\gamma \in R$. The new residual vector $r_j^{CG} = b - A x_j^{CG}$ can then be computed from

$$r_j^{CG} = r_{j-1}^{CG} - \gamma_j A p_{j-1} \quad (1.2.3b)$$

Finally the new search direction is chosen to be the component of r_j^{CG} which is conjugate to p_{j-1} , i.e.

$$p_j = r_j^{CG} + \delta_j p_{j-1} \quad (1.2.3c)$$

where δ_j is determined so that $p_j^T A p_{j-1} = 0$. δ_j and γ_j are computed as

$$\gamma_j = (r_{j-1}^{CG})^T r_j^{CG} / p_{j-1}^T A p_{j-1} \quad (1.2.3d)$$

$$\delta_j = (r_j^{CG})^T r_j^{CG} / (r_{j-1}^{CG})^T r_j^{CG} \quad (1.2.3e)$$

The algorithm is started by setting $x_0^{CG} = 0$, $r_0^{CG} = b$, $p_0 = b$. Formulas (1.2.3) constitute the conjugate gradient algorithm. Because of (1.2.3d) the algorithm is used only for positive definite A . There are several equivalent formulations of CG, which differ only in the way the quantities are computed (c.f. Reid [33]).

Some properties of the method will be stated here without proof. The residual vectors r_j^{CG} are orthogonal, i.e. $(r_j^{CG})^T r_k^{CG} = 0$ for $1 \leq k < j$, and the search directions p_j are conjugate, i.e. $p_j^T A p_k = 0$ for $1 \leq k < j$. Furthermore $\text{span}\{p_0, p_1, \dots, p_{j-1}\} = \text{span}\{r_0^{CG}, r_1^{CG}, \dots, r_{j-1}^{CG}\} = K^j(b; A)$. Finally from (1.2.3a) together with the conjugacy of the p_j it follows that x_j^{CG} minimizes $\|x - s\|_A$ over all $s \in K^j$. Hence $x_j^{CG} = x_j$ and the superscript can be dropped from now on. This implies that also $r_j^{CG} = r_j$, since both vectors are defined as the corresponding residuals.

The relationship between LAN and CG becomes clearer, if CG is written in matrix form. Let

$$Q_j = \left(\frac{r_0}{\|r_0\|}, \frac{r_1}{\|r_1\|}, \dots, \frac{r_{j-1}}{\|r_{j-1}\|} \right)$$

$$P_j = \left(\frac{p_0}{\|p_0\|}, \frac{p_1}{\|p_1\|}, \dots, \frac{p_{j-1}}{\|p_{j-1}\|} \right)$$

$$L_j = \begin{bmatrix} 1 & 0 & \dots & 0 \\ -\sqrt{\delta_1} & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & -\sqrt{\delta_{j-1}} & 1 \end{bmatrix}$$

$$D_j = \text{diag}(\gamma_1^{-1}, \gamma_2^{-1}, \dots, \gamma_j^{-1}) \quad (1.2.4)$$

Q_j here is identical with the matrix Q_j from the Lanczos algorithm. Then the j equations (1.2.3b) become in matrix form

$$AP_j D_j^{-1} = Q_j L_j - \frac{r_j}{\|r_{j-1}\|} e_j^T \quad (1.2.5)$$

and using $p_0 = r_0$ equation (1.2.3c) can be written

$$P_j L_j^T = Q_j \quad (1.2.6)$$

P_j can be eliminated from (1.2.5) and (1.2.6) and one obtains

$$AQ_j - Q_j(L_j D_j L_j^T) = \frac{-r_j e_j^T}{\|r_{j-1}\|} \quad (1.2.7)$$

Formula (1.2.7) is obviously the CG counterpart of formula (1.1.8a) for the Lanczos algorithm, since r_j is only a scalar multiple of q_{j+1} . The comparison with (1.1.8a) shows that CG implicitly computes the factorization

$$T_j = L_j D_j L_j^T \quad (1.2.8)$$

and (1.2.8) written out element by element reveals the relationship between the α_j 's and β_j 's of LAN and the corresponding γ_j 's and δ_j 's of CG. This

relation between the two methods has been pointed out by Householder in [13, pg. 139-141].

(1.2.6) shows from a different perspective why CG in general cannot be used for indefinite A . In this case T_j may be singular for certain values of j , then the factorization (1.2.8) does not exist, and hence the algorithm breaks down. Paige and Saunders [26] recognized that this difficulty can be overcome, if a different factorization of T_j is chosen. They suggest that instead of (1.2.8) the orthogonal factorization

$$T_j = \bar{L}_j^T Z_j \quad (1.2.9)$$

is used, where $Z_j^T Z_j = I_j$ and $\bar{L}_j$ is a lower trapezoidal matrix. This factorization always exists and is numerically stable. Paige and Saunders use a sequence of plane rotations in order to obtain the factorization (1.2.9), and derive a new iterative method for updating z_j in the same manner as CG. This new method is called SYMMLQ. The approximate solution computed in this way is identical to the one computed by either LAN or CG. There is however a subtle modification in SYMMLQ which should be mentioned. At the j -th step the matrix $\bar{L}_j$ in (1.2.9) is sometimes replaced by the matrix $\bar{L}_j$, which is the $j \times j$ leading part of $\bar{L}_{j+1}$, and which differs from $\bar{L}_j$ in the (j, j) element only. This change produces an approximate solution z_j^0 , which is different from $z_j^0 = z_j$. The algorithm chooses at each step among z_j^0 and z_j^0 the one with the smaller residual norm. z_j^0 is usually chosen when T_j is ill-conditioned. In [26] it is reported that SYMMLQ worked well on indefinite problems, but was slower than CG on positive definite problems due to a larger number of operations per step.

There are several other methods, which attempt to modify CG in order to make it also applicable for indefinite matrices, e.g. Luenberger's

algorithm [19] using hyperbolic pairs or Fletcher's [8] method, taking two CG steps at once if necessary, which is equivalent to carrying out the Bunch-Parlett [3] algorithm using 2×2 pivots on the tridiagonal matrix T_j .

Other Krylov subspace methods for solving indefinite systems are derived by using different subspaces and/or a different characterization of the approximate solution vector at the j -th step. Two important classes of algorithms are:

- 1) z_j minimizes $\|b - Az\|$ over all $z \in K^j(b, A)$.

These methods were discussed by Rutishauser [35], Reid [33] (versions 4.5.6, and 8 of the CG algorithm), and Paige and Saunders [26] (MINRES).

- 2) z_j minimizes $\|z - s\|$ over all $s \in K^j(Ab, A)$.

The feasibility of methods of this type was first recognized by Fridman [7], his algorithm is however unstable. Fletcher [8] (orthogonal direction algorithm), and Stoer and Freund [36] (SF-method) present stable versions of Fridman's method. Fletcher also shows that the vectors z_j^q , which are occasionally used in SYMMLQ, are identical to the iterates in his method.

All these methods do have their relative merits in terms of savings in storage, number of operations, or in terms of applicability to certain types of problems. However they are all iterative in nature and, since they all at least implicitly involve the Lanczos algorithm, they will all suffer from the same type of errors in finite precision arithmetic.

It is important to realize that the proposed algorithm LAN, although it is equivalent on an abstract level to the various version of CG differs radically in its practical implementation from all the methods mentioned. This radical

difference lies in the storage of the Lanczos vectors $q_1, q_2, \dots, q_j$. By keeping them and maintaining orthogonality among them, LAN actually computes something like an approximate factorization of the matrix A . LAN therefore can be looked upon as an intermediate method in between direct and iterative methods. It is iterative in the sense that implicitly at each step an approximate solution vector is updated and improved, and it is direct in the sense that as a byproduct the factorization of a low rank approximation to A is computed. The increased cost of LAN as compared to CG therefore is justified if the Lanczos vectors can be used for subsequent right hand sides in a similar fashion as the once computed triangular factors in Gaussian elimination are used. This question is pursued further in section 1.4 and in the numerical tests. CG and SYMMLQ were introduced here in some more detail, because they will be used in Chapter 4 for numerical comparisons.

1.3. Convergence Properties.

From section 1.1 it is clear that the Lanczos algorithm replaces a complicated problem by a simpler one, but it is not evident that it is also smaller, i.e. that the residual norm becomes small already for a $j \ll n$. In the general case we only know from the minimizing property of z_j that for positive definite A the error $\|x - z_j\|_A$ is monotonically decreasing with increasing j . This does not imply that the error $\|x - z_j\|$ or the residual norm $\|b - Az_j\|$ are monotonically decreasing. In fact it is typical for the algorithm that $\|r_j\|$ oscillates as it decreases. The behavior of $\|x - z_j\|$, $\|x - z_j\|_A$, and $\|b - Az_j\|$ for a sample run of LAN is shown in Figure 1.1.

For positive definite A it is possible to derive an a priori estimate on the number of steps required to reduce the error in the energy norm by a certain given factor. Since $z_j \in K'(b; A)$, z_j can be also expressed as $z_j = \omega(A)b$, where $\omega(\zeta)$ is a polynomial of degree $j-1$. Denote by P^j the set of polynomials of degree $\leq j$ and by $P_0^j = \{\omega | \omega \in P^j, \omega(0) = 1\}$. Then using the minimizing property of z_j one obtains

$$\begin{aligned} \|x - z_j\|_A &= \min_{\omega \in P^{j-1}} \|x - \pi(A)b\|_A = \min_{\omega \in P^{j-1}} \|A^{-1/2}b - \pi(A)b\|_A \\ &= \min_{\omega \in P^{j-1}} \|(I - \pi(A)A^{-1/2})b\| \leq \min_{\omega \in P_0^j} \|\omega(A)\| \|x\|_A \end{aligned} \quad (1.3.1)$$

Now let the spectrum of A be contained in the interval $[\nu, \mu]$ with $0 < \nu \leq \mu$. Then

$$\frac{\|x - z_j\|_A}{\|x\|_A} \leq \min_{\omega \in P_0^j} \max_{\lambda \in [\nu, \mu]} |\omega(\lambda)| \quad (1.3.2)$$

This min-max problem is solved by the Chebyshev polynomial T_j of degree j adapted to the interval $[\nu, \mu]$ and normalized so that $\omega(0) = 1$ (c.f. Rivlin [34]).

$$\frac{\|x - z_j\|_A}{\|x\|_A} \leq \frac{1}{|T_j(1 + 2\frac{\mu - \nu}{\mu + \nu})|} = \frac{1}{|T_j(\frac{\mu + \nu}{\mu - \nu})|} \quad (1.3.3)$$

Using $T_j(\zeta) = \frac{1}{2}[(\zeta + \sqrt{\zeta^2 - 1})^j + (\zeta - \sqrt{\zeta^2 - 1})^j]$, we obtain after some computations

$$\frac{1}{|T_j(\frac{\mu + \nu}{\mu - \nu})|} = \frac{2(\frac{\sqrt{\mu - \nu}}{\sqrt{\mu + \nu}})^j}{1 + (\frac{\sqrt{\mu - \nu}}{\sqrt{\mu + \nu}})^{2j}} \leq 2(\frac{\sqrt{\mu - \nu}}{\sqrt{\mu + \nu}})^j \quad (1.3.4)$$

where $\kappa = \frac{\mu}{\nu}$ is the spectral condition number of A . In order to reduce the initial error by a factor of δ , j has to be chosen such that

$$2(\frac{\sqrt{\kappa - 1}}{\sqrt{\kappa + 1}})^j \leq \delta \quad (1.3.5)$$

i.e.

$$j \geq \frac{\ln 2 - \ln \delta}{\ln(\frac{\sqrt{\kappa + 1}}{\sqrt{\kappa - 1}})} \quad (1.3.6)$$

Since $\ln(\frac{\sqrt{\kappa + 1}}{\sqrt{\kappa - 1}}) \geq \frac{2}{\sqrt{\kappa}}$, we finally have that

$$j \geq \frac{\sqrt{\kappa}}{2} \ln \frac{2}{\delta} \quad (1.3.7)$$

The number of steps needed in order to reduce the error in the A -norm by a given factor is according to (1.3.7) proportional to the square root of the condition number of the matrix A .

The bound derived here is sometimes a very crude estimate, as the example in Figure 1.1 shows. But in this generality it is the best possible. A priori bounds of this type can be refined in two ways, by using more information about the polynomial π (c.f. Greenbaum [10]) or by using more information about the spectrum of A (cf. Axelsson [2]). Allstam [1] reports an

extension for the indefinite case. The resulting bounds are however no improvement over bounds directly obtainable by considering $A'Ax = A'b$.

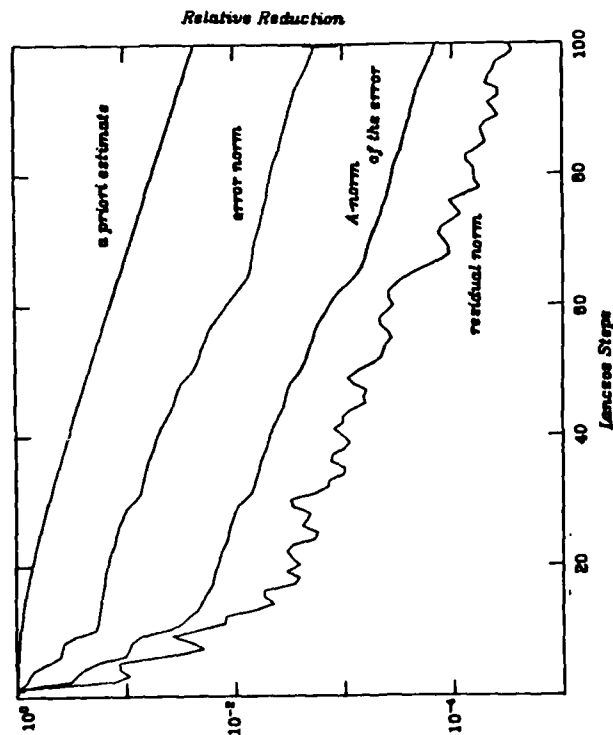


Figure 1.1. Behavior of the Error in Various Norms for a Sample Run of LAN.

1.4. Updating the Residual Norm and Solving the Triangular System.

We mentioned already in section 1.2 that $T_j f_j = \beta_j e_j$ has to be solved in a stable way, since for some $k \leq j$ the matrix T_k may be very ill-conditioned, particularly when A has both positive and negative eigenvalues. In this case the LDL' factorization of T_j may break down or give unreliable results. Since the updating of the residual norm according to (1.1.12) also requires the factorization of T_j , both tasks can be treated simultaneously. Here we want to use the QR -factorization in order to achieve this task.

The QR -factorization can be accomplished in an efficient way by using fast scaled rotations (Hammerling [11], Parlett [29, pg. 100-104]). Since we are not interested in a similarity transformation of T_j , but only in a reduction of T_j to upper triangular form, we simply premultiply T_j and simultaneously the right hand side $\beta_j e_j$, by matrices of the type

$$\begin{bmatrix} 1 & & & \\ & 1 & -\sigma & \\ & \sigma & 1 & \\ & & & 1 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} 1 & & & \\ & \tau & -1 & \\ & 1 & \tau & \\ & & & 1 \end{bmatrix} \quad (1.4.1)$$

Let us call the matrix on the left of type I and the matrix on the right of type II. The scalars σ and τ in (1.4.1) are chosen to reduce the subdiagonal element β_k to zero. The matrices in (1.4.1) represent Givens plane rotations, scaled by a factor of $\cos \phi$ (type I) or $\sin \phi$ (type II), where ϕ is the angle of rotation. Therefore $\sigma = \tan \phi$ and $\tau = \cot \phi$, and we can choose between both types such that $|\sigma| \leq 1$ or $|\tau| < 1$. This enhances the stability of the reduction.

If we consider in detail the i -th step, we obtain for type I:

$$\begin{bmatrix} 1 & -a_{i+1} & 0 \\ a_{i+1} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \bar{a}_i & \bar{\beta}_{i+1} & 0 \\ \bar{a}_{i+1} & a_{i+1} & \bar{\beta}_{i+2} \\ 0 & \bar{\beta}_{i+2} & a_{i+2} \end{bmatrix} = \begin{bmatrix} \bar{a}_i - a_{i+1}\bar{\beta}_{i+1} & \bar{\beta}_{i+1} - a_{i+1}a_{i+2} & -a_{i+1}\bar{\beta}_{i+2} \\ \bar{a}_{i+1} + \bar{\beta}_{i+1}a_{i+2} & a_{i+1} + \bar{\beta}_{i+1}a_{i+2} & \bar{\beta}_{i+2} \\ 0 & \bar{\beta}_{i+2} & a_{i+2} \end{bmatrix}.$$

and for type II

$$\begin{bmatrix} \tau_{i+1} & -1 & 0 \\ 1 & \tau_{i+1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \bar{a}_i & \bar{\beta}_{i+1} & 0 \\ \bar{a}_{i+1} & a_{i+1} & \bar{\beta}_{i+2} \\ 0 & \bar{\beta}_{i+2} & a_{i+2} \end{bmatrix} = \begin{bmatrix} \tau_{i+1}\bar{a}_i - \bar{\beta}_{i+1} & \tau_{i+1}\bar{\beta}_{i+1} - a_{i+1} & -\bar{\beta}_{i+2} \\ \bar{a}_i + \tau_{i+1}\bar{\beta}_{i+1} & \bar{\beta}_{i+1} + \tau_{i+1}a_{i+1} & \tau_{i+1}\bar{\beta}_{i+2} \\ 0 & \bar{\beta}_{i+2} & a_{i+2} \end{bmatrix}.$$

In order to reduce the subdiagonal element to zero we have to set for type I:

$$a_{i+1}\bar{a}_i + \bar{\beta}_{i+1} = 0 \quad \text{i.e.,} \quad a_{i+1} = -\frac{\bar{\beta}_{i+1}}{\bar{a}_i}. \quad (1.4.3a)$$

and for type II:

$$\bar{a}_i + \tau_{i+1}\bar{\beta}_{i+1} = 0 \quad \text{i.e.,} \quad \tau_{i+1} = -\frac{\bar{a}_i}{\bar{\beta}_{i+1}}. \quad (1.4.3b)$$

Hence we choose type I if $\bar{\beta}_{i+1} \leq |\bar{a}_i|$ and type II if $|\bar{a}_i| < \bar{\beta}_{i+1}$. Note that in both case we change the diagonal element in position $(i+1, i+1)$, and for type II the super diagonal element in position $(i+1, i+2)$. The - indicates these elements changed from the previous step. This algorithm cannot break down because $\bar{\beta}_{i+1} > 0$. If $\bar{a}_i = 0$ we have a scaled rotation of type II, which interchanges the rows. Finally the right hand side β_{i+1} is modified as follows:

$$\begin{bmatrix} 1 & -a_{i+1} \\ a_{i+1} & 1 \end{bmatrix} \begin{bmatrix} \eta_i \\ 0 \end{bmatrix} = \begin{bmatrix} \eta_i \\ a_{i+1}\eta_i \end{bmatrix} = \beta_{i+1} \begin{bmatrix} \eta_i \\ \eta_{i+1} \end{bmatrix}. \quad (1.4.4a)$$

$$\begin{bmatrix} \tau_{i+1} & -1 \\ 1 & \tau_{i+1} \end{bmatrix} \begin{bmatrix} \eta_i \\ 0 \end{bmatrix} = \begin{bmatrix} \tau_{i+1}\eta_i \\ \eta_i \end{bmatrix} = \beta_{i+1} \begin{bmatrix} \eta_i \\ \eta_{i+1} \end{bmatrix}. \quad (1.4.4b)$$

From this material we can construct algorithms for both the updating of the residual norm, and the solution of the symmetric tridiagonal system. In the practical algorithm we will store only τ_i or $1/a_i$ in an array named τ .

Therefore $|\tau_i| < 1$ indicates that a scaled rotation of type II was performed at step i , and $|\tau_i| \geq 1$ indicates a scaled rotation of type I with $a_i = 1/\tau_i$.

In order to update the residual norm we just update the quantities $\bar{a}_i$, $\bar{\beta}_{i+1}$ and η_i as we go along with the Lanczos algorithm. Then the residual norm according to (1.1.12) can be found as $\bar{\beta}_{i+1}|\eta_i/\bar{a}_i|$. The nice feature is that there is no need to keep the $\bar{a}_i$, $\bar{\beta}_i$ and η_i . These quantities can be recomputed from the τ_i , when needed for the solution $f_j = (\varphi_1, \varphi_2, \dots, \varphi_j)^T$, e.g. the $\bar{a}_i$ can be found from the relation $\bar{a}_i + \tau_i\bar{\beta}_{i+1} = 0$, or $\bar{a}_i/\tau_{i+1} + \bar{\beta}_i = 0$. Similarly we can find $\bar{\beta}_{i+1}$ and η_i . The algorithm for the updating procedure is given in Table 1.3, where only four variables $\bar{a}$, $\bar{\beta}$, η , and η_{old} are used. The corresponding algorithm for the back substitution is given in Table 1.4.

```

1. Initialization
    $\bar{a} \leftarrow a_1, \bar{\beta} \leftarrow \beta_2, \eta_{old} \leftarrow \beta_1$ 
2. Loop: for  $i = 2, 3, \dots$  do
    $\tau_i \leftarrow -\bar{a}/\bar{\beta}_i$ 
    $\eta \leftarrow \eta_{old}/\tau_i$ 
    $\bar{a} \leftarrow \bar{a} + \bar{\beta}_i/\tau_i$ 
    $\bar{\beta} \leftarrow \bar{\beta}_i + \bar{\beta}_i/\tau_i$ 
   if  $(|\tau_i| < 1)$  then
      $\bar{a} \leftarrow \bar{a}\tau_i$ 
      $\bar{\beta} \leftarrow \bar{\beta}\tau_i$ 
      $\eta \leftarrow \eta_{old}\tau_i$ 
   residual norm  $\leftarrow \bar{\beta}_{i+1}|\eta/\bar{a}|$ 

```

Table 1.3. Algorithm for Updating the Residual Norm.

1. Initialization	
$\eta_j \leftarrow \eta_j / 2$	
$\beta \leftarrow \begin{cases} \beta_j & \text{if } \tau_{j-1} \geq 1 \\ \tau_{j-1} \beta_j & \text{if } \tau_{j-1} < 1 \end{cases}$	
$\eta \leftarrow \tau_j \eta$	
$\eta_{j-1} \leftarrow \begin{cases} \eta - (\beta - \alpha_j / \tau_j) \eta_j / [-\beta_j (\tau_j + 1 / \tau_j)] & \text{if } \tau_j \geq 1 \\ \eta - ((\tau_j \beta - \alpha_j) \eta_j) / [-\beta_j (1 + \tau_j^2)] & \text{if } \tau_j < 1 \end{cases}$	
2. For $k = 3, \dots, j$ do	
$i \leftarrow j + 1 - k$	
$\beta \leftarrow \begin{cases} \beta_{i+1} & \text{if } \tau_i \geq 1 \\ \tau_i \beta_{i+1} & \text{if } \tau_i < 1 \end{cases}$	
$\eta \leftarrow \begin{cases} \eta_{i+1} & \text{if } \tau_{i+2} \geq 1 \\ \tau_{i+1} \eta_{i+1} / \tau_{i+2} & \text{if } \tau_{i+2} < 1 \end{cases}$	
$\text{temp} \leftarrow \begin{cases} \tau_i \tau_{i+1} & \text{if } \tau_{i+1} \geq 1 \\ \tau_i & \text{if } \tau_i < 1 \end{cases}$	
$\eta_i \leftarrow \frac{\text{temp} - (\beta_{i+1} \tau_{i+1} - \alpha_{i+1}) \eta_{i+1} + \beta_{i+2} \eta_{i+2}}{-\beta_{i+1} (1 + \tau_{i+1}^2)}$	

Table 1.4. Back Substitution Algorithm for Solving $T_j f_j = \beta_j e_1$.

The actual implementation can be simplified further, for example it is possible to use only two if-statements per loop. Finally it should be noted that this idea is not restricted to the special right hand side $\beta_j e_1$, i.e., it is possible to implement an algorithm for solving symmetric tridiagonal systems based on scaled rotations, which uses only one extra vector of storage (for τ), and leaves the elements of the matrix unchanged.

1.6. Initial Guess, Starting Vector, and Treatment of Several Right Hand Sides.

So far we have concerned ourselves only with the case where the system $Ax = b$ had to be solved, and no additional information was available. In many applications, however, an initial guess x_0 for the solution is available. The standard procedure is then to write $x = x_0 + x_1$, where x_1 is a correction to the initial guess, and to solve

$$Ax_1 = r_0 = b - Ax_0 \quad (1.5.1)$$

instead of the original equation.

One may ask whether there is something more sophisticated which can be done when a good guess is given, for example, one might want to start the Lanczos algorithm with x_0 , i.e. set $q_1 \leftarrow x_0 / \|x_0\|$. It is however easy to check that even if x_0 were the exact solution, the algorithm would not recognize that and would proceed until the residual norm becomes small. Another suggestion is to use a starting vector which has large components in the direction of eigenvectors corresponding to the small eigenvalues of A . But in general it is not advisable to use any other vector than the right hand side b as a starting vector for the Lanczos algorithm for the following reason. Suppose we used the vector g , where g is an arbitrary n -vector as a starting vector for the Lanczos algorithm. Then the computation of the residual norm according to (1.1.11) must be modified to

$$\begin{aligned} r_j &= b - AQ_j T_j^{-1} Q_j^T b \\ &= b - (Q_j T_j + \beta_{j+1} q_{j+1} q_j^T) T_j^{-1} Q_j^T b \\ &= b - Q_j Q_j^T b - \beta_{j+1} q_{j+1} q_j^T T_j^{-1} Q_j^T b \\ &= (I - Q_j Q_j^T) b - \beta_{j+1} q_{j+1} q_j^T \end{aligned} \quad (1.5.2)$$

where e_j denotes the j -th component of the vector A_j , solution to $T_j A_j = Q_j b$. The important difference to (1.1.11) is not that $Q_j b$ is now a full j -vector, but the additional $(I - Q_j Q_j^T)b$ -term in the residual. This term accounts for the fact that in the general case the right hand side is not necessarily contained in the Krylov subspace K^j , and $(I - Q_j Q_j^T)b$ is just the orthogonal complement of b . If g is not related to b , there is therefore no reason to hope that by some coincidence $\|(I - Q_j Q_j^T)b\|$ might be small for $j < n$. In other words, we want $b \in K^j$ for small j . Unless we take $q_1 = b/\|b\|$, the only way to guarantee this is by choosing a q_1 such that $\pi(A)q_1 = b$, where π is some polynomial of degree $j-1$. This however requires to solve a system of linear equations, which is even more complicated than the original one. In general there seems to be no better alternative to choosing the right hand side as starting vector, and it appears that given an initial guess z_0 , it is best to utilize it as in (1.5.1), and then to proceed with the Lanczos algorithm using $b - Ax_0$ as starting vector.

This discussion is of certain relevance for the treatment of a sequence of right hand sides. Let us consider the case where we have stopped the first Lanczos run for solving $Ax^{(1)} = b^{(1)}$ at step j , because the residual became negligible. The Lanczos vectors Q_j and T_j are then still available and can be used for computing an approximation to the solution $x^{(2)}$ of the problem

$$Ax^{(2)} = b^{(2)} \quad (1.5.3)$$

where $b^{(2)}$ is a new right hand side. We can find an initial approximation $z_0^{(2)}$ to $x^{(2)}$ from span (Q_j) as follows:

$$z_0^{(2)} = Q_j T_j^{-1} Q_j^T b^{(2)} \quad (1.5.4)$$

This is just (1.1.10) for the new right hand side $b^{(2)}$. We have to form $Q_j^T b^{(2)}$, which is now in general a full j -vector, then solve $T_j A_j = Q_j^T b^{(2)}$, and finally

assemble $z_0^{(2)} = Q_j A_j$. The initial guess $z_0^{(2)}$ will be utilized as outlined above, i.e. we compute a solution $x_j^{(2)} = r_j^{(2)} = b^{(2)} - Ax_j^{(2)}$ by starting a new Lanczos run. The numerical results in Chapter 4 indicate that this second run of the algorithm will need considerably fewer steps than did the first run, provided that the second right hand side represents a physically related problem. For an arbitrary right hand side it is not clear that upon (Q_j) is a good subspace for approximation, and there may be little to be gained from computing $z_0^{(2)}$. In this case it will be better to use a more sophisticated procedure for the treatment of consecutive right hand sides. In [30] it is described how the Lanczos vectors from the second run can be kept orthogonal to the already computed Lanczos vectors from the first run. This algorithm will not only reduce the number of Lanczos steps for the second right hand side, but it will also provide an orthonormal basis for a larger subspace, which can be used for the third right hand side, and so on.

However it should be clear from the initial discussion in this section that it does not pay simply to continue the old Lanczos recurrence from the first right hand side, and wait until the residual norm becomes small, because we cannot expect that $b^{(2)}$ will be well represented in $K^j(b^{(1)}, A)$.

2. ANALYSIS OF THE SYMMETRIC LANCZOS ALGORITHM IN FINITE PRECISION

2.1. A Mathematical Model of the Lanczos Algorithm in the Presence of Roundoff

Most error analyses start out by making some assumptions on the roundoff errors which will occur when elementary operations like addition, etc., are carried out in floating point computation with relative precision ϵ . Based on these assumptions upper bounds on the errors in vector inner products, matrix-vector multiplications, etc., are derived or the reader is referred to Wilkinson [40]. After providing these tools then finally the object of analysis itself is approached. Lengthy and complicated derivations finally yield error bounds which are rigorous but, in most cases, unrealistically pessimistic.

The error analyst's dilemma is that he has to take into account any possible contrived worst case example at each step in his analysis in order to make it rigorous, but he also knows that this combination will hardly ever occur in practice. By including all these cases it is not only more complicated to read and understand the analysis, but it is also difficult to prove facts which appear to be "true" from practical experience with an algorithm.

We try here to find a way out of this dilemma by using a different approach. In this section we are going to state a set of assumptions on the behavior of the Lanczos algorithm in finite precision. These assumptions constitute a model for the actual computation. A model which includes features (the essential ones in my opinion), but discards others (the irrelevant ones). On this model we build a rigorous analysis. The simplification of the results and their relation to the observed behavior of the Lanczos algorithm must eventually justify our choice of model.

The presentation of the Lanczos algorithm in section 1.1 assumed an ideal mathematical setting. However, Lanczos himself [17] was already aware of the strong influence which roundoff had on the algorithm. The computed quantities can differ greatly from their theoretical counterparts.

In the context of finite precision arithmetic the basic three term recurrence (1.1.2) between the Lanczos vectors at the j -th step can be written

$$\beta_{j+1}q_{j+1} = Aq_j - \alpha_j q_j - \beta_j q_{j-1} - f_j \quad (2.1.1)$$

where the n -vector f_j accounts for the local roundoff errors at the j -th step, and the α_j, β_j, q_j denote from now on the corresponding computed quantities. As in (1.1.6) the first j equations (2.1.1) can be written in matrix form

$$AQ_j - Q_j T_j = \beta_j q_{j+1} q_j^T + F_j \quad (2.1.2)$$

where the $n \times j$ matrix F_j is given by $F_j = (f_1, f_2, \dots, f_j)$. A bound on $\|F_j\|$ depends on the specific implementation of the Lanczos algorithm, and on the machine roundoff unit. Parlett [29, pg.288] reports that no exception has been observed to the assertion that

$$\|F_j\| \leq \epsilon \|A\| \quad (2.1.3)$$

This claim is supported by a study of $\|f_j\|$, reported in section 3.1. In the following analysis we assume that (2.1.3) holds, i.e. that the local errors are at roundoff level.

Let the $j \times j$ matrix $W_j = (w_k)$ be defined by

$$W_j = Q_j^T Q_j \quad (2.1.4)$$

Ideally the Lanczos vectors should be orthogonal, i.e. $W_j = I_j$. But this relation is completely destroyed by the effects of finite precision arithmetic. No

implementation of the Lanczos algorithm as described in Chapter 1 yields a small a priori bound on $\|W_j - I_j\|$. In fact the elements of $W_j - I_j$ can become as big as 1. The computed Lanczos vectors do not only lose orthogonality but become linearly dependent to working precision. The growth of the elements of $W_j - I_j$ will be referred to as the *loss of orthogonality* among the Lanczos vectors. Let the j first Lanczos vectors $q_1, q_2, \dots, q_j$ satisfy

$$|q_k^T q_\ell| \leq \omega_j \quad (2.1.5)$$

for $\ell = 1, \dots, j$; $k = 1, \dots, j$; $k \neq \ell$ and $0 \leq \omega_j \leq 1$. The smallest ω_j for which (2.1.5) holds will be called the *level of orthogonality* among the Lanczos vectors. If $\omega_j = \sqrt{\epsilon}$, then the Lanczos vectors will be called *semiorthogonal*. Clearly, if $\omega_j = 0$ the vectors are orthonormal. The following example illustrates the typical loss of orthogonality as the Lanczos algorithm proceeds.

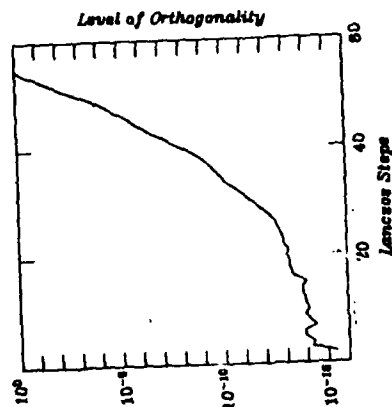


Figure 2.1. The Loss of Orthogonality among the Lanczos Vectors.

In Figure 2.1 the level of orthogonality among the Lanczos vectors is plotted

on a logarithmic scale for the first 56 steps of a run of the algorithm with a matrix of order $n=861$, resulting from an approximation to Poisson's equation on the unit square with 31×31 grid points. The starting vector is $q_1 = (1, 1, \dots, 1)^T / \sqrt{861}$.

Some more assumptions are necessary in order to simplify the technical details of the analysis of the loss of orthogonality. It will be assumed that the Lanczos vectors are exactly normalized, i.e. that

$$q_k^T q_k = 1 \quad \text{for } k = 1, \dots, j \quad (2.1.6)$$

and that locally the level of orthogonality among the q_k 's is of the size of the roundoff unit, i.e. that

$$|q_k^T q_\ell| \leq \epsilon_1 \quad \text{for } k = 1, \dots, j \quad (2.1.7)$$

Here ϵ_1 is some constant $1 \gg \epsilon_1 > \epsilon$. In practice it turns out that $q_k^T q_\ell$ occasionally may become large if β_{j+1} is small, or equivalently if the angle between Aq_j and q_j is small. This is actually not a problem peculiar to the Lanczos algorithm, but of orthogonalizing two vectors which form an small angle. It is solved by reorthogonalizing q_{j+1} immediately (within the Lanczos step) against q_j if β_{j+1} drops below some threshold. We therefore assume that ϵ_1 is a modest multiple of the roundoff unit. As long as $\epsilon_1 \ll \sqrt{\epsilon}$ the actual size of ϵ_1 is not important for the following analysis. Similarly the later analysis will show that roundoff errors in the normalization of the q_j 's are inconsequential for the loss of orthogonality.

Finally let us assume that

$$\text{no } \beta_{j+1} \text{ ever becomes negligible.} \quad (2.1.8)$$

This is almost always true in practice, and the rare cases where a β_{j+1} does become small, are actually the lucky ones, since then the algorithm should

be terminated, having found an invariant subspace.

(2.1.1)-(2.1.8) constitute the mathematical model of the Lanczos algorithm which we are going to investigate further. The goal of the remaining chapter is to identify a mechanism which causes the loss of orthogonality in the Lanczos algorithm, and then to analyze the algorithm in the light of this new understanding. The results will help to clarify the role of the $\sqrt{\epsilon}$ threshold, which appears both in Parlett and Scott's [32] and Grcar's [9] work. The insight will also lead to a new orthogonalization procedure, which will be discussed in the Chapter 3.

2.2 The Loss of Orthogonality.

The loss of orthogonality and the associated "instability" of the Lanczos algorithm in the past has been credited to "an accumulation of roundoff and cancellation errors" (Golub, Underwood, Wilkinson [8]). Paige [24] was the first to provide a better understanding of what exactly is happening when orthogonality is lost. However Paige does not convincingly identify a mechanism, which would explain the loss of orthogonality directly. Recently Grcar [9] gave a new interpretation and his work is closely related to the results of this section. He is the first one to regard the loss of orthogonality as an amplification of the local errors which can be explained through recurrence formulas. We follow his ideas and express the loss of orthogonality in terms of computed quantities. Thus we obtain a simpler and easier understandable recurrence formula. This is done here in form of Theorem 2.1. Paige's main result concerning the loss of orthogonality then immediately follows from Theorem 2.1.

The situation becomes clearer, if one follows a simple geometrical argument. Suppose the algorithm was carried out for j steps without any error and the vectors $q_1, \dots, q_j$ were perfectly orthogonal. Now at the $j+1$ step a small error occurs, such that q_{j+1} is no longer orthogonal to the previous Lanczos vectors. From then on the algorithm is again continued without error. Even if q_{j+2} were constructed perfectly orthogonal to q_{j+1} and q_j , it would no longer be orthogonal to the vectors $q_1, \dots, q_{j-1}$, because q_{j+1} was not orthogonal to them. The same is true for all consecutive Lanczos vectors. The once introduced error is propagated to future Lanczos vectors.

Now if two consecutive Lanczos vectors q_{k-1} and q_k deviate slightly from their correct direction then of course the vector Aq_k will be also slightly

wrong. This by itself would not be so bad, but this already slightly wrong A_k will now additionally be orthogonalized against already deviating vectors and thus the resulting q_{k+1} will differ even more from its true direction. Once introduced, the error is thus not only propagated, but depending on the geometry of the q_i 's it may be additionally amplified.

The loss of orthogonality therefore can be viewed as the result of an amplification of each local error after its introduction into the computation. The following theorem is the arithmetic equivalent of the geometric considerations above. It quantifies precisely how the local error is propagated in the algorithm, and how the level of orthogonality rises due to the mechanisms of the algorithm. It is the core of our analysis.

Theorem 2.1. The elements ω_{kj} of the $j \times j$ matrix $W_j = Q_j^T Q_j$ satisfy the following recurrence:

$$\begin{aligned} \omega_{kk} &= 1 & \text{for } k = 1, \dots, j \\ \omega_{k,k-1} &= \varepsilon_k & \text{for } k = 2, \dots, j \end{aligned} \quad (2.2.1)$$

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

for $1 \leq k < j$, and $\omega_{j,j+1} = \omega_{k+1,j}$. Here $\omega_{k0} = 0$ and $\varepsilon_k = q_k^T q_{k-1}$.

Proof. Write (2.1.1) for j and for k :

$$\beta_{j+1} q_{j+1} = A_j - \alpha_j q_j - \beta_j q_{j-1} - f_j \quad (2.2.2)$$

$$\beta_{k+1} q_{k+1} = A_k - \alpha_k q_k - \beta_k q_{k-1} - f_k \quad (2.2.3)$$

Forming $q_k^T(2.2.2) - q_j^T(2.2.3)$ and simplifying yields the result.

Theorem 2.1 was already published by Takekoshi and Natori [39], but rediscovered here independently.

Note that (2.2.1) can be also obtained in vector form. First premultiply (2.1.1) by Q_j :

$$\begin{aligned} \beta_{j+1} Q_j^T q_{j+1} &= Q_j^T A_j - \alpha_j Q_j^T q_j - \beta_j Q_j^T q_{j-1} - Q_j^T f_j \\ &= T_j Q_j^T q_j - \alpha_j Q_j^T q_j - \beta_j Q_j^T q_{j-1} + \beta_{j+1} q_j^T q_j + f_j^T q_j - Q_j^T f_j \end{aligned} \quad (2.2.4)$$

Let R_j be the strictly upper triangular part of W_j , i.e. R_j is zero on and below the diagonal, and let $w_1, w_2, \dots, w_j$ be the columns of R_j . And let $w_{j+1} = Q_j^T q_{j+1}$. Then from (2.2.4) it follows that

$$\beta_{j+1} w_{j+1} = T_j w_j - \alpha_j w_j - \beta_j w_{j-1} + g_j \quad (2.2.5)$$

where $g_j = f_j^T q_j - Q_j^T f_j$. Equation (2.2.5) could have been obtained directly from (2.2.1), by writing (2.2.1) in vector form for $k = 1, \dots, j$. From (2.2.5) we can obtain an estimate for the loss of orthogonality:

$$\beta_{j+1} \|w_{j+1}\| \leq (\|T_j\| + |\alpha_j|) \|w_j\| + \beta_j \|w_{j-1}\| + O(\varepsilon \|A\|) \quad (2.2.6)$$

$$\leq 2\|A\| \max\{\|w_j\|, \|w_{j-1}\|\} + O(\varepsilon \|A\|) \quad (2.2.7)$$

Therefore the level of orthogonality grows at most by a factor of $2\|A\|/\beta_{j+1}$ at each step. A small β_{j+1} will cause a great loss of orthogonality. A Lanczos run, which has rapidly decreasing or greatly varying β_k 's will therefore suffer from a larger loss of orthogonality, than a run with nearly constant β_k 's.

The recurrence formula (2.2.1) shows that the loss of orthogonality is merely initiated by the local error f_j . The growth of the elements of W_j depends mainly on the α_k 's and β_k 's. It is therefore definitely not due to an accumulation of roundoff or cancellation errors. Once the ω_{jk} have grown to a certain level of about $O(\varepsilon^4)$, the local error terms $q_j^T f_k - q_k^T f_j$, which are $O(\varepsilon)$ contribute negligibly to the growth of the loss of orthogonality.

Paige [25] puts considerable effort into analyzing computational variants of the Lanczos algorithm in order to determine among several possible

variations of the algorithm the one for which the local error f_j is smallest. In the light of (2.2.1) this is irrelevant as regards the loss of orthogonality. As long as the local error remains at the roundoff level, all computational variants will suffer from the same loss of orthogonality. The loss of orthogonality is a phenomenon which is started by the f_j , but from then on its growth is determined by the σ_j 's and β_j 's, i.e. by the eigenvalue distribution of A and by the starting vector g .

The way in which orthogonality is lost can be understood better if equation (2.2.5) is analyzed further. Let the exact spectral factorization of T_j be given by $T_j S_j = S_j \Theta_j$, where $\Theta_j = \text{diag}(\theta_j^{(1)}, \dots, \theta_j^{(j)})$, $S_j = (s_j^{(1)}, \dots, s_j^{(j)})$, and $S_j^* = S_j^{-1}$, and define the vectors $y_i = Q_j s_i$ for $i = 1, \dots, j$. Note that contrary to (1.1.13) we consider here the exact eigen decomposition of the computed T_j . Therefore the $\theta_j^{(i)}$ and $y_i^{(i)}$ should be referred to as the computed Ritz values and vectors. They may differ from their ideal counterparts as defined in (1.1.13). Especially there is no reason to expect the computed Ritz vectors to be orthonormal. Nevertheless we will refer to them here simply as Ritz values and vectors, since no confusion with the ideal quantities is likely. Furthermore let $\sigma_k = \sigma_k^{(j)}$, the bottom element of the eigenvector $s_k^{(j)}$, and let the eigenvectors $s_k^{(j)}$ be normalized to make σ_k positive.

With all this notation the remaining analysis becomes quite simple. Considering the first j steps of the algorithm, the corresponding instances of formula (2.2.5) can be combined in matrix form as

$$\beta_{j+1} w_{j+1} \sigma_j^* = T_j R_j - R_j T_j + G_j \quad (2.2.6)$$

where G_j is the strictly upper triangular part of $F_j^* Q_j - Q_j^* F_j$. Forming $s_i^*(2.2.6)_k$, one obtains

$$\beta_{j+1} w_{j+1} \sigma_j^* = \sigma_j s_i^* R_j s_i - s_i^* R_j s_i \sigma_j + s_i^* G_j s_i \quad (2.2.9)$$

This is precisely Paige's theorem:

Theorem 2.4. (Paige). Let S_j , Θ_j , G_j , σ_j , and γ_k be defined as above. Then the vectors $y_k = Q_j s_k$, for $k = 1, \dots, j$ satisfy

$$y_k^{(j+1)} = \frac{\gamma_k}{\beta_{j+1} \sigma_j} \quad (2.2.10)$$

Formula (2.2.10) describes the way in which the orthogonality is lost. We have assumed in (2.1.8) that no β_{j+1} becomes negligible. If we also assume that γ_k is tiny like $\varepsilon \|A\|$, the only way that $y_k^{(j+1)}$ can become large is by σ_k becoming small. As Paige pointed out

$$\|A y_k - y_k \sigma_k\| = \|A Q_j s_k - Q_j s_k \sigma_k\| = \|\beta_{j+1} \Theta_{j+1} s_k^* s_k + F_j s_k\| \leq \beta_{j+1} \sigma_k + \varepsilon \|A\|$$

and so a small σ_k indicates that (ϕ_k, y_k) is an approximate eigenpair of the matrix A . Paige's theorem therefore can be stated as: loss of orthogonality implies convergence of a Ritz pair to an eigenpair.

2.3. Lemmas.

In this section we will state and prove several Lemmas, which will be needed in the later analysis of the Lanczos algorithm. These Lemmas are mainly concerned with certain properties of the matrix $W_j = Q_j^* Q_j$ and related matrices, and are therefore completely independent of any properties of the Lanczos algorithm.

Let the $j \times j$ matrix W be given by $W = (w_{kl})$, with $w_{kl} = 1$ for $k=l, \dots, j$, and $-1 \leq w_{kl} \leq 1$ for $k \neq l$, $k, l = 1, \dots, j$. Then define $\omega = \max_{1 \leq k, l \leq j} |w_{kl}|$ and

$$W_+ = (1-\omega)I + \omega \omega^*, \quad W_- = (1+\omega)I - \omega \omega^*, \quad \text{where the } j\text{-vector } \omega^* = (1, 1, \dots, 1).$$

Denote by $\lambda_1(W)$ the smallest and by $\lambda_j(W)$ the largest eigenvalue of the matrix W .

Lemma 1.

$$a) \lambda_1(W) \geq 1 - (j-1)\omega.$$

$$b) \lambda_j(W) \leq 1 + (j-1)\omega.$$

$$c) \|W\| \leq 1 + (j-1)\omega.$$

$$d) \text{ If } \omega < \frac{1}{j-1} \text{ then } W^{-1} \text{ exists and } \|W^{-1}\| \leq \frac{1}{1-(j-1)\omega}.$$

Proof. Application of Gershgorin's theorem.

Lemma 2. Let $\omega \leq \frac{1}{2(j-2)}$. Then $LL^* = W$ the Choleski factorization of W exists and

$$\|L\| = \|L^*\| \leq 2.$$

$$\|L^{-1}\| = \|L^{*-1}\| \leq 2.$$

Proof

$$\|L\| = \sqrt{\lambda_j(LL^*)} = \sqrt{\lambda_j(W)} \leq (1 + (j-1)\omega)^{\frac{1}{2}} \leq 2$$

$$\|L^{-1}\| = \sqrt{\lambda_1(L_j^{-1}L_j^{-1})} = \sqrt{(\lambda_1(W))^{-1}} \leq (1 - (j-1)\omega)^{-\frac{1}{2}} \leq 2.$$

Lemma 3. The Choleski factor L_* of the $j \times j$ matrix W_*

$$W_* = \begin{bmatrix} 1 & \omega & \dots & \omega \\ \omega & 1 & \dots & \omega \\ \vdots & \vdots & \ddots & \vdots \\ \omega & \omega & \dots & 1 \end{bmatrix} \quad (2.3.1)$$

where $0 \leq \omega < 1$, is given by

$$L_* = \begin{bmatrix} \delta_1 & 0 & \dots & 0 \\ \eta_1 & \delta_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \eta_1 & \eta_2 & \dots & \delta_{j-1} & 0 \\ \eta_1 & \eta_2 & \dots & \eta_{j-1} & \delta_j \end{bmatrix} \quad (2.3.2)$$

where

$$\delta_k = \sqrt{\frac{1-\omega(1+(k-2)\omega)}{1+(k-2)\omega}}, \quad k = 1, \dots, j \quad (2.3.3)$$

$$\eta_k = \omega \sqrt{\frac{1-\omega}{(1+(k-2)\omega)(1+(k-1)\omega)}}, \quad k = 1, \dots, j-1. \quad (2.3.4)$$

Proof. With $0 \leq \omega < 1$ the matrix W is symmetric and positive definite, and its Choleski factorization $W_* = L_*L_*^*$ exists.

Consider the first step of the factorization:

$$W_* = W_1 = \begin{bmatrix} \delta & \omega^* \\ \omega & W_1 \end{bmatrix} =$$

$$= \begin{bmatrix} \sqrt{\delta} & 0^* \\ \frac{\omega}{\sqrt{\delta}} & I_{j-1} \end{bmatrix} \begin{bmatrix} 1 & 0^* \\ 0 & W_1 - \frac{\omega\omega^*}{\delta} \end{bmatrix} \begin{bmatrix} \sqrt{\delta} & \omega^* \sqrt{\delta} \\ 0 & I_{j-1} \end{bmatrix}$$

Therefore $\delta_1 = \sqrt{\delta} = \sqrt{j}$ and $\eta_1 = \omega/\sqrt{\delta} = \omega$ and

$$W_2 = W_1 - \frac{\omega\omega^*}{\delta} = \begin{bmatrix} 1-\omega^2 & \omega-\omega^2 & \dots & \omega-\omega^2 \\ \omega-\omega^2 & 1-\omega^2 & \dots & \omega-\omega^2 \\ \vdots & \vdots & \ddots & \vdots \\ \omega-\omega^2 & \omega-\omega^2 & \dots & 1-\omega^2 \end{bmatrix}$$

Denote now by $\tilde{\delta}_k$ and $\tilde{\eta}_k$ the diagonal and off diagonal elements in the matrix W_* , which contains the part of W which is still to be factored at the k -th step.

Then considering the k -th step in the same way as the first step, one finds that $\tilde{\delta}_k$ and $\tilde{\eta}_k$ can be computed by the recurrence

$$\tilde{\delta}_k = \tilde{\delta}_{k-1} - \frac{\tilde{\eta}_{k-1}^2}{\tilde{\delta}_{k-1}}$$

$$\tilde{\eta}_k = \tilde{\eta}_{k-1} - \frac{\tilde{\eta}_{k-1}^2}{\tilde{\delta}_{k-1}}$$

where $\tilde{\delta}_1 = 1$ and $\tilde{\eta}_1 = \omega$. It can be shown by induction that

$$\tilde{\delta}_k = \frac{(1-\omega)(1+(k-1)\omega)}{1+(k-2)\omega}, \quad k = 1, \dots, j$$

$$\tilde{\eta}_k = \frac{(1-\omega)\omega}{1+(k-2)\omega}, \quad k = 1, \dots, j-1.$$

Finally δ_k and η_k are obtained from

$$\delta_k = \sqrt{\tilde{\delta}_k}$$

$$\eta_k = \frac{\tilde{\eta}_k}{\sqrt{\tilde{\delta}_k}}.$$

Lemma 4. If $\omega < \frac{1}{j-1}$ the Choleski factor L_+ of the matrix W_+

$$W_+ = \begin{bmatrix} 1 & -\omega & \dots & -\omega \\ -\omega & 1 & \dots & \dots \\ \vdots & \vdots & \ddots & \vdots \\ -\omega & \dots & \dots & 1 \end{bmatrix}$$

where $0 \leq \omega < 1$, is given by

$$L_- = \begin{bmatrix} \delta_1^{-1} & 0 & \dots & 0 \\ \eta_1^{-1} & \delta_2^{-1} & \dots & \dots \\ \vdots & \vdots & \ddots & \vdots \\ \eta_{j-1}^{-1} & \dots & \dots & \delta_{j-1}^{-1} \\ \eta_1^{-1} & \eta_2^{-1} & \dots & \eta_{j-1}^{-1} & \delta_j^{-1} \end{bmatrix}.$$

where

$$\delta_k^{-1} = \sqrt{\frac{(1+\omega)(1+(k-1)\omega)}{1-(k-2)\omega}}, \quad k = 1, \dots, j$$

$$\eta_k^{-1} = \omega \sqrt{\frac{1+\omega}{(1-(k-2)\omega)(1-(k-1)\omega)}}, \quad k = 1, \dots, j-1.$$

Proof. Replace ω by $-\omega$ in Lemma 3.

Lemma 5. Let W_- and L_- be as in Lemma 3 and W_+ and L_+ as in Lemma 4. If $(j-1)\omega^2/2 \leq \varepsilon$, then $\delta_j = 1 + O(\varepsilon)$.

Proof. From Lemma 3 we have

$$\begin{aligned} \delta_j &= \sqrt{\frac{(1-\omega)(1+(j-1)\omega)}{1+(j-2)\omega}} \\ &= \sqrt{1-(j-1)\omega^2 + O(j^2\omega^4)} \\ &= 1 - (j-1)\omega^2/2 + O(j^2\omega^4) \\ &= 1 + O(\varepsilon) + O(\varepsilon^2). \end{aligned}$$

Similarly

$$\delta_j^{-1} = 1 + O(\varepsilon).$$

2.4. Analysis of the simple Lanczos algorithm.

There are two quantities at hand, which could be the object of an error analysis of the Lanczos algorithm: the Lanczos vectors q_j and the matrix T_j formed by the a_i and β_i . It is important to note the following fact at the outset of any further analysis: if a matrix $\tilde{A}$ is close to A this does not imply that the sequence of Lanczos vectors computed from $\tilde{A}$ is in any way close to the sequence of Lanczos vectors computed from A , as the following example shows.

Example. Consider

$$A = \begin{bmatrix} 1 & \eta & 0 \\ \eta & 2 & 3 \\ 0 & 3 & 4 \end{bmatrix}; \quad q_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

where η is a real parameter. Then the first Lanczos step yields:

$$Aq_1 = \begin{bmatrix} 1 \\ \eta \\ 0 \end{bmatrix}; \quad a_1 = 1; \quad q_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

However for the same q_1 and

$$\tilde{A} = \begin{bmatrix} 1 & 0 & \eta \\ 0 & 2 & 3 \\ \eta & 3 & 4 \end{bmatrix}$$

one obtains that

$$\tilde{A}q_1 = \begin{bmatrix} 1 \\ 0 \\ \eta \end{bmatrix}; \quad \tilde{a}_1 = q_1^T \tilde{A}q_1 = 1; \quad \tilde{q}_2 = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

Hence $q_2^T \tilde{q}_2 = 0$ independent of η , and even a small perturbation in the matrix may therefore result in totally different, i.e. orthogonal Lanczos vectors.

This example shows the limitations of a forward error analysis of the Lanczos algorithm. Since the Lanczos vectors may differ considerably, even when there is only a small perturbation, any error analysis, which attempts to compare the "ideal" (i.e. exact arithmetic) Lanczos vectors with the computed ones, has to run into considerable difficulties. Gear [9] avoids this difficulty in his forward analysis, by making the strong assumption a priori that the error between "ideal" and computed Lanczos vectors is small.

The example above indicates that the situation for the Lanczos algorithm is comparable to other methods for tridiagonalizing a symmetric matrix, like Givens' or Householder's method. The computed, intermediate quantities may differ from their ideal counterparts, but this is not the important issue if one performs a backward analysis. For these methods it can be shown that the computed tridiagonal matrix is exactly similar to a perturbation of the original matrix A , where the relative size of the perturbation is a modest multiple of the roundoff unit and $\|A\|$. We feel that a similar approach is also appropriate for the Lanczos algorithm, since the Lanczos vectors are ruled out as an object of a backward analysis by the above example. Our main attention will therefore be directed towards the matrix T_j .

Let us first remark that the loss of linear independence of the Lanczos vectors does not concern us here. From Lemma 1 we can conclude that as long as $\omega < \frac{1}{f-1}$ the Lanczos vectors are linearly independent. This means in a typical situation with $\epsilon = 10^{-16}$, $f = 100$, that ω can be as large as 10^{-7} , and the Lanczos vectors will be still linearly independent. Hence the level of orthogonality can grow by a factor of 10^{13} without affecting the linear independence of the q_k . The loss of linear independence is therefore a consequence rather than a cause of the loss of orthogonality, and will not concern

us for the moment.

The following theorem shows how the loss of orthogonality affects the matrix $\tilde{T}_j$.

Theorem 2.3. If ω , the level of orthogonality among the Lanczos vectors $q_1, q_2, \dots, q_{j+1}$, satisfies $\omega \leq \frac{1}{2} \frac{1}{j-2}$, then the computed tridiagonal matrix $\tilde{T}_j$ is similar to a matrix $\tilde{T}_j$, where $\tilde{T}_j$ is a perturbation of the orthogonal projection of A onto $\text{span}(Q_j)$. If A_p denotes this projection, then

$$\|A_p - \tilde{T}_j\| \leq 2\sqrt{j}\omega\beta_{j+1} + 2\epsilon\|A\| + O(\epsilon^2) \quad (2.4.1)$$

Proof. Since $q_1, q_2, \dots, q_{j+1}$ are linearly independent, the QR factorization of Q_j has the form $Q_j = N_j L_j^*$, where N_j is a $n \times j$ matrix with orthonormal columns, and L_j^* is a $j \times j$ upper triangular matrix with positive diagonal elements. Moreover N_j and L_j^* are uniquely determined. Since $\tilde{W}_j = Q_j^* Q_j = L_j^* N_j^* N_j L_j^* = L_j L_j^*$, L_j is also the Choleski factor of $\tilde{W}_j$.

Similarly $Q_{j+1} = N_{j+1} L_{j+1}^*$, where $N_{j+1} = [N_j \mid n_{j+1}]$ and

$$L_{j+1}^* = \begin{bmatrix} L_j^* & L_{j+1}^* \\ 0^* & \delta_{j+1} \end{bmatrix} = \begin{bmatrix} L_j^* & | & L_{j+1}^* \\ | & & | \\ 0^* & & \delta_{j+1} \end{bmatrix}$$

Now the basic Lanczos equation can be rearranged as follows:

$$A Q_j - Q_j \tilde{T}_j = \beta_{j+1} q_{j+1} e_j^* + F_j$$

$$Q_j^* Q_j \tilde{T}_j = Q_j^* A Q_j - \beta_{j+1} Q_j^* q_{j+1} e_j^* - Q_j^* F_j$$

$$L_j L_j^* \tilde{T}_j = L_j N_j^* A N_j L_j^* - \beta_{j+1} L_j N_j^* n_{j+1} e_j^* - L_j N_j^* F_j$$

$$L_j^* \tilde{T}_j L_j^{*-} = N_j^* A N_j - \beta_{j+1} N_j^* n_{j+1} e_j^* L_j^{*-} - N_j^* F_j L_j^{*-} \quad (2.4.2)$$

Let $\tilde{T}_j = L_j^* \tilde{T}_j L_j^{*-}$; then $\tilde{T}_j$ is similar to $\tilde{T}_j$. $\tilde{T}_j$ can now be considered as a perturbation of $A_p = N_j^* A N_j$, the orthogonal projection of A onto $\text{span}(Q_j)$. The

norm of this perturbation can be bounded by

$$\|A_p - \tilde{T}_j\| \leq \beta_{j+1} \|N_j^* n_{j+1} e_j^* L_j^{*-}\| + \|N_j^* F_j L_j^{*-}\|$$

Now $N_j^* N_{j+1} = \tilde{T}_j = [I_j \mid 0]$ and $L_j^{-1} e_j = \delta_j^{-1} e_j$, where δ_j is the bottom diagonal element of the Choleski factor L_j . As $\|N_j^*\| \approx 1$ and $\|L_j^{-1}\| \leq 2$ by Lemma 2, we obtain:

$$\begin{aligned} \|A_p - \tilde{T}_j\| &\leq \beta_{j+1} \|\delta_j^{-1} L_{j+1} e_j^*\| + 2\|F_j\| (1 + O(\epsilon)) \\ &\leq \beta_{j+1} \delta_j^{-1} \|L_{j+1}\| + 2\epsilon\|A\| + O(\epsilon^2) \end{aligned}$$

Now assume that the worst case happens, i.e. that $\tilde{W}_j = W_-$, where W_- is the matrix defined in section 2.3. Then applying the results of Lemma 4 one obtains

$$\begin{aligned} \|A_p - \tilde{T}_j\| &\leq \beta_{j+1} \sqrt{\frac{1-j-2\omega}{(1+\omega)(1-j-1)\omega}} \omega \sqrt{\frac{1+\omega}{(1-j-1)\omega(1-j\omega)}} \sqrt{j} + 2\epsilon\|A\| + O(\epsilon^2) \\ &\leq 2\omega\sqrt{j}\beta_{j+1} + 2\epsilon\|A\| + O(\epsilon^2) \end{aligned}$$

Theorem 2.3 says that the norm of the perturbation is proportional to the level of orthogonality among the Lanczos vectors, as long as the loss of linear independence among the Lanczos vectors is not imminent. The conclusion of Theorem 2.3 no longer holds when the Lanczos vectors begin to lose their linear independence, because then $\|L_j^{-1}\|$ and δ_j^{-1} can no longer be bounded.

It is also interesting to note that an attempt to relate $\tilde{T}_j$ to the projection of a perturbed matrix fails. In this case one would write

$$L_j^* \tilde{T}_j L_j^{*-} = N_j^* (A - \beta_{j+1} N_{j+1} L_{j+1}^{-1} e_j^* L_j^{*-} N_j^* - F_j L_j^{-1} N_j^*) N_j \quad (2.4.3)$$

instead of (2.4.2). Then $\|N_j^* L_{j+1}^{-1} e_j^* L_j^{*-} N_j^*\| \leq \|q_{j+1} n_j^* \delta_j^{-1}\| \leq \delta_j^{-1} < 1 + \omega$. This is not surprising, considering the example at the beginning of this section.

Another important quantity to consider is the matrix W_j , which ideally should be the identity matrix I_j . Much of the material in the previous section was related to the question, how much we can allow W_j to deviate from I_j , and yet satisfy some useful properties enjoyed by I_j . Lemma 5 is the most important result in this direction. This Lemma gives a first insight as to, why the particular bound $\omega \leq \sqrt{\frac{2\epsilon}{j-1}}$ is crucial for the Lanczos algorithm. In this case W_j has still, at least up to roundoff, the property of the identity matrix, that its lower triangular part is also its Choleski factor. This can be also seen by writing $W_j = (I - R_j^T)(I - R_j) = I - R_j^T R_j + R_j^T R_j$. Now if the bound on omega holds $R_j^T R_j$ becomes negligible like ϵ . It seems that this property of W_j is enough to assure that the Lanczos algorithm in finite precision behaves up to roundoff like its ideal counterpart. This will be shown in the following

Theorem 2.4. If $(j-1)\omega^2/2 \leq \epsilon$, i.e. if

$$\omega \leq \sqrt{\frac{2\epsilon}{j-1}} \quad (2.4.4)$$

then

$$N_j^T A N_j = T_j + V_j \quad (2.4.5)$$

where the elements of V_j are of order $O(\epsilon \|A\|)$, and N_j is the matrix as defined in Theorem 2.3.

Proof. Since $N_j^T A N_j$ is symmetric, it is sufficient to show by induction that the last column of $N_j^T A N_j$ and T_j differ only by a vector of order $O(\epsilon \|A\|)$.

For $j=1$ this is trivial since $n_1 = q_1$.

For general j transpose and rearrange equation (2.4.2) from Theorem 2.3:

$$N_j^T A N_j = L_j^{-1} T_j L_j + \beta_{j+1} L_j^{-1} e_j e_j^T L_j^{-1} + L_j^{-1} F_j^T N_j$$

The last column of $N_j^T A N_j$ then is given by

$$N_j^T A N_j e_j = L_j^{-1} T_j L_j e_j + \beta_{j+1} L_j^{-1} e_j e_j^T L_j^{-1} e_j + L_j^{-1} F_j^T N_j e_j \quad (2.4.6)$$

The proof now rests upon the fact that the last two terms in (2.4.6) are small and that $L_j^{-1} T_j L_j$ is a lower Hessenberg matrix and thus only the last two elements of $L_j^{-1} T_j L_j e_j$ are nonzero. Since L_j and L_j^{-1} are almost like the identity matrix, we should obtain that $L_j^{-1} T_j L_j e_j \approx \alpha_j e_j + \beta_j e_{j-1}$. In the remaining part of the proof we are going to show that these statements are true.

Let the elements in the bottom right corner of L_j be denoted by

$$L_j = \begin{bmatrix} & & & \\ & & & 0 \\ & & \eta_{j-1,j-1} & \\ & & \eta_{j,j-1} & \eta_{j,j} \end{bmatrix}$$

Then the corresponding elements of L_j^{-1} are

$$L_j^{-1} = \begin{bmatrix} & & & \\ & & & 0 \\ & & \eta_{j-1,j-1}^{-1} & \\ & & \eta_{j,j-1}^{-1} & \eta_{j,j}^{-1} \end{bmatrix}$$

$$\text{where } \eta_{j,j-1}^{-1} = -\frac{\eta_{j,j-1}}{\eta_{j,j} \eta_{j-1,j-1}}$$

With this notation we can evaluate the terms in (2.4.6) further.

$$\begin{aligned} L_j^{-1} T_j L_j e_j &= L_j^{-1} T_j (\eta_{j,j} e_j) = \eta_{j,j} L_j^{-1} (\alpha_j e_j + \beta_j e_{j-1}) \\ &= \eta_{j,j} \alpha_j L_j^{-1} e_j + \eta_{j,j} \beta_j L_j^{-1} e_{j-1} \\ &= \eta_{j,j} \alpha_j \eta_{j,j}^{-1} e_j + \eta_{j,j} \beta_j (\eta_{j-1,j-1}^{-1} e_{j-1} + \eta_{j,j-1}^{-1} e_j) \\ &= \alpha_j e_j + \eta_{j,j} \eta_{j-1,j-1}^{-1} \beta_j e_{j-1} + \eta_{j,j} \eta_{j,j-1}^{-1} \beta_j e_j \end{aligned}$$

For the second term one obtains

$$\beta_{j+1} L_j^{-1} e_j e_j^T L_j^{-1} e_j = \beta_{j+1} \eta_{j,j}^{-1} \eta_{j+1,j} e_j$$

Here $\eta_{j,j+1}$ is the j -th element of the vector l_{j+1} .

Applying Lemma 5 we obtain now that

$$\eta_{j-1,j-1} = 1 + O(\varepsilon)$$

$$\eta_{jj} = 1 + O(\varepsilon) .$$

and therefore also $\eta_j^j = 1 + O(\varepsilon)$, $\eta_{j-1,j-1}^{-1} = 1 + O(\varepsilon)$. Since there is no element growth during a Choleski factorization, and we assumed in (2.1.7) that $|\omega_{j,j-1}| \leq \varepsilon$, we also have that

$$\eta_{j,j-1} = O(\varepsilon) ,$$

$$\eta_{j+1,j} = O(\varepsilon) .$$

Then also

$$\eta_j^{j-1} = -\eta_{j,j-1} \eta_j^j \eta_{j-1,j-1}^{-1} = O(\varepsilon)(1 + O(\varepsilon))^\varepsilon = O(\varepsilon) .$$

Taking all this together we have

$$\begin{aligned} N_j^j A N_j e_j &= \alpha_j e_j + (1 + O(\varepsilon))^\varepsilon \beta_{j,j-1} + (1 + O(\varepsilon)) O(\varepsilon) \beta_{j+1,j} + L_j^{-1} F_j^* \eta_j \\ &= \alpha_j e_j + \beta_j e_{j-1} + (\beta_j e_{j-1} + \beta_{j+1} e_j) O(\varepsilon) + L_j^{-1} F_j^* \eta_j . \end{aligned}$$

Since we can assume that $\beta_j + \beta_{j+1} \leq \|A\|$, we are almost done. It only remains to be shown that also the elements of $L_j^{-1} F_j^* \eta_j$ are also $O(\varepsilon \|A\|)$. But as $\|L_j^{-1}\| \leq 2$ by Lemma 2.2, it follows that

$$\|L_j^{-1} F_j^* \eta_j\| \leq 2\varepsilon \|A\| = O(\varepsilon \|A\|)$$

and this concludes the proof .

In the proof of Theorem 2.4 it was assumed that $\frac{1}{2}(j-1)\omega^2$ implies $\omega \leq \frac{1}{j-1}$. This will be only true, as long as $j \leq \varepsilon^{-1}$, a condition which will be satisfied for all practical applications of the Lanczos algorithm. From now on we will assume that $j \ll \varepsilon^{-1}$.

Theorem 2.4 sets the stage for the next section. If it is possible to keep orthogonality at a level of ε , then the Lanczos algorithm actually computes a

matrix T_j , which is, up to roundoff, the orthogonal projection of A onto $\text{span}(Q_j)$, even though the Lanczos vectors themselves are no longer orthogonal to working precision.

2.5. Semiorthogonalization Strategies.

As soon as the level of orthogonality deteriorates so much that $|q'_{j+1}q_k| > \omega_0 = \sqrt{\frac{2\epsilon}{j-1}}$ for some $k < j$, the nice result $T_j = N'_j A N_j$ does not hold any longer. The main goal in this section is to show that if by some means semiorthogonality (i.e. $|q'_k q_k| \leq \sqrt{\epsilon}$ for $k \neq j$) can be maintained among the Lanczos vectors, the result of Theorem 2.4 will still hold for the modified algorithm.

Traditionally one was advised to perform the Lanczos algorithm with full reorthogonalization of the Lanczos vectors (Lanczos [17], Wilkinson [41]). This modification aims at maintaining orthogonality to working precision among the Lanczos vectors. Theorem 2.4 shows that not all this effort is necessary. More recently selective orthogonalization (Parlett and Scott [32]) and periodic reorthogonalization (Grear [9]) have been suggested as means of keeping semiorthogonality among the Lanczos vectors. The analysis for all these orthogonalization methods can be unified with the concept of a semiorthogonalization strategy for the Lanczos algorithm.

Suppose at the j -th step of the Lanczos algorithm

$$\beta_{j+1} q'_{j+1} = A q_j - \alpha_j q_j - \sum_{k=1}^{j-1} \xi_k q_k - f'_j$$

$$\text{and } |q'_{j+1} q_k| = |\omega_{j+1,k}| > \omega_0 \text{ for some } k < j.$$

Then we choose $j-1$ real numbers $\xi_1, \dots, \xi_{j-1}$ and form

$$\beta_{j+1} q'_{j+1} = \beta_{j+1} q'_{j+1} - \sum_{k=1}^{j-1} \xi_k q_k - f_j \quad (2.5.1)$$

The algorithm will be continued with q'_{j+1} instead of q_{j+1} . This modification of the Lanczos algorithm will be called a *semiorthogonalization strategy* if the following conditions are satisfied:

- 1) The numbers ξ_k , $k=1, \dots, j-1$ are chosen such that

$$|q'_k q'_{j+1}| = |\omega_{j+1,k}| \leq \omega_0 \quad (2.5.2)$$

$$(2.5.2)$$

$$\text{where } \omega_0 = \sqrt{\frac{2\epsilon}{j-1}} \text{ and } |q'_k q_{j+1}| \leq \epsilon_1.$$

- 2) The computation of the ξ_k 's and the formation of q'_{j+1} causes at most roundoff errors of $O(\epsilon \|A\|)$, i.e. we have

$$\beta_{j+1} q'_{j+1} = A q_j - \alpha_j q_j - \sum_{k=1}^{j-1} \xi_k q_k - f_j \quad (2.5.3)$$

$$\text{and } \|f_j\| \leq \epsilon \|A\|.$$

It seems somewhat artificial not to include an orthogonalization against q_j in (2.5.1). But we assumed initially in (2.1.7) that the simple Lanczos algorithm produces already a q'_{j+1} with $|q'_j q'_j| \leq \epsilon_1$. A proof analogous to (3.3.5) shows that then also $|q'_j q_j| \leq \epsilon_1$ holds. An extra orthogonalization of q'_{j+1} against q_j is hence unnecessary (c.f. the remarks following (2.1.7)).

All orthogonalization methods mentioned above can be summarized under the new concept of a semiorthogonalization strategy. The details, which are nontrivial in the case of selective orthogonalization, will be discussed later. Surprisingly under very general assumptions we can prove the following

Theorem 2.5. Let T_j be the tridiagonal matrix computed by the Lanczos algorithm with a semiorthogonalization strategy. Then $N'_j A N_j$, the orthogonal projection of A on $\text{span}(Q_j)$ satisfies

$$N'_j A N_j = T_j + V_j \quad (2.5.4)$$

where the elements of V_j are of order $O(\epsilon \|A\|)$.

Proof. For a certain number of steps the algorithm will be just the ordinary Lanczos algorithm and Theorem 2.4 can be applied. Suppose now at step j for the first time the semiorthogonalization strategy comes into play:

$$\beta_{j+1}g_{j+1} = Ag_j - \alpha_j g_j - \sum_{s=1}^{j-1} \beta_{js} g_{j+1} - \sum_{s=1}^{j-1} \epsilon_{js} g_s - f_j$$

or in terms of matrices:

$$\begin{aligned} A Q_j &= Q_j T_j + \sum_{s=1}^{j-1} \epsilon_{js} g_s e_j^* + \beta_{j+1} g_{j+1} e_j^* + F_j \\ &= Q_j (T_j + \sum_{s=1}^{j-1} \epsilon_{js} e_s e_j^*) + \beta_{j+1} g_{j+1} e_j^* + F_j \end{aligned} \quad (2.5.5)$$

Transposing and multiplying by Q_j one obtains

$$Q_j^* A Q_j = (T_j + \sum_{s=1}^{j-1} \epsilon_{js} e_s e_j^*) Q_j^* Q_j + \beta_{j+1} e_j^* g_{j+1} Q_j + F_j^* Q_j$$

As before let $Q_j = N_j L_j^*$, $Q_j^* Q_j = L_j L_j^*$, then

$$N_j^* A N_j = L_j^{-1} (T_j + \sum_{s=1}^{j-1} \epsilon_{js} e_s e_j^*) L_j + \beta_{j+1} L_j^{-1} e_j^* g_{j+1} + L_j^{-1} F_j^* N_j$$

Now comes the important observation when we consider as before the j -th column of the matrix $N_j^* A N_j$. The perturbation term in L_j simply cancels out, as $e_s^* L_j e_j = \eta_{js} e_s^* e_j = 0$, for $s = 1, \dots, j-1$. Thus

$$N_j^* A N_j = L_j^{-1} T_j \eta_{jj} e_j + \beta_{j+1} L_j^{-1} e_j^* g_{j+1} + L_j^{-1} F_j^* \eta_j \quad (2.5.6)$$

Now we can estimate the terms in (2.5.6) in the same way as in Theorem 2.4, and the result follows.

Suppose that from step j onward, at each step, an orthogonalization occurs. If not, then we can simply set the corresponding $\epsilon_s = 0$. Then the governing equation (2.5.5) at step $m > j$ can be written:

$$A Q_m - Q_m T_m = \beta_{m+1} g_{m+1} e_m^* + F_m$$

where

$$F_m = T_m + \sum_{i=1}^m \sum_{s=1}^m \epsilon_{is} e_i e_s^*$$

However it is clear that again $F_m^* e_m = T_m^* e_m = T_m e_m$, and then the argument of Theorem 2.4 can be also used for the general case.

At the first glance the result of Theorem 2.5 is very surprising. Because any orthogonalization appears to be such a disruption of the otherwise simple structure of the Lanczos algorithm, that one might expect the output of the algorithm to be changed drastically as well. But this is only true if one thinks in terms of the exact algorithm. There the matrix T_j loses its simple tridiagonal structure, when it is modified to $\tilde{T}_j$. In finite precision the quantity to consider is not T_j , but $L_j^* T_j L_j^{-1}$, which is almost the exact projection of A on $\text{span}(Q_j)$. Moreover it is an upper Hessenberg matrix but so is $L_j^* T_j L_j^{-1}$. Therefore the modification of T_j due to an orthogonalization actually does not change the structure of the important quantities in the algorithm. This explains the relative ease with which Theorem 2.5 follows from Theorem 2.4.

In order to prove Theorem 2.5 within our model we had to assume that the semiorthogonalization strategy maintains a level of orthogonality of $\epsilon_0 = \sqrt{\frac{2\epsilon}{j-1}}$ among the Lanczos vectors. The dependence on j in ϵ_0 is a nuisance, since the practical experience shows that semiorthogonality, i.e. a $\sqrt{\epsilon}$ level, is enough for computing an accurate T_j . The reason that Theorem 2.5 is weaker than we would like to have, resides in the assumption implicitly made by using Lemma 3 that all offdiagonal elements of $\tilde{W}_j$ assume the maximum value ϵ_0 . In the view of Theorem 2.1 this is an unrealistic assumption. Only some elements in the last column of $\tilde{W}_j$ will be as large as ϵ_0 and force

an orthogonalization. The majority of off-diagonal elements of H_j will be well below this threshold. Therefore the use of $\sqrt{\epsilon}$ for a practical algorithm is justified, although we did not prove it rigorously.

2.5.1. Partial Reorthogonalization.

Originally the Lanczos algorithm was executed only with full reorthogonalization (PRO). This amounted to an orthogonalization of the new q_{j+1} against all previous q_j at every step, i.e.

$$r_j = \beta_{j+1} q'_{j+1} = Aq_j - \alpha_j q_j - \beta_j q_{j-1} - f_j$$

$$r_j = r'_j - \sum_{k=1}^j (r'_j, q_k) q_k.$$

It is clear that PRO will satisfy (2.5.2) and (2.5.3) for a general semiorthogonalization strategy. Actually we expect that $|q'_j q_{j+1}| \leq \sqrt{n} \epsilon$, i.e. much more than necessary for (2.5.2).

There is a minor point still to be considered. In (2.5.1) we do not consider an orthogonalization against q_j . However since $|r'_j q_j| = \beta_{j+1} |q'_j q_j| \leq \epsilon \|A\|$, we can write the PRO as

$$\beta_{j+1} q'_{j+1} = Aq_j - \alpha_j q_j - \beta_j q_{j-1} - \sum_{k=1}^{j-1} \xi_k q_k - f_j,$$

where $f_j = f'_j + \beta_{j+1} (q'_j, q_j) q_j$. Therefore with $\xi_k = r'_j q_k$, $k = 1, \dots, j-1$, PRO is a semiorthogonalization strategy for the Lanczos algorithm and Theorem 2.5 holds. On the other hand Theorem 2.5 assures us that only a level of orthogonality of ϵ_0 among the Lanczos vectors is sufficient. PRO is therefore not efficient, since the extra orthogonality gained does not produce a more accurate T_j .

This insight is the basis for Grear's [9] periodic reorthogonalization. In this method one has to update an n -vector which simulates the error in the

current Lanczos vector as compared to the ideal Lanczos vector. If this estimate for the error rises above the $\sqrt{\epsilon}$ -level a full reorthogonalization of the current Lanczos vector against all the previous ones is performed. If the error estimate is correct, Grear's analysis shows that the Lanczos algorithm with periodic reorthogonalization computes a T_j which is accurate up to roundoff.

Periodic reorthogonalization can be improved in two ways by using the recurrence from Theorem 2.1. Based on this recurrence we only update a j -vector, which contains estimates $u_{j+1,k}$ for the terms $q'_j q_k$, $k = 1, \dots, j$. Secondly, since the $u_{j+1,k}$'s indicate against which previous Lanczos vectors orthogonality has been lost, the current Lanczos vector has to be orthogonalized only against some of the previous Lanczos vectors. The resulting new method is called *partial reorthogonalization* (PRO).

The success of PRO depends very much on an accurate estimate for $q'_j q_k$. This is not a trivial task since the recurrence (2.2.1) involves among others terms of the type $f_j q_k - f_k q_j$, which are not directly available in the algorithm, yet crucial for the recurrence. This problem is discussed in detail in Section 3.1. Similarly it is not obvious against which previous Lanczos vectors to orthogonalize when the recurrence signals that orthogonality beyond the threshold value of $\sqrt{\epsilon}$ has been lost. Of course PRO forces an orthogonalization against all q_k where $|q'_j q_k|$ exceeds the threshold. It turns out that it is most economical to perform orthogonalizations against "batches" of Lanczos vectors, containing the offending ones and a certain number of neighboring vectors. The details will be given in section 3.3.

The Lanczos algorithm with PRO at an abstract level therefore can be written as follows:

- 1) Perform a regular Lanczos step:

$$\beta_{j+1}q'_{j+1} = Aq_j - \alpha_j q_j - \beta_j q_{j-1} - f_j \quad (2.5.7a)$$

- 2) Update the estimates $w_{j+1,k}$ for $q'_{j+1}q_k$, for $k = 1, \dots, j$ using the recurrence (2.3.1).

- 3) Based on the information from the $w_{j+1,k}$, determine a set of indices $L(j) = \{k \mid 1 \leq k \leq j\}$ and compute

$$\beta_{j+1}q_{j+1} = \beta_{j+1}q'_{j+1} - \sum_{k \in L(j)} (\beta_{j+1}q'_{j+1}q_k) q_k - f_j \quad (2.5.7b)$$

Clearly with $\ell_k = \beta_{j+1}q'_{j+1}q_k$ PRO is a semiorthogonalization strategy. Theorem 2.5 can be applied and guarantees the computation of a T_j accurate up to the roundoff level.

2.5.2 Selective Orthogonalization

The previous section was a natural application of Theorem 2.5. In order to check whether selective orthogonalization (SO) is also a semiorthogonalization strategy for the Lanczos algorithm, let us first recall the result of Paige's theorem (Theorem 2.2), which forms the basis for SO.

Paige's theorem describes how the new vector q'_{j+1} behaves, when orthogonality is lost: it is tilted towards the vectors w_k , which are approximate eigenvectors for the matrix A . The quantity $\gamma_k/\beta_{j+1}\sigma_k$ is a measure for the loss of orthogonality in direction of a certain vector w_k . Our general assumptions on the Lanczos algorithm imply that γ_k is of the order of the roundoff unit. Let us therefore assume that $|\gamma_k| \leq \varepsilon \|A\|$. The only way that $w_k q'_{j+1}$ can become large is by $\beta_k = \beta_{j+1}\sigma_k$ becoming small. SO therefore computes and monitors some of the β_k . If one β_k becomes smaller than a

certain threshold value ε_j , then q'_{j+1} is orthogonalized against the corresponding w_k . The j -th step of the Lanczos algorithm with SO can therefore be written as follows:

- 1) Perform a regular Lanczos step:
- $$\beta_{j+1}q'_{j+1} = Aq_j - \alpha_j q_j - \beta_j q_{j-1} - f_j \quad (2.5.8a)$$

- 2) Determine the set

$$L(j) = \{k \mid 1 \leq k \leq j, \beta_k < \varepsilon_j\} \quad (2.5.8b)$$

- 3) Compute $w_k = q_j w_k$ for $k \in L(j)$. Then the next Lanczos vector is given by

$$\beta_{j+1}q_{j+1} = \beta_{j+1}q'_{j+1} - \sum_{k \in L(j)} (\beta_{j+1}q'_{j+1}w_k) w_k - f_j \quad (2.5.8c)$$

The set $L(j)$ may be empty, then nothing will be done in step 3. This is a simplified version of an actual implementation of SO, for example the w_k are not recomputed. However (2.5.8) catches the main features of SO, and it is sufficient to consider here as a model of the actual computation.

It is not obvious at all that SO as defined in (2.5.8) is a semiorthogonalization strategy. We want to show first that SO formally follows the pattern in (2.5.2).

$$\begin{aligned} \beta_{j+1}q_{j+1} &= \beta_{j+1}q'_{j+1} - \sum_{k \in L(j)} (\beta_{j+1}q'_{j+1}w_k) q_k - f_j \\ &= \beta_{j+1}q'_{j+1} - \sum_{j=1, k \in L(j)} (\beta_{j+1}q'_{j+1}w_k) \sigma_k q_k - f_j \end{aligned} \quad (2.5.9)$$

Recall that the eigen decomposition of T_j is given by $T_j S_j = S_j \Theta_j$, $S_j^* = (q_1, \dots, q_N)$. Also note that for the j -th term

$$|\ell_j| = \sum_{k \in L(j)} \frac{|\gamma_k|}{\beta_{j+1}\sigma_k} q_k = |L(j)| |\gamma_k| \leq \varepsilon \|A\|$$

$$w_{j+1} = \sum_{i=1}^j \phi_i \epsilon_i \quad \text{with } \phi_i = w_{j+1}^T \epsilon_i.$$

Then (2.5.12) becomes:

$$\begin{aligned} \beta_{j+1} w_{j+1} &= \beta_{j+1} \left(\sum_{i=1}^j \phi_i \epsilon_i - \sum_{i \in L(j)} \phi_i w_i \right) - Q_j^T f_j \\ &= \beta_{j+1} \left(\sum_{i \in L(j)} \phi_i \epsilon_i + \sum_{i \in L(j)} \phi_i (\epsilon_i - w_i) - Q_j^T f_j \right) \end{aligned} \quad (2.5.13)$$

Therefore

$$\begin{aligned} \beta_{j+1} \|w_{j+1}\| &\leq \beta_{j+1} (j - |L(j)|) \max_{i \in L(j)} |\phi_i| + |L(j)| \|w_j - w_j\| \max_{i \in L(j)} |\phi_i| + \epsilon \|A\| \\ &\leq \beta_{j+1} j \max_{i \in L(j)} |\phi_i| + j \|w_j - w_j\| \max_{i \in L(j)} |\phi_i| + \epsilon \|A\| \end{aligned} \quad (2.5.14)$$

We can estimate the terms in (2.5.14) further. Consider the definition of $L(j)$ in (2.5.8b). It follows that $\epsilon \phi L(j)$ in $\beta_{j+1} \sigma_k \geq \epsilon_j$. Using Paige's Theorem we have $\beta_{j+1} \sigma_k = \frac{\gamma_k}{\phi_1}$, hence for $i \in L(j)$ we can estimate:

$$|\phi_i| = \frac{|\gamma_k|}{\beta_{j+1} \sigma_k} \leq \frac{\epsilon \|A\|}{\epsilon_j} \leq \frac{\sqrt{\epsilon}}{j}. \quad (2.5.15)$$

with the choice of $\epsilon_j \geq j \|A\| \sqrt{\epsilon}$.

On the other hand if $i \in L(j)$, we simply estimate $\|\phi_i\| \leq \|w_{j+1}\|$, and $\|w_{j+1}\|$ can be estimated by using (2.3.7) and the semiorthogonality of the first j Lanczos vectors. One obtains

$$\beta_{j+1} \|\phi_i\| \leq \beta_{j+1} \|w_{j+1}\| \leq 2 \|A\| \sqrt{j \epsilon} + O(\epsilon \|A\|) \quad (2.5.16)$$

Finally using a result analog to Lemma 1 and again the semiorthogonality, it follows that

$$\|f_j - w_j\| \leq (j-1) \sqrt{\epsilon} \quad (2.5.17)$$

Assume now that $|L(j)| = k$, where k is a small integer, then substituting (2.5.15)-(2.5.17) into (2.5.14) it follows that

The effects of SO regarding the j -th Lanczos vector are hence of the same order as the roundoff unit and we can subsume them in the f_j -term. Hence SO is formally a semiorthogonalization strategy with

$$\epsilon_k = \sum_{i \in L(j)} (\beta_{j+1} w_{j+1}^T \epsilon_i) w_i, \quad k = 1, \dots, j-1.$$

The method of SO is due to Parlett and Scott [32]. They suggest the use of the threshold $\epsilon_j = \sqrt{\epsilon} \|A\|$ in order to maintain semiorthogonality among the Lanczos vectors. The following Theorem shows, from a different perspective, why this is the right choice.

Theorem 2.8. If the first j Lanczos vectors are semiorthogonal, if $\|w_k\| \leq \epsilon \|A\|$, and if ϵ_j is chosen such that $\epsilon_j \geq j \|A\| \sqrt{\epsilon}$, then one step of SO according to (2.5.8) produces a vector q_{j+1} such that

$$\|Q_j^T q_{j+1}\| \leq \sqrt{\epsilon} + O(j^{-1} \|A\| \epsilon) \quad (2.5.10)$$

Proof. Let $w_k = Q_j^T q_k$ for $k = 1, \dots, j+1$ and $w_{j+1} = Q_j^T q_{j+1}$. Then $w_j = (w_1, \dots, w_j)$. Let $S_j = (s_1, \dots, s_j)$ be defined as before. Multiplying (2.5.8) by Q_j^T and using this notation, one obtains:

$$\begin{aligned} \beta_{j+1} w_{j+1} &= \beta_{j+1} w_{j+1} - \beta_{j+1} \sum_{i=1}^j (q_i^T w_{j+1}) \sigma_i w_i - Q_j^T f_j \\ &= \beta_{j+1} w_{j+1} - \beta_{j+1} \sum_{i \in L(j)} (w_{j+1}^T \epsilon_i) \sum_{k=1}^j \sigma_k w_k - Q_j^T f_j \end{aligned} \quad (2.5.11)$$

Because of the symmetry of w_j it follows that

$$\beta_{j+1} w_{j+1} = \beta_{j+1} (w_{j+1} - \sum_{i \in L(j)} (w_{j+1}^T \epsilon_i) w_i) - Q_j^T f_j \quad (2.5.12)$$

Since w_{j+1} is a j -vector it can be expanded in terms of the orthonormal vectors s_k .

$$\begin{aligned} \beta_{j+1} \|w_{j+1}\| &\leq \beta_{j+1} \sqrt{\frac{\epsilon}{2}} + \epsilon(j-1) \sqrt{\epsilon} (2\|A\| \sqrt{j\epsilon} + O(\epsilon\|A\|)) + O(\epsilon\|A\|) \\ &\leq \beta_{j+1} \sqrt{\epsilon} + O(j^{\frac{1}{2}}\|A\|\epsilon). \end{aligned} \quad (2.5.18)$$

Now it can be shown that $\beta_{j+1} = (1 + O(\epsilon))\beta_{j+1}$ and (2.5.10) follows.

In order to appreciate Theorem 2.8 several more remarks are necessary. The proof of Theorem 2.8 seems to indicate that from an SO point of view it would be more natural to define the level of orthogonality by using $\|Q'_{j+1}\|$ instead of using $\|Q'_{j+1}\|_{\infty}$ as we did. Assuming that $\|Q'_{j+1}\| < \sqrt{\epsilon}$, it would be possible to prove (2.5.10) with an $O(j\epsilon\|A\|)$ term. With this more realistic interpretation Theorem 2.8 indeed shows that SO maintains semiorthogonality among the Lanczos vectors in the sense that $\|Q'_{k+1}\| \leq \sqrt{\epsilon}$ for $k = 1, \dots, j$.

Formula (2.5.13) makes clear how SO goes about maintaining semiorthogonality. The loss of orthogonality vector w'_{j+1} is decomposed into its eigencomponents. The components which have grown too large ($i \in L(j)$) are reduced by orthogonalization to roundoff level, the other components ($i \notin L(j)$) remain unchanged. The key to the understanding why SO maintains semiorthogonality hence lies in (2.5.13). The remaining part of the proof of Theorem 2.8 only translates the informal argument above into exact estimates.

Formula (2.5.13) also illustrates why SO had some problems in gaining wide acceptance as a means of maintaining semiorthogonality. The proper way to study SO is in terms of the W_k or as in (2.5.13) in terms of the z_k . This is conceptually more difficult than the apparent and "natural" way to study SO in terms of the Lanczos vectors. This different point of view only involves a change of basis in $\text{span}(Q_j)$, however the failure to recognize this prompted

wrong judgements about SO, e.g. [9,p.124].

Finally by making the requirements on ϵ_j more stringent, it is possible to show that SO will also be able to maintain a level of orthogonality of ω_0 . To be precise, we have the following

Corollary. If the level of orthogonality among the Lanczos vectors is ω_0 , if $7\omega_0 \leq \epsilon\|A\|$, and if ϵ_j is chosen such that $\epsilon_j \geq j^{\frac{1}{2}}\|A\|\sqrt{\frac{\epsilon}{2}}$, then one step of selective orthogonalization produces a q_{j+1} such that

$$\max_{1 \leq k \leq j} |q'_j, q_k| \leq \omega_0 + O(j\epsilon\|A\|).$$

The proof is analog to Theorem 2.8. With the help of this corollary we can apply Theorem 2.5, and it follows that also SO produces a matrix $\tilde{T}_j$, which is accurate up to roundoff.

2.6. Applications.

So far we have discussed the Lanczos algorithm in finite precision only as a way of tridiagonalizing the given matrix A . The main application we had in mind, however, was solving linear systems of equations. Recall from section 1.1. that in order to compute an approximate solution vector x_j to $Ax = b$, we solved $T_j h_j = \beta_j e_1$, and then computed $x_j = Q_j h_j$. Suppose that we have employed some semiorthogonalization strategy, computed T_j and Q_j , and determine now

$$x_j = Q_j T_j^{-1} \beta_j e_1. \quad (2.6.1)$$

In this case it is easier to compare x_j to z_j , rather than to estimate $b - Ax_j$. Here z_j is the best approximation from $\text{span}(Q_j)$, i.e. using orthogonal projections,

$$z_j = N_j (N_j^* A N_j)^{-1} N_j^* b. \quad (2.6.2)$$

Recall that $N_j N_j^*$ is the orthogonal projector onto $\text{span}(Q_j)$, where $N_j = Q_j L_j^*$ is defined as in section 2.4. Since $q_1 = n_1$, we have $z_j = N_j (N_j^* A N_j)^{-1} \beta_j e_1$. According to Theorem 2.5 we have that $T_j + V_j = N_j^* A N_j$, where the elements of V_j are of $O(\epsilon \|A\|)$. Therefore it does not make any difference if we replace $N_j^* A N_j$ by T_j , since the error introduced this way in the computation of z_j is of the same order as the error which we have to take into account anyway when solving linear systems. It depends only on $\kappa(T_j)$, the condition number of T_j .

The only way that the finite precision Lanczos algorithm affects the computation of z_j versus x_j , is through the formation of x_j as a linear combination of the q_k , which are not orthogonal. This effect can be estimated by comparing Q_j with N_j :

$$\|Q_j - N_j\| \leq \|N_j L_j^* - N_j\| = \|L_j^* - I_j\| \leq \sqrt{2j}\epsilon, \quad (2.6.3)$$

where we have used that the level of orthogonality among the Lanczos vectors is ϵ_0 . Hence if the Lanczos algorithm is used for solving linear systems of equations, and the required accuracy is not less than $\sqrt{\epsilon}$, the solution x_j computed from (2.6.1) is as good as the "ideal" solution obtainable from $\text{span}(Q_j)$. Only when a higher accuracy is required additional steps have to be taken (c.f. Parlett[30]).

If the algorithm is used for computing eigenvalues the situation is even better. Theorem 2.5 assures us that the eigenvalues of T_j are up to roundoff the Rayleigh-Ritz approximations from $\text{span}(Q_j)$. This is best we could hope for.

3. PARTIAL REORTHOGONALIZATION.

3.1. Computing the Loss of Orthogonality.

The method of partial reorthogonalization (PRO) was introduced in (2.5.7), but many detailed questions were left unanswered in Chapter 2. One of these questions concerns the evaluation of the recurrence

$$\begin{aligned} \omega_{k0} &= 1 & \text{for } k = 1, \dots, f \\ \omega_{k, k-1} &= \varepsilon_k & \text{for } k = 2, \dots, f \\ \beta_{j,1} \omega_{j+1,k} &= \beta_{k,1} \omega_{k+1,j} + (\alpha_k - \alpha_j) \omega_{j,k} + \beta_j \omega_{j-1,k} + q_j' f_k + q_k' f_j - q_k' f_{k-1} \end{aligned} \quad (3.1.1)$$

for $1 \leq k < j$, and $\omega_{j, j+1} = \omega_{k+1, j}$. Here $\omega_{k0} = 0$ and $\varepsilon_k = q_k' q_{k-1}$.

An accurate evaluation of (3.1.1) 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 immediately available in the event of an reorthogonalization. However (3.1.1) involves the local error vectors f_j , which are unknown unless one wants to compute them in double precision. But even this may be impossible if the matrix vector product is inaccessible and truly in black box form. The unavailability of f_j appears to make (3.1.1) useless, but there is a way to utilize (3.1.1) 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 computation of $\omega_{j,k}$. 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 $q_j' q_k$ can be simulated by replacing the unknown quantities by random values from appropriate ranges as follows:

$$\begin{aligned} \omega_{k0} &= 1 & \text{for } k = 1, \dots, f \\ \omega_{k, k-1} &= \psi_k & \text{for } k = 2, \dots, f \\ \omega_{j,1,k} &= \frac{1}{\beta_{j,1}} [\beta_{k,1} \omega_{k+1,j} + (\alpha_k - \alpha_j) \omega_{j,k} + \beta_j \omega_{j-1,k} - \beta_j \omega_{j-1,k}] + \psi_{jk} \end{aligned} \quad (3.1.2)$$

for $1 \leq k < j$, and $\omega_{j, j+1} = \omega_{k+1, j}$. Here $\omega_{k0} = 0$, and ψ_{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 (3.1.2) as the *computed* or *estimated* orthogonality components, in contrast to the true components which are given by the inner products $q_j' q_k$.

Formula (3.1.2) can be regarded as a simulation of how the loss of orthogonality would occur on a different machine which generated numbers ψ_{jk} and ψ_k as actual roundoff errors. From our interpretation of Theorem 2.1 we concluded that the loss of orthogonality mainly depends on the α_j and β_j , and from Theorem 2.3 and 2.4 we know that the computed α_j and β_j are exact up to roundoff. Therefore the computed loss of orthogonality from formula (3.1.2) will behave like the true loss of orthogonality as soon as the ω_{jk} 's reach the critical region of about $\sqrt{\varepsilon}$.

This is illustrated by the following examples, where we examined the dependence of formula (3.1.2) on the choice of ψ_{jk} and ψ_k . For a matrix of order $n = 128$, which is part of the matrix in Example 1, Chapter 4, and with starting vector $q_1' = (1, \dots, 1) / \sqrt{128}$, 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 (3.1.2). 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 $\psi_{jk} \in N(0, \varepsilon)$, with $\varepsilon = 1.0, 10.0, 100.0, 1000.0$. Then we kept ψ_{jk} fixed and varied ψ_k . The true and the estimated loss of

orthogonality are plotted in Figure 3.1.

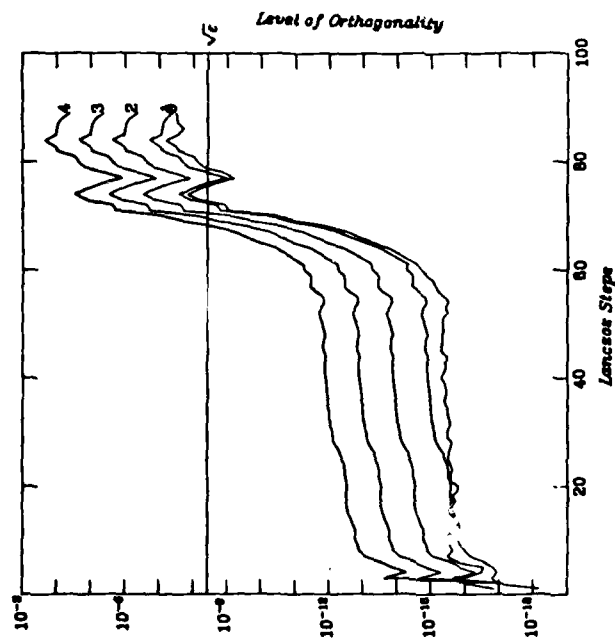


Figure 3.1. True and estimated level of orthogonality for various choices of α .

- 0 - True level of orthogonality
- 1 - Estimated level of orthogonality, $\alpha = 1.0$
- 2 - Estimated level of orthogonality, $\alpha = 10.0$
- 3 - Estimated level of orthogonality, $\alpha = 100.0$
- 4 - Estimated level of orthogonality, $\alpha = 1000.0$

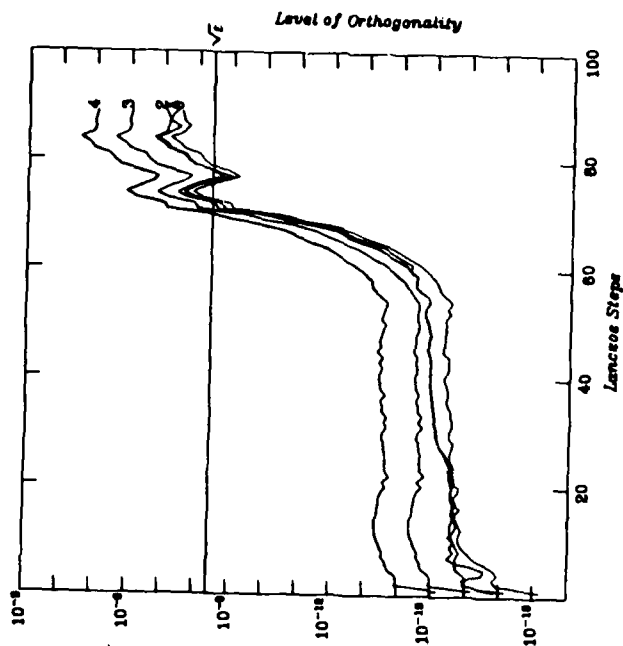


Figure 3.2. True and estimated level of orthogonality for various choices of ψ .

(Graphs labeled as in Figure 3.1.)

Figures 3.1 and 3.2 show that the estimated level of orthogonality with formula (3.1.2) 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 $\sqrt{\epsilon}$ region. Even the curve with the largest overestimate reaches the $\sqrt{\epsilon}$ threshold only three steps to early at step 68. In spite of the dependence on the random terms, (3.1.2) therefore appears to produce a quite accurate estimate for the level of orthogonality.

We repeated these tests with the example from Figure 2.1. In this example the level of orthogonality reaches the threshold of $\sqrt{\epsilon}$ after 11 steps. The behavior is shown in Figures 3.3 and 3.4.

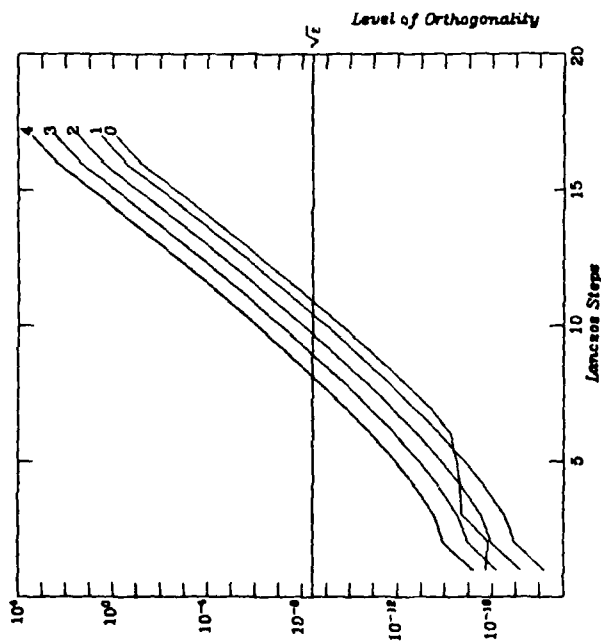


Figure 3.3. True and estimated level of orthogonality for various choices of v_p .
(Graphs labeled as in Figure 3.1.)

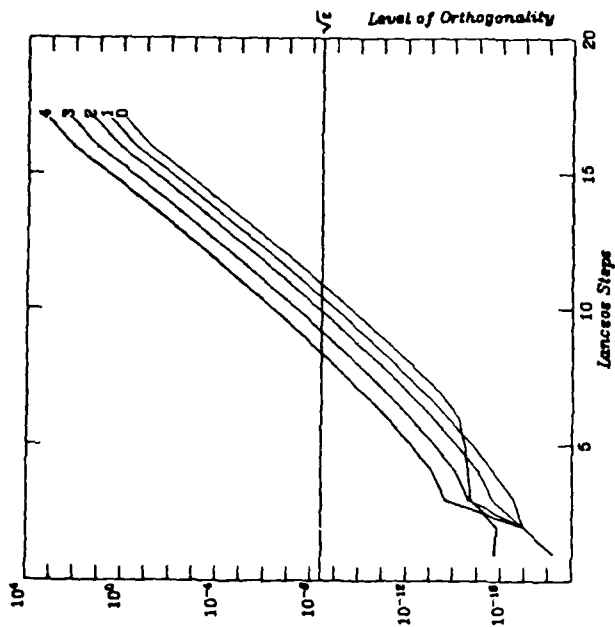


Figure 3.4. True and estimated level of orthogonality for various choices of v_p .
(Graphs labeled as in Figure 3.1.)

The conclusions we can draw from Figures 3.3 and 3.4 are the same as from Figures 3.1 and 3.2. No matter whether the level of orthogonality begins to increase early or late, the recurrence (3.1.2) yields a qualitatively quite accurate estimate of the true level of orthogonality in the sense that (3.1.2) signals at about the right Lanczos step that the $\sqrt{\epsilon}$ -level has been

n	mean	stand. dev.	no. of items
40	1.2531	0.5375	192
160	1.1863	0.3786	302
640	1.2304	0.6449	315
960	1.3279	0.8687	324

Table 3.1. $\|f_j\|/(\beta_{j+1}, \epsilon)$

The following figure gives some more information about the distribution of $\|f_j\|/(\beta_{j+1}, \epsilon)$:

Frequency in intervals of Length 0.1

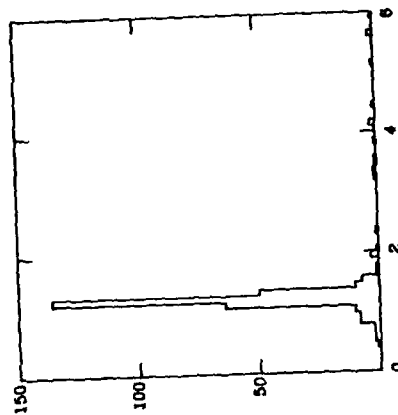
Figure 3.5. Distribution of $\|f_j\|/(\beta_{j+1}, \epsilon)$ for $n=960$.

Table 3.1 and Figure 3.5 show that in all our examples $\|f_j\|/\omega_{j+1, \epsilon}$. In no case $\|f_j\|$ was larger than $6\beta_{j+1, \epsilon}$. This means that in the examples considered here in most of the cases the evaluation of $A_{ij} - \alpha_j \beta_j - \beta_j \beta_{j-1}$ was exact up to one unit in the last place.

reached. These tests also show that the recurrence is relatively insensitive to moderate overestimates in the error terms. For example, as Figure 3.1 shows, an increase in the estimate for the $q_j^2/\alpha_j - q_j^2/\beta_j$ -term by a factor 1000, resulted in ω_{j+1} 's which reached the threshold only 3 steps too early. For a practical computation of the loss of orthogonality with (3.1.2) 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 (3.1.2) so accurately that the direct computation of q_j^2/α_j can be saved and the values ω_{j+1} can be used instead in the reorthogonalization process.

In order to obtain more information about the behavior of the $q_j^2/\alpha_j - q_j^2/\beta_j$ -terms and q_j^2/α_j it is quite useful to study first $\|f_j\|$. For that purpose we computed $\|f_j\|/(\beta_{j+1}, \epsilon)$ in double precision¹ for test runs of the single precision Lanczos algorithm with a set of twelve test matrices. At each step the loss of orthogonality was computed directly and a full reorthogonalization was performed, when $\sqrt{\epsilon}$ was exceeded. The algorithm terminated either after 50 steps or when the residual norm was reduced by a factor of 10^{-3} . The tests were repeated for matrices of order $n = 40, 160, 640, 960$. The following results were obtained:

¹All 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 rounded unit $\epsilon \approx 2 \cdot 10^{-8}$, for double precision $\epsilon \approx 2 \cdot 10^{-16}$.

This information about $\|f_j\|$ is very helpful for the study of $q_j'f_k - q_k'f_j$. Since q_j and q_k are unit vectors, $\|f_j\| = \beta_{j,j}$, and $\|f_k\| = \beta_{k,k}$, we expect that $(q_j'f_k - q_k'f_j)/(\beta_{j,j} + \beta_{k,k})$ is. This expectation proved to be correct. The results of the test runs with the same test matrices as before are summarized in the following table:

n	mean	stand. dev.	no. of items
40	-0.003858	0.1423	2898
180	-0.004855	0.09503	5100
840	-0.007217	0.08282	4582
960	-0.001733	0.08580	5188

Table 3.2. $\frac{q_j'f_k - q_k'f_j}{\epsilon(\beta_{j,j} + \beta_{k,k})}$

Based on the results in Table 3.2 we decided to set

$$\phi_k = \epsilon(\beta_{k,k} + \beta_{j,j})\theta \quad (3.1.3)$$

where $\theta \in N(0,0.3)$. θ is more than twice the largest standard deviation observed in Table 3.2 and will therefore yield an estimate for $q_j'f_k - q_k'f_j$ which will be too large in most of the cases. This is desirable for PRO according to the discussion above.

In order to determine some estimates for $q_j'q_k$ we used the following relation

$$\beta_{j,j}q_j'q_{j+1} = q_j'q_j - \alpha_j q_j'q_j - \beta_j q_j'q_{j-1} - q_j'f_j \quad (3.1.4)$$

Therefore

$$q_j'q_{j+1} = -\frac{\beta_j}{\beta_{j,j}}q_j'q_{j-1} + \delta_j \quad (3.1.5)$$

where $\delta_j = (q_j'q_j - \alpha_j q_j'q_j - q_j'f_j)/\beta_{j,j}$, and δ_j is at roundoff level. If in (3.1.5) the corresponding equations for $j, j-1, \dots$ are substituted, one obtains

$$q_j'q_{j+1} = -\frac{\beta_j}{\beta_{j,j}}q_j'q_{j-1} + \delta_j \quad (3.1.6)$$

where δ_j is the sum of all remaining terms. We therefore computed the term $\frac{\beta_{j,j}q_j'q_{j+1}}{\beta_{j,j}}$. The extra n was introduced, since we are trying to estimate the error of an inner product. The following results were obtained as before:

n	mean	stand. dev.	no. of items
40	-0.001594	0.2893	180
180	-0.004411	0.3167	290
840	-0.004258	0.3086	303
960	-0.008928	0.09782	312

Table 3.3. $\frac{\beta_{j,j}q_j'q_{j+1}}{\beta_{j,j}}$

According to these results we choose

$$\psi_k = \epsilon n \frac{\beta_k}{\beta_{j,j}} \psi \quad (3.1.7)$$

where $\psi \in N(0,0.8)$.

There is one more error term to be considered. After a reorthogonalization has been performed, the terms $q_j'q_k$ 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 computed $q_j'q_k/\epsilon$ for our set of test problems, whenever a reorthogonalization occurred.

n	mean	stand. dev.	no. of items
40	-0.1389	0.2412	892
180	-0.1859	0.3451	358
840	-0.2272	0.3743	578
960	-0.1703	0.8925	838

Table 3.4. $q_j'q_k/\epsilon$ after reorthogonalization.

Since the two vectors were already semiorthogonal before the reorthogonalization, the values in Table 3.4 are quite small, and there is no strong dependence on n . Based on this result we are choosing $\omega_{j+1} \in N(0, 1.5)$ after a reorthogonalization has been performed.

3.2. The Behavior of the Computed Level of Orthogonality.

In section 3.1 we discussed how an estimate for the level of orthogonality can be computed with formula (3.1.2) and appropriately chosen random numbers for ϕ_{jk} (3.1.3), ψ_k (3.1.7), and ω_{jk} after reorthogonalization (3.1.8). In this section we will examine how well the thus computed ω_{jk} reflects the behavior of the true level of orthogonality.

We tested (3.1.2) with several examples where a full reorthogonalization was performed whenever one ω_{jk} became larger than the threshold of $\sqrt{\epsilon}$. In Figures 3.6 and 3.7 the true level of orthogonality and the computed estimated are plotted for two of the sample runs.

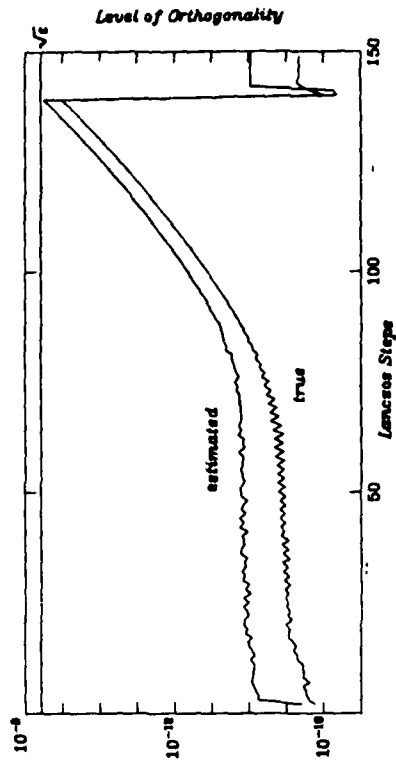


Figure 3.6.a. True and computed level of orthogonality for $A = \text{diag}(1, 2^2, \dots, 1000^2)$ and $q_j^* = (1, 1, \dots, 1)/\sqrt{1000}$.

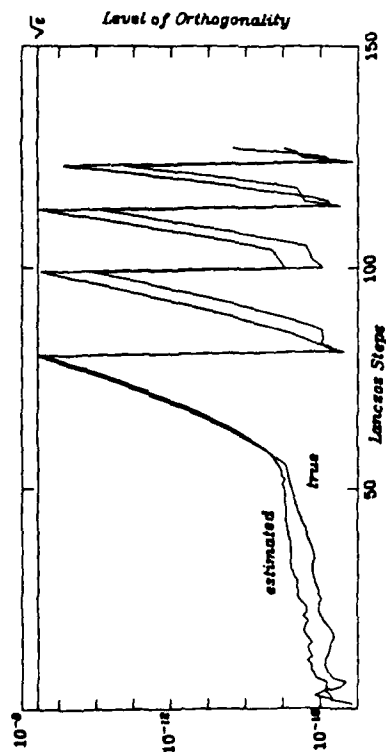


Figure 3.7. True and computed level of orthogonality for the matrix used in Figures 3.1 and 3.2.

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 $\epsilon^{\frac{2}{3}}$. 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 3.6 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 this is also true in a finite precision environment we

repeated the sample run from Figure 3.6.a with a similarity transformation of the diagonal matrix A . The starting vector was changed accordingly. We obtained:

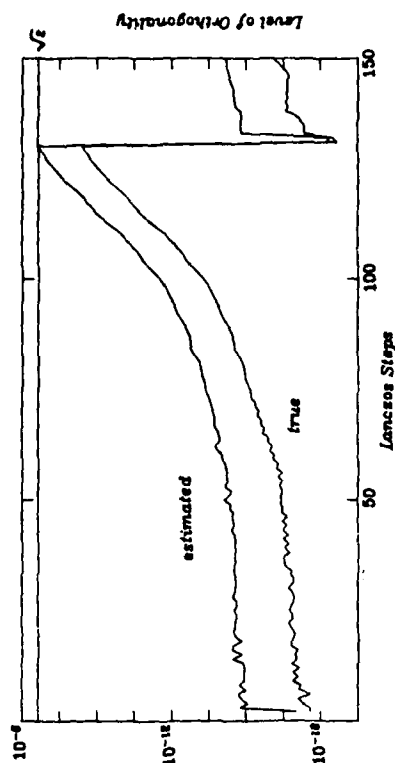


Figure 3.8.b. True and computed level of orthogonality for a matrix similar to A from Figure 3.6.a.

The level of orthogonality is different from Figure 3.6.a. 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 3.2. 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. Their mutual relation is not affected 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 easier) 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 (3.1.2) discussed above turn the formula into a useful tool for predicting the level of orthogonality. It would be even more convenient if the w_j would be so accurate that the inner products $q_i^* q_j$ would not have to be recomputed. Let us recall that by Paige's Theorem (Theorem 2.4) the vector $u_j = Q_j^* q_{j+1} = (q_1^* q_{j+1}, q_2^* q_{j+1}, \dots, q_j^* q_{j+1})^T$ tilts towards an eigenvector of T_j , when the corresponding Ritz value is about to converge against an eigenvalue of A . Let us consider now the vector $w_j = (w_{j+1}, w_{j+2}, \dots, w_{j+10})^T$ computed by (3.1.2). Earlier we expressed the view that the computation of (3.1.2) can be considered as a simulation of the level of orthogonality as it would happen on a different machine, where the random numbers chosen for ϕ_j and ψ_j 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.

How $u_j = Q_j^* q_{j+1}$ and $w_j = (w_{j+1}, w_{j+2}, \dots, w_{j+10})^T$ behave in practice can be seen from the figures in Table 3.5. For the matrix $A = 10^2 \text{diag}(1, 1/2, 1/3, \dots, 1/80)$ and the starting vector $q_1^* = (1, 1, \dots, 1)/\sqrt{80}$ we computed for the first 10 Lanczos steps the following quantities:

Step	$\cos \chi(w_j, s^{(j)})$	$\cos \chi(u_j, s^{(j)})$	$\cos \chi(u_j, w_j)$	$\ u_j\ _\infty$
2	0.9899	0.8032	0.7105	0.53e-18
3	0.9848	0.9603	0.2082	0.23e-15
4	0.2475	0.9903	0.1808	0.96e-15
5	0.4809	0.9994	0.4399	0.86e-14
6	0.9188	0.9999	0.9159	0.63e-13
7	0.9888	1.0000	0.9880	0.75e-12
8	0.9982	1.0000	0.9981	0.11e-10
9	0.9997	1.0000	0.9997	0.18e-09
10	0.9999	1.0000	0.9999	0.35e-08

Table 3.5. Eigenexpansion of the Orthogonality Components.

Here $s^{(j)}$ denotes the eigenvector corresponding to the largest Ritz value of T_j . This example was chosen since after 10 steps the largest Ritz value of T_j has already converged to an eigenvalue of A . The first three columns of Table 3.5 show the angles of w_j and u_j with $s^{(j)}$. The figures show that with increasing level of orthogonality both u_j and w_j are tilted increasingly towards $s^{(j)}$ and that upon convergence of the associated Ritz value all three vectors practically point into the same direction.

These observations are not surprising, since they are just a consequence of Paige's Theorem. Unfortunately the situation is not always as simple as in Table 3.5. As a second example we consider the matrix $A = 1000 \text{diag}(-1, -1/2, \dots, -1/30, +1/30, \dots, +1/2, 1)$ with a random starting vector. Because of the symmetry in the spectrum the Ritz values converge in this example in pairs. The first pair $(-1$ and $+1)$ converges after 15 steps. In Table 3.6 below we summarize the corresponding angles. Here we denote by S^j the plane spanned by the two eigenvectors of T_j corresponding to the largest and the smallest Ritz value.

Step	$\cos \langle u_j, S^j \rangle$	$\cos \langle u_j, S^j \rangle$	$\cos \langle u_j, w_j \rangle$	$\ u_j\ $
2	1.0000	1.0000	-0.1342	0.43e-17
3	0.9997	0.9785	0.2279	0.29e-16
4	0.3765	0.7777	0.8481	0.34e-16
5	0.6713	0.8826	0.2408	0.18e-16
6	0.3471	0.3039	0.6262	0.88e-16
7	0.7718	0.6047	0.0910	0.30e-15
8	0.9132	0.8948	-0.5388	0.15e-14
9	0.9931	0.9820	-0.6822	0.62e-14
10	0.9988	0.9985	-0.7200	0.69e-13
11	0.9997	0.9989	-0.7988	0.51e-12
12	0.9999	0.9996	-0.7488	0.46e-11
13	0.9999	0.9999	-0.6982	0.29e-10
14	1.0000	1.0000	-0.6927	0.37e-09
15	1.0000	1.0000	-0.7352	0.26e-08

Table 3.6. Projection of the Orthogonality Components.

The results of Table 3.6 can be interpreted as follows: as both extreme Ritz values converge, both u_j and w_j tilt increasingly towards the plane S^j . However the angle between them does not tend to zero as before, but they make a nearly fixed angle as they converge. The $\cos \langle u_j, w_j \rangle$ seems to settle at about -0.7 in this example. Some more runs with the same matrix and different starting vectors showed the same behavior, only the angle between u_j and w_j settled down at a different value for each run.

This behavior of u_j and w_j is consistent with Paige's Theorem. 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.

Therefore the u_j are here only used for estimating the level of orthogonality, but not for the computation of the q_{j+1}^2 .

It turns out that if the computed orthogonality components are used for the reorthogonalization in a situation as in Table 3.6, the level of orthogonality is indeed reduced to a value below the threshold level, however not to roundoff level. This raises the additional problem of how the recurrence (3.1.2) has to be restarted after a reorthogonalization has been performed. If it turns out that the new level of orthogonality has to be recomputed then nothing has been gained by using the orthogonality components directly. It might be possible to proceed analog to section 3.1 and derive some statistical estimate for the new level orthogonality, but this question was not pursued further. The topic warrants further investigation.

3.3. Choosing Reorthogonalizations.

In section 3.1 and 3.2 we saw how to compute the level of orthogonality with (3.1.2), 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 reorthogonalized.

From the analysis in Chapter 2 it follows that it is always necessary to orthogonalize q_{j+1} against some previous Lanczos vectors, if $|q_{j+1}^T q_k| > \sqrt{\epsilon}$ for some k . $\sqrt{\epsilon}$ is the optimal threshold here, since it is the largest level 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 [37, 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 [32]. 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 $q_{j+1}^T q_k$ are. There is a direct justification of this additional reorthogonalization through formula (3.1.1). By reorthogonalizing at step j we make $q_{j+1}^T q_k = O(\epsilon)$ for all $k \leq j$. Then

$$\beta_{j+1} q_{j+1}^T q_k = -\beta_{j+1} q_j^T q_k + O(\epsilon) \quad (3.3.1)$$

But if for some k , $|q_j^T q_k| \geq \sqrt{\epsilon}$ before the reorthogonalization, then also $q_j^T q_k$ must have been comparatively large, i.e. almost as big as $\sqrt{\epsilon}$, since by (2.3.7) there is a bound on the growth of the level of orthogonality. One reorthogonalization by itself therefore does not help very much to reduce

the size of $q_j^T q_k$. If however two reorthogonalizations are performed in a row, then formula (3.1.1) yields

$$\beta_{j+2} q_{j+2}^T q_k = O(\epsilon) \quad (3.3.2)$$

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

So far we always 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 3.8. Here we plotted on a logarithmic scale $q_k^T q_k$, $k = 1, \dots, 42$ for a run of the Lanczos algorithm with $A = \text{diag}(1.4, 9, \dots, 100^4)$ and $q_1^T = (1, 1, \dots, 1)/10$.

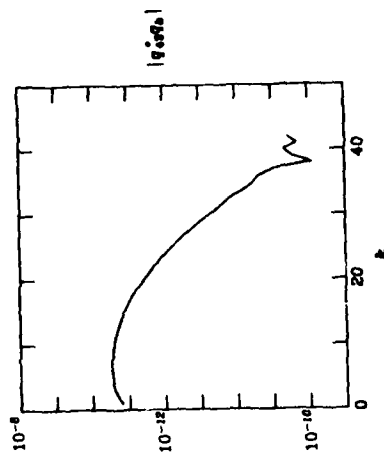


Figure 3.8. $|q_k^T q_k|$ for fixed j and k .

Figure 3.8 shows the typical pattern in the loss of orthogonality. Usually only

some neighboring q_j, q_k have grown to about the $\sqrt{\epsilon}$ 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 (3.1.2) gives a reliable prediction of the level of orthogonality the old Lanczos vectors against which q_{j+1} has to be orthogonalized, can be picked with the help of (3.1.2). It is clear that an orthogonalization only against those q_k with $|q_{j+1}^T q_k| > \sqrt{\epsilon}$ is not successful. The same argument which was used to introduce two successive orthogonalizations at consecutive Lanczos steps can be applied again. Formula (3.3.1) says that $q_{j+1}^T q_k$ depends on $q_{j+1}^T q_{k-1}$, $q_{j+1}^T q_k$, and $q_{j+1}^T q_{k+1}$. Therefore it does not help to make only $q_j^T q_k$ and $q_{j+1}^T q_k$ small, also the neighboring $q_j^T q_{k+1}$ and $q_{j+1}^T q_{k-1}$ have to be reduced in order to make the orthogonalization useful, i.e., not to allow $q_j^T q_k$ to become large again. However, in order to keep these small for some more Lanczos steps their neighbors in turn have to be small.

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

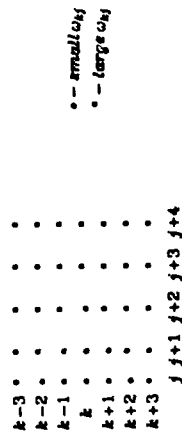


Figure 3.9. Propagation of the Loss of Orthogonality.

Figure 3.9 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 group them into "batches". One batch contains all the offending Lanczos vectors, i.e., all the q_k with $|q_{j+1}^T q_k| > \sqrt{\epsilon}$, 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 $q_j^T q_k$ has grown in l steps from the roundoff level to $\sqrt{\epsilon}$. 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 $q_{j+m}^T q_k$ stays small for another l steps ($m \leq l$). But some thought about this problem indicates that this is too much work. The terms $q_j^T q_{k+p}$ for $p \leq l$, p "far" away from k , may be already quite small (c.f. Figure 3.8) by themselves and their influence is only felt in $q_{j+p}^T q_k$, i.e., after p more steps. But then $q_{j+p}^T q_k$ may have grown already by the dynamics of formula (3.1.1) to a magnitude where the $q_j^T q_{k+p}$ because of its small size plays no role any more.

The question of how many neighboring Lanczos vectors should be used for orthogonalizing against at the same time apparently cannot be answered a priori. Therefore the following numerical experiment was carried out. At each Lanczos step the recurrence (3.1.2) was updated. If any of the $|q_{j+1}^T q_k|$ was larger than $\sqrt{\epsilon}$, then the neighboring $q_{j+1}^T q_{k-1}, q_{j+1}^T q_{k+1}, \dots$ and $q_{j+1}^T q_{k-2}, q_{j+1}^T q_{k+2}, \dots$ were checked until $q_{j+1}^T q_{k-1}$ and $q_{j+1}^T q_{k+1}$ were found with $|q_{j+1}^T q_{k-1}| \leq \eta$ and $|q_{j+1}^T q_{k+1}| \leq \eta$. Then q_{j+1} was orthogonalized against $q_{k-1}, q_k, q_{k+1}, \dots, q_{k+l-1}, q_{k+l}$. At the following Lanczos step q_{j+2} was orthogonalized against $q_{k-1}, q_k, q_{k+1}, \dots, q_{k+l-1}, q_{k+l}$. Orthogonalizations against q_{k-1} and q_{k+l} are not necessary any more at the $j+1$ st step (except for $k=1$), because the inner products $q_j^T q_{k-1}$ and $q_j^T q_{k+l}$ will deteriorate any

way due to the influence of unorthogonalized neighbors (cf. Figure 3.9). These runs were repeated for different values of η . Table 3.7 summarizes the results for two examples. For each example we list in the first column the number of orthogonalizations and in the second column the number of recalls, i.e., the number of steps at which reorthogonalizations occurred.

η	Example I		Example II	
	Orthogonalizations	Recalls	Orthogonalizations	Recalls
$\sqrt{\epsilon}$	824	28	1518	40
$10^{-1}\sqrt{\epsilon}$	520	21	1178	29
$10^{-2}\sqrt{\epsilon}$	528	17	781	22
$10^{-3}\sqrt{\epsilon}$	507	15	875	15
$10^{-4}\sqrt{\epsilon}$	478	12	817	11
$10^{-5}\sqrt{\epsilon}$	504	10	872	9
$10^{-6}\sqrt{\epsilon}$	578	10	705	8
$10^{-7}\sqrt{\epsilon}$	820	10	843	8
$10^{-8}\sqrt{\epsilon}$	758	10	925	8

Table 3.7. Influence of the Lower Bound η on the Reorthogonalizations.

Here Example I is the matrix $A = 10^4 \text{diag}(1, 1/2, 1/3, \dots, 1/1000)$ and Example II is the matrix $A = \text{diag}(100, 49, 5, 48, 5, \dots, -49, 5)$ both with $q_1 = (1, 1, \dots, 1)/10$ 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 $\eta = 10^{-4}$. This can be explained as follows: if η is decreased further the batches of Lanczos vectors become larger and more orthogonalizations are made against vectors where the inner product $q_j^T q_k$ is still quite small. If η

is increased then the batches become smaller, the effects of the reorthogonalization are wiped out already after a few steps (cf. Figure 3.9), and a new reorthogonalization is necessary.

There is however a second cost factor which we have ignored so far. For large examples it will be not possible any more 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 on the system which is used and it is therefore difficult to compare it to savings in the orthogonalizations. The numbers in Table 3.7 suggest that the number of recall operations or rewinds of the tape with the Lanczos vectors is constant as long as $\eta \leq 10^{-4}\sqrt{\epsilon}$ and then increases only slowly. Therefore the optimal choice for η regarding both cost factors lies somewhere between $10^{-4}\sqrt{\epsilon}$ and $10^{-5}\sqrt{\epsilon}$, regardless of the precise relation between both cost factors. In order to determine an η independent from the machine used, we suggest $\eta = \epsilon^{\frac{1}{4}}$. On the VAX 11/780 this choice yields $\eta \approx 0.2274 \cdot 10^{-18}$, which is slightly smaller than $10^{-4}\sqrt{\epsilon} \approx 0.3725 \cdot 10^{-18}$. This also seems to be a satisfactory choice in the sense that $\eta = \epsilon^{\frac{1}{4}}$ is "halfway" between $\sqrt{\epsilon}$ (semiorthogonality) and ϵ (orthogonality to working precision) on a logarithmic scale. The examples from Table 3.7 were run again with this η , and the following results were obtained.

	No. of Orthogonalizations	No. of Recalls
Example I	501	11
Example II	807	10

Table 3.8. Results with $\eta = \epsilon^{\frac{1}{4}}$.

The figures in Table 3.8 indicate that $\eta = \epsilon^{\frac{2}{3}}$ yields almost the minimum number of both orthogonalizations and recalls. With this choice of η we have finally determined against which previous Lanczos vectors the current Lanczos vector has to be orthogonalized, and thus completed the definition of partial reorthogonalization.

A good insight into the mechanism of PRO can be gained from the following Figures 3.10 and 3.11. 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.

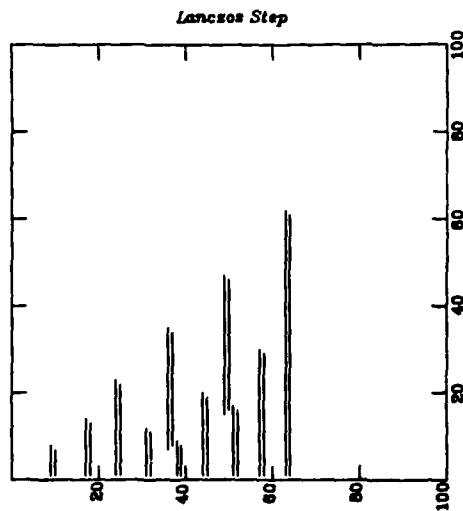


Figure 3.10. Range of Reorthogonalizations for Example 1, $\eta = \epsilon^{\frac{2}{3}}$.

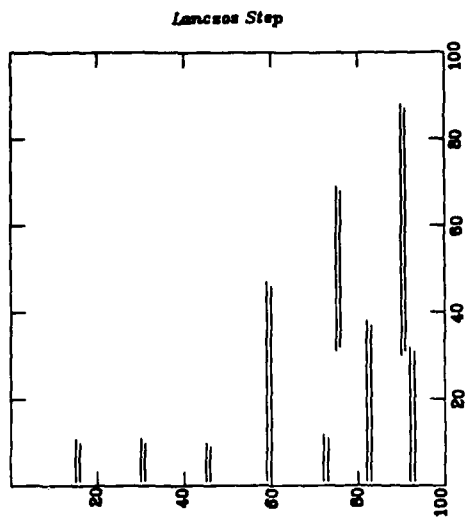


Figure 3.11. Range of Reorthogonalizations for Example 2, $\eta = \epsilon^{\frac{2}{3}}$.

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

Parameters:

r_1 = index, which indicates the first vector in batch i
 s_1 = index, which indicates the last vector in batch i

Initiatives:

second step $\leftarrow$ false

$r_1 \leftarrow 0$

$s_1 \leftarrow 0$

Subroutine PRO of the j -th Lanczos step:

- 1.) for $k=1, \dots, j$ do
 update the recurrence for $u_{j+1,k}$
- 2.) if (second step) go to 7
- 3.) for $k=1, \dots, j$ do
 if $(|u_{j+1,k}| \geq \sqrt{\epsilon})$ go to 5
- 4.) go to 9
- 5.) $r_1 \leftarrow 0$; $s_1 \leftarrow 0$
 for all k such that $|u_{j+1,k}| > \sqrt{\epsilon}$ do
 determine r_1 and s_1 , such that $|u_{j+1,k}| > \eta$,
 where $t=r_1, r_1+1, \dots, k-1, k+1, \dots, s_1-1, s_1$
- 7.) for $i=r_1, r_1+1, \dots, s_1-1, s_1, r_1+1, \dots, s_1-1, s_1, r_1, \dots$ do
 orthogonalize $\beta_{j+1} q_i$ against q_1
- 8.) if (not second step) then
 second step $\leftarrow$ true;
 $r_1 \leftarrow r_1+1$;
 $s_1 \leftarrow s_1-1$;
- 9.) return

Table 3.9. Algorithm for Partial Reorthogonalization.

3.4. 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 2.5.7)

$$q_{j+1} = q'_{j+1} - \sum_{a \in L(j)} (q'_{j+1}, q_a) q_a \quad (3.4.1)$$

Then for $q_1, i \in L(j)$:

$$q'_{j+1}, q_i = q'_{j+1}, q_i - \sum_{a \in L(j)} (q'_{j+1}, q_a) (q_a, q_i) \quad (3.4.2)$$

Since semiorthogonality is maintained, we know that $|q_a, q_i| \leq \sqrt{\epsilon}$, and from (2.3.7) we know that $|q'_{j+1}, q_a| \approx \sqrt{\epsilon}$. Hence

$$q'_{j+1}, q_i = q'_{j+1}, q_i + O(|L(j)| \epsilon \|A\|) \quad (3.4.3)$$

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 (Pariett [29, p.28]).

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 (3.3.1). Suppose we orthogonalized at the j -th step q_{j+1} against q_a . Then at the $(j+1)$ -st step by (3.3.1)

$$\beta_{j+1} q'_{j+1}, q_a = -q_{j+1}, q'_{j+1} q_a + O(\epsilon \|A\|) \quad (3.4.4)$$

Since we orthogonalized in batches, also the inner products q'_{j+1}, q_a and q'_{j+1}, q_{a-1} are at roundoff level, and we obtain (3.4.4) 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 already at the j -th step the vector

$$y_j = -\beta_{j+1} \sum_{i=1}^j (q_i/q_n) q_i \quad (3.4.5)$$

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

$$\beta_{j+1} q_{j+1} = \beta_{j+1} q_{j+1} - y_j \quad (3.4.6)$$

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.

3.5. Comparison with Selective Orthogonalization.

3.5.1. Maintenance of Semiorthonormality.

Selective orthogonalization (SO) was briefly discussed in section 2.5 as an alternative method for maintaining semiorthonormality. Practical numerical experience with SO for eigenvalue problems (Nour-Omid, Parlett, and Taylor [23]) and for the solution of linear systems (Nour-Omid [22]) shows that SO works very efficiently. Since SO maintains orthogonality with respect to the Ritz vectors rather than with respect to the Lanczos vectors, it is also of certain theoretical interest to compare both methods.

As a first example we chose the matrix A of order $n=961$ derived from an approximation to Poisson's equation on the unit square with 31×31 grid points. We solved the system $Ax = b$, where b was chosen such that $x^* = (1, 1, \dots, 1)$, both with SO and PRO. The corresponding algorithms were stopped as soon as an approximate solution was found, which reduced the residual norm by a factor of 10^{-10} . Some information of the runs with PRO and SO is given in the following table.

	PRO	SO
Orthogonalizations of q_{j+1} against	q_1 to q_m at step 40 q_1 to q_m at step 41	y_n at step 43 y_{n-1} at step 46 y_{n-2} at step 50 y_n at step 50 y_n at step 46 y_n at step 43

Table 3.10. Comparison between PRO and SO, Example 1.

This example can be considered as typical for a two dimensional problem. The amount of orthogonalizations in both cases was small and convergence

occurred already after 50 steps. Both algorithms produced the same α_i 's and β_i 's in agreement with Theorem 2.5. However the way in which this was achieved is quite different as Table 3.10 shows. Both algorithms recognize about the same time that orthogonality is going to be lost, and decide that orthogonalizations are necessary. But the only correspondence appears to be that PRO performs a reorthogonalization at about the same time when SO performs the orthogonalization against the dominant Ritz vectors.

In order to understand PRO in terms of the Ritz vectors we computed the vector $v \in \text{span}(Q_i)$ against which q_{i+1} was orthogonalized during PRO, e.g. at step 40 the vector $v = \sum_{k=1}^i (q_i^T q_k) q_k$, normalized it, and then expanded it in terms of the Ritz vectors y_k . The results for Example 1 are given in Table 3.11.

Step	Components of v in direction of					
	y_0	y_{n-1}	y_{n-2}	y_s	y_t	y_1
40	0.89	0.27	-1.1e-1	-48e-2	.56e-1	0.38
41	-0.87	-0.26	.11e-1	.48e-2	-.56e-1	-0.40

Table 3.11. Expansion of PRO Vectors in Terms of Ritz Vectors, Example 1.

The results of Table 3.11 are again a verification of Paige's theorem. PRO almost does the same as SO: a reorthogonalization against the two converging Ritz vectors. However it also reduces at the same time components in direction of the other Ritz vectors, where Ritz values converge only a couple of steps later.

In order to understand the relation between PRO and SO better, we repeated these numerical experiments with a different matrix. We chose as

second example the matrix $A = 100 \cdot \text{diag}(1.1/2, 1/3, 1/4, \dots, 1/100)$ with $q_i^T = (1, 1, 1, \dots, 1)/10$ as starting vector. The algorithm was stopped, as soon as an approximate solution to $Ax = q_1$ was found, which reduced the residual norm by a factor of $10^{-4}(\approx \sqrt{\epsilon})$. This is a very contrived and artificial example, which was chosen on purpose, because an interesting pattern of reorthogonalizations occurs already quite early. The results for this example are shown in Tables 3.12 and 3.13.

	PRO	SO
Orthogonalizations of q_{j+1} against	q_1 to q_6 at step 11 q_1 to q_{10} at step 18 q_1 to q_{24} at step 25 q_1 to q_{30} at step 30	y_n at steps 11, 19, 28 y_{n-1} at steps 14, 23 y_{n-2} at steps 18, 26 y_{n-3} at steps 19, 30 y_{n-4} at step 21 y_{n-5} at step 23 y_{n-6} at step 25 y_{n-7} at step 27 y_{n-8} at step 29

Table 3.12. Comparison between PRO and SO, Example 2.

Step	Components of v in direction of									
	y_0	y_{n-1}	y_{n-2}	y_{n-3}	y_{n-4}	y_{n-5}	y_{n-6}	y_{n-7}	y_{n-8}	y_{n-9}
11	-1.00	.14e-3	.28e-5	.28e-6	.55e-7	.19e-8	.19e-9	.49e-7		
12	1.00	.32e-3	-.12e-4	-.13e-5	.89e-6	.19e-5	.36e-6			
18	1.00	.25e-1	.84e-1	-.51e-1	.16e-1	.12e-2	.71e-5			
19	-0.94	-.50e-1	-.21e+0	.24e+0	-.11e+0	.11e-1	.15e-3			
25	-1.00	.48e-2	-.13e-2	-.11e-2	.45e-2	.11e-1	.15e-1			
28	0.98	-.95e-2	-.42e-2	.51e-2	-.27e-1	.80e-1	.15e+0			
30	0.96	-.28e+0	-.13e-1	.19e-1	.60e-2	.19e-3	.99e-3			

Table 3.13. Expansion of PRO Vectors in Terms of Ritz Vectors, Example 2.

PRO apparently cannot be easily interpreted in terms of the Ritz vectors. The figures in Table 3.13 show that the reorthogonalization mainly occurred in direction of the dominant Ritz vector, but that 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 (e.g. against y_{n-1} at step 30).

3.5.2. Comparison of Costs.

Let us go back to Tables 3.10 and 3.12 and compare the cost of PRO and SO. The longer list of orthogonalizations for SO is somewhat misleading, because orthogonalizations against recurring Ritz vectors are very inexpensive. They involve only two inner products and the recalling of the corresponding Ritz vector from secondary storage. Although SO appears to be more expensive, because of a larger number of orthogonalizations, it is actually not, which can be shown by counting the inner products involved.

Again we are faced with the question, how to relate the I/O cost to the cost of the arithmetic operations. But even without giving a precise answer to this question, we gained from a large number of examples the experience that the cost of the two algorithms for orthogonalizations is comparable. The examples above favors PRO slightly, but there are also examples, where the situation is reversed.

Also the overhead costs for SO and PRO are comparable. PRO needs two extra y -vectors for the updating of the $\omega_{j+1,0}$, which can be done in $O(j)$

operations. On the other hand SO needs one y -vector for the eigenvector of $\mathcal{E}_j$ and some extra vectors of dimension $< j$ for analyzing T_j . One can determine the interesting Ritz values and (if necessary) the corresponding eigenvectors of T_j in $O(j)$ operations [31].

Hence it appears that neither method has a clear edge over the other one. The differences between the two methods can be better understood if we look at examples where either of them performs very poorly. The following example are again contrived in order to present the extreme possible cases.

Let us first consider $A = \text{diag}(1, 2, 3, \dots, 999, 2000)$, $q_1^* = (1, 1, \dots, 1) / \sqrt{1000}$, where for both SO and PRO the algorithm was stopped after 60 steps.

	PRO	SO
Orthogonalizations of q_{j+1} against	q_1 to q_6 at step 15 q_1 to q_8 at step 27 q_1 to q_{10} at step 39 q_1 to q_{10} at step 51	y_n at steps 11, 27, 39, 53

Table 3.14. Comparison between PRO and SO, Example 3.

Again we observe that reorthogonalizations in PRO occur about the same step, when in SO an orthogonalization against the dominant eigenvector is performed. Because of the wide separation of the dominant eigenvalue $\lambda_n = 2000$ from the rest of the spectrum, these reorthogonalizations are, contrary to example 1, the only ones, which are necessary in SO, and hence can be performed very cheaply. PRO does not have this information available, performs the more expensive reorthogonalizations, and needs about four times as many orthogonalizations in order to achieve the same result.

On the other hand, the following example shows that also the opposite situation can occur.

Consider the matrix

$A = \text{diag}(100, 48.5, 47.5, \dots, -47.5, -48.5, -49.5)$ and the right hand side $b = (100, 48.5, 47.5, \dots, -47.5, -48.5, -49.5)$. Solving $Ax = b$ we encounter the special situation that the solution vector has equal components in direction of all eigenvectors, and that the eigenvalues of A are evenly distributed (with the exception of λ_0). It is therefore not surprising that the Lanczos algorithm needs n steps, when the stopping requirement is to reduce the residual norm by $\sqrt{\epsilon}$. It is too cumbersome to list all the individual orthogonalizations in this example. Here SO turns out to be more expensive than PRO in all respects. Because the algorithm terminates only for $j = n$, SO computes a large number of Ritz vectors and performs orthogonalizations against them. The situation does not improve, if we stop in SO the performance of new orthogonalizations after a certain fixed number of Ritz values has converged, and continue only orthogonalizations against already computed Ritz vectors. Because all eigenvectors of A contribute equally strongly to the solution, such a procedure only delays the convergence of some of the Ritz vectors and thus also delays the convergence of the algorithm.

This situation seems to be typical if we want to solve linear systems of equations, since we have to wait until all eigencomponents of the solution vector are well represented in $\text{span}(Q_j)$. This can mean that quite a large number of Ritz vectors is converging without necessarily improving the approximate solution. In this situation SO is forced to perform many orthogonalizations, and also many recalls of Ritz vectors, whereas PRO can handle the situation more efficiently. Besides SO is forced to compute Ritz values and vectors, which are of no direct relevance for the solution of linear systems.

If we consider on the other hand eigenvalue computations, then the original problem can be changed by shifting and inverting in such a way that the Ritz values and vectors which converge first are indeed the desired ones. In this case (compare example 3) SO fares much better, since the loss of orthogonality will be exclusively in direction of the few desired and already computed Ritz vectors. Only cheap repeated orthogonalizations in direction of these wanted Ritz vectors have to be performed, whereas PRO has to recall each time the Lanczos vectors and perform a reorthogonalization, which is as expensive as the first one. In addition to that PRO does not provide a priori any information on the progress of the convergence of the Ritz values, and would require the additional cost of computing Ritz values and vectors if applied to the eigenvalue problem.

Hence we can draw the conclusion that PRO appears to be 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.

4. NUMERICAL EXAMPLES

The Lanczos algorithm with partial reorthogonalization (LANPRO) as described in Chapter 3 was tested on several examples arising from finite element approximations to problems in structural engineering. The corresponding stiffness matrices were computed using the finite element approximation program FEAP [42, Chapter 23]. In all reported examples the algorithms were stopped when the initial residual norm was reduced by a factor of 10^{-6} .

Example 1. Here we consider a beam problem with one end encastre and one end free. A finite element approximation using 80 elements with 3 degrees of freedom per node yields the positive definite matrix H_{80} of order $n = 237$ with about six nonzero elements per row. We solved the problem

$$H_{80}x = e_{150} \quad (4.1)$$

which corresponds to applying a unit load to about the middle of the beam, both with LANPRO and with CG. This type of problem is one of the most difficult to solve with an iterative procedure. A comparison of the residual norms for both methods is given in Figure 4.1.

Figure 4.1 shows the typical behavior of the conjugate gradient algorithm, which needs 888 steps ($\approx 3.7n$ steps) to achieve the desired reduction in the residual norm, whereas LANPRO needs only 158 steps. The maintenance of semiorthogonality among the Lanczos vectors yields, as expected, a large reduction in the number of necessary steps. This reduction is of course not free - LANPRO needed about 7016 extra inner products for orthogonalizations in order to achieve this result.

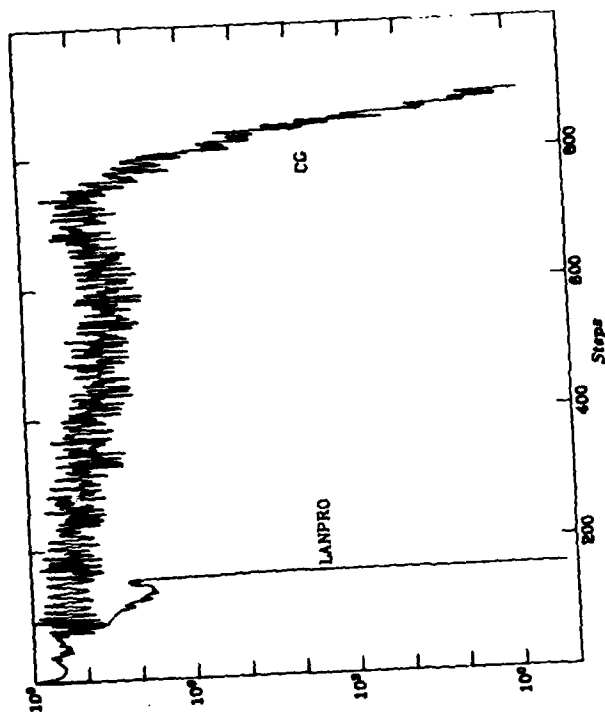


Figure 4.1. Residual Norms for Example 1.

In this particular example the matrix-vector multiplication is relatively cheap and counting the total number of operations, Lanczos turns out to be slightly more expensive. This is however quite irrelevant for the assessment of LANPRO, because one could easily find a matrix with the same eigenvalue distribution as H_{80} , such that the matrix vector product turns out to be very expensive. Then the savings of 730 matrix vector multiplications in LANPRO versus CG would make Lanczos very cost effective.

What is even more important, the maintenance of semiorthogonality guarantees termination of the Lanczos algorithm after at most n steps. For CG in finite precision there is no such guarantee and the algorithm may not terminate at all.

Another advantage of LANPRO is the availability of the Lanczos vectors for computing an initial approximation to the solution if consecutive right hand sides have to be processed. We computed an initial guess for a couple of new right hand sides according to (1.4.9), and then restarted the Lanczos algorithm. The results are shown in the table below.

Right Hand Side	Number of Steps (LANPRO)
$\mathbf{e}_{100}$	4
$\mathbf{e}_{101}$	4
$\mathbf{e}_{100} - \mathbf{e}_{99}$	5
$\mathbf{e}_{100} - \mathbf{e}_{100}$	4

Table 4.1. Consecutive Right Hand Sides for Example 1.

CG has to start for each new right hand side completely from the beginning and needs a full run (probably another 888 steps) in order to achieve the required accuracy. This has to be contrasted with the numbers for LANPRO in Table 4.1. The Lanczos algorithm obtains the solution for consecutive right hand sides almost for free.

Example 2. We consider the same beam problem as before, but use now 240 elements and obtain H_{607} , a positive definite matrix of order $n = 957$. We solve the corresponding problem as in Example 1.

$$H_{607} \mathbf{x} = \mathbf{e}_{408} \quad (4.2)$$

and obtain the following graph for the residual norms.

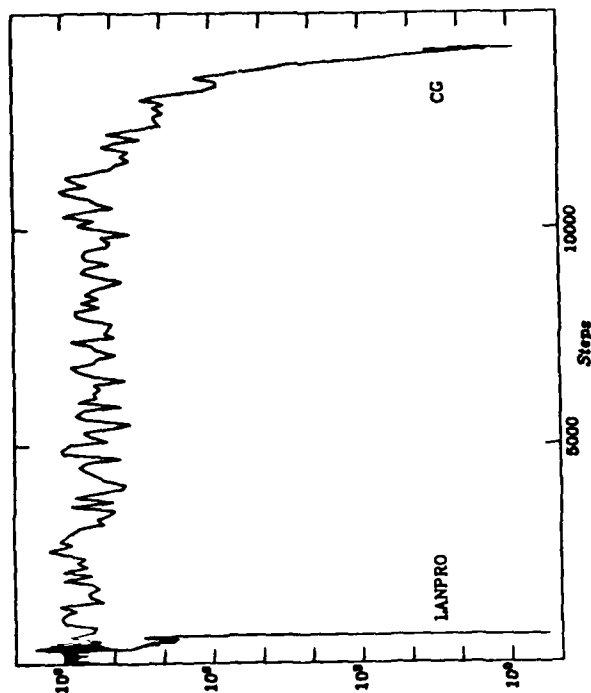


Figure 4.2. Residual Norms for Example 2.

The conjugate gradients algorithm here needs 14,169 steps, an enormous amount as compared to only 638 steps for LANPRO. Since this problem, except for its size, is the same as Example 1, all the remarks made above apply here. Also for consecutive right hand sides we obtain comparable results. The savings in LANPRO are now even more dramatic.

Right Hand Side	Number of Steps (LANPRO)
δ_{111}	12
δ_{112}	12
$\delta_{113} - \delta_{121}$	12
$\delta_{114} - \delta_{122}$	12

Table 4.2. Consecutive Right Hand Sides for Example 2.

Example 3. Here we consider

$$(H_{227} - 2000)x = (1, 1, \dots, 1)^T \quad (4.3)$$

where H_{227} is the matrix from example 1. The shift of -2000 makes the problem indefinite. Because CG in general is not applicable to indefinite problems, we compared our algorithm here with the algorithm SYMMLQ (c.f. section 1.2). The indefinite problem (4.3) causes no problem for LANPRO, and it compares very favorably in cost with SYMMLQ which is more expensive per step than conjugate gradients and needs here 2003 steps for convergence.

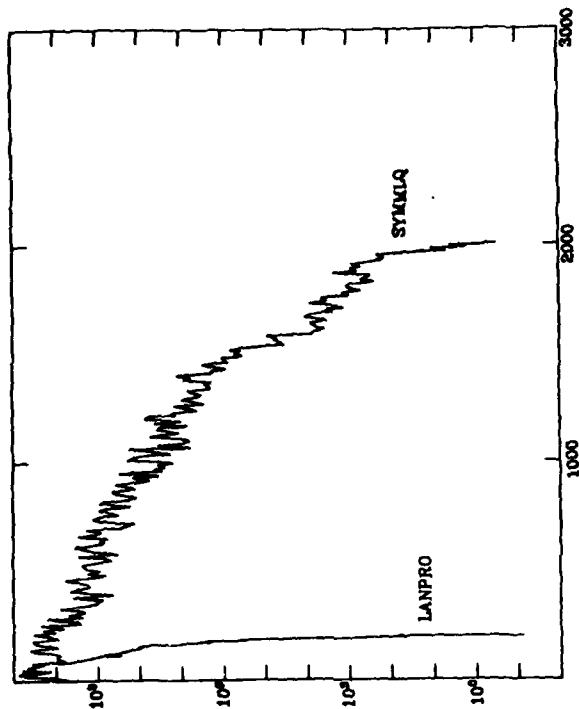


Figure 4.3. Residual Norms for Example 3.

Example 4. Here we consider a three dimensional problem arising from the finite element approximation to the building in Figure 4.4. The resulting system of linear equations is of order 488. For this example the residual norms behave quite differently (right hand side = δ_{120}) as Figure 4.5 shows.

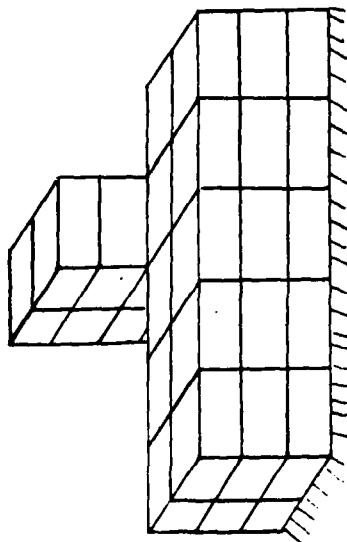


Figure 4.4. Building.

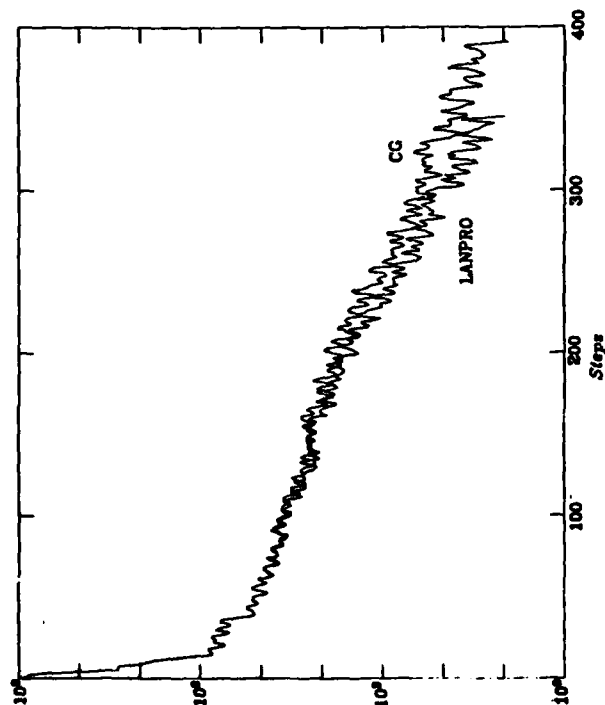


Figure 4.5. Residual Norms for Example 4.

LANPRO is here about 50 steps faster than CG, but in turn needs reorthogonalizations only twice. The steep drop of the residual norm during the first couple of steps indicates an advantage of the Lanczos algorithm as compared to direct methods. In many applications the solution vector is only required to a moderate accuracy. LANPRO may deliver such an approximate solution very cheaply after only a couple of steps. A direct method however always computes the solution vector to full accuracy, whether needed or not. Lanczos will therefore be a very attractive method when only a moderately accurate solution is required.

The comparison of simple CG with LANPRO might be criticized as unrealistic, since in practice CG will be always used in connection with preconditioning. However, as on an abstract level both CG and Lanczos are identical, both should be affected by preconditioning in the same way. If preconditioning improves the performance of CG, then it will improve Lanczos as well, and a comparison between preconditioned versions of both algorithms will yield the same qualitative insights, as a comparison between the not preconditioned versions. To illustrate this point, we run Example 4 with a simple diagonal preconditioning. CG now terminated after 243 steps. LANPRO took 192 steps, but needed now only one reorthogonalization (at step 102/103). Hence for both preconditioned algorithms the original cost is considerable reduced, and preconditioning did not improve one algorithm more than the other.

The nice feature about LANPRO becomes very clear in this example. For little extra cost we were 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 one reorthogonalization are the small price to pay for it and yet LANPRO still

compares favorably with CG in terms of cost.

Admittedly the discussion of the numerical results left out one important point: the cost of I/O operations, which may be essential if the problem is very large. There is however nothing conclusive which can be said about this cost factor at this point, since it is totally dependent on the system. All numerical tests reported here were carried out on a machine with virtual memory, and the true I/O cost are therefore hidden. If the problem under consideration is however so large that the matrix cannot be kept in core, and each matrix-vector multiplication needs additional I/O operations, then we can expect that LANPRO will compare very favorably with CG, as LANPRO minimizes the number of matrix vector multiplications. This advantage may be set off by additional I/O operations for recalling the Lanczos vectors. It seems however that for problems as considered in Examples 1 or 2, LANPRO will have a clear advantage since the difference in steps is very large.

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

- finds a solution in $\leq n$ steps,
- is economical for the treatment of several right hand sides,
- can handle definite and indefinite problems equally well,
- can take advantage of low accuracy requirements,
- compares favorably with CG when the matrix vector product is expensive.

References

- [1] E. Aitken, Tchebyscheff-Polynomials for Sets Consisting of Two Disjoint Intervals with Application for Convergence Estimates for the Conjugate Gradient Method, Report 77.08R, Dept. of Computer Science, Chalmers Univ., Göteborg, Sweden, 1978.
- [2] O. Axelsson, Solution of Linear Systems of Equations: Iterative Methods, in "Sparse Matrix Techniques Copenhagen 1978" ed. V.A. Barker, Lecture Notes in Mathematics No.572, Springer Verlag, Berlin 1977.
- [3] J.R. Bunch and B.N. Parlett, Direct Methods for Solving Symmetric Indefinite Systems of Linear Equations, SIAM J. Num. Anal., 8, 639-655, 1971.
- [4] J. Cullum and R. Willoughby, Lanczos and the Computation in Specified Intervals of the Spectrum of Large Sparse Real Symmetric Matrices, in "Sparse Matrix Proceedings" ed. I. Duff and G.W. Stewart, SIAM, Philadelphia, 1979.
- [5] J. Cullum and R. Willoughby, Computing Eigenvectors and Eigenvalues of Large Sparse Symmetric Matrices Using Lanczos Tridiagonalization, in "Numerical Analysis Proceedings, Dundee 1979" ed. G.A. Watson, Springer Verlag, Berlin, 1980.
- [6] R. Fletcher, Conjugate Gradient Methods for Indefinite Systems, in "Proceedings of the Dundee Conference on Numerical Analysis, 1975", ed. G.A. Watson, pg. 73-89, Springer Lecture Notes No. 508, Berlin 1978.
- [7] V.M. Fridman, The Method of Minimum Iterations with Minimum Errors for a System of Linear Algebraic Equations with a Symmetrical Matrix, USSR Comp. Math. and Math. Phys. 3, 382-383, 1963.

- [8] G. Golub, R. Underwood, and J.H. Wilkinson, The Lanczos Algorithm for the Symmetric $Ax = \lambda Bx$ Problem, Tech. Rep. STAN-CS-72-720, Comp. Sci. Dep., Stanford University, 1972.
- [9] J. Greer, Analyses of the Lanczos Algorithm and of the Approximation Problem in Richardson's Method, PhD Thesis, University of Illinois at Urbana-Champaign, 1981.
- [10] A. Greenbaum, Convergence Properties of the Conjugate Gradient Algorithm in Exact and Finite Precision Arithmetic, PhD Thesis, Dept. of Math., University of California, Berkeley, 1981.
- [11] S. Hammerling, A Note on Modification to the Givens Plane Rotation, J. Inst. Math. Appl. 13, 215-218, 1974.
- [12] M.R. Hestenes and E. Stiefel, Methods of Conjugate Gradients for Solving Linear Systems, J. Res. Bur. Standards 49, 409-436, 1952.
- [13] A. Householder, The Theory of Matrices in Numerical Analysis, Dover (reedition), 1975.
- [14] A. Jennings and G.M. Malik, The Solution of Sparse Linear Equations by the Conjugate Gradient Method, Int. J. Num. Meth. Eng. 12, 141-158, 1978.
- [15] S. Kanell, Estimates for some Computational Techniques in Linear Algebra, Math. Comp. 20, 389-378, 1968.
- [16] D.S. Kershaw, The Incomplete Choleski-conjugate gradient Method for the Iterative Solution of Systems of Linear Equations, J. Comp. Physics 24, 43-65, 1978.
- [17] C. Lanczos, An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators, J. Res. Nat. Bur. Standards 45, 255-282, 1950.

- [18] C. Lanczos, Solution of Systems of Linear Equations by Minimised Iterations, J. Res. Nat. Bur. Standards 49, 33-53, 1952.
- [19] D.G. Luenberger, Hyperbolic Pairs in the Method of Conjugate Gradients, SIAM J. Appl. Math. 17, 1283-1287, 1969.
- [20] T.A. Manteuffel, Shifted Incomplete Choleski Factorization, pp. 41-61 in "Sparse Matrix Proceedings" ed. I. Duff and G.W. Stewart, SIAM, Philadelphia, 1979.
- [21] J.A. Meijerink and H.A. van der Vorst, An Iterative Solution Method for Linear Systems of which the Coefficient Matrix is a symmetric M-matrix, Math. Comp. 31, 149-167, 1977.
- [22] 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.
- [23] B. Nour-Omid, B.N. Perlett, and R. Taylor, Lanczos versus Subspace Iteration for the Solution of Eigenvalue Problems, Report UCB/SESM-80/08, Dept. of Civil Engineering, University of California, Berkeley, 1980.
- [24] C. Paige, The Computation of Eigenvalues and Eigenvectors of Very Large Sparse Matrices, Ph.D. Thesis, Univ. of London, 1971.
- [25] C. Paige, Computational Variants of the Lanczos Method for the Eigenproblem, J. Inst. Math. Appl. 10, 373-381, 1972.
- [26] C. Paige, Error Analysis of the Lanczos Algorithm for Tridiagonalizing a Symmetric Matrix, J. Inst. Math. Appl. 18, 341-349, 1978.
- [27] C. Paige, Accuracy and Effectiveness of the Lanczos Algorithm for the Symmetric Eigenproblem, Lin. Alg. Appl., 34, 235-258, 1980.

- [28] C. Paige and M. Saunders, Solution of Sparse Indefinite Systems of Linear Equations, *SIAM J. Num. Anal.*, 12, 617-629, 1975.
- [29] B.N. Parlett, The Symmetric Eigenvalue Problem, Prentice Hall, Englewood Cliffs, 1980.
- [30] B.N. Parlett, A New Look at the Lanczos Algorithm for Solving Symmetric Systems of Linear Equations, *Lin. Alg. Appl.* 29, 323-346, 1980.
- [31] B.N. Parlett, private communication.
- [32] B.N. Parlett and D. Scott, The Lanczos Algorithm with Selective Orthogonalization, *Math. Comp.* 33, 217-238, 1979.
- [33] J. Reid, On the Method of Conjugate Gradients for the Solution of a Sparse Large System of Linear Equations, in "Proceedings of the Conference on Large Sparse Sets of Linear Equations" ed. J. Reid, Academic Press, London - New York, 1971.
- [34] T.J. Rivlin, Chebyshev Polynomials, Wiley, New York, 1978.
- [35] H. Rutishauser, Theory of Gradient Methods, Chapter 2 of "Refined Iterative Boundary Value Problems" by M. Engeli, Th. Ginzburg, H. Rutishauser, and E. Stiefel, Birkhauser, Basel, 1959.
- [36] Y. Saad, On the Rate of Convergence of the Lanczos and the Block Lanczos Method Methods, *SIAM J. Num. Analysis*, 1981.
- [37] D.S. Scott, Analysis of the Symmetric Lanczos Process, Ph.D. Thesis, Dept. of Math., University of California, Berkeley 1978.
- [38] J. Stoer and R. Freund, On the Solution of Large Indefinite Systems of Linear Equations by Conjugate Gradient Algorithms, Tech. Report, Inst. für Ang. Math. und Statistik, Univ. Würzburg, W. Germany, 1981.

- [39] H. Takahasi and M. Natori, Eigenvalue Problem of Large Sparse Matrices, Rep. Comp. Centre, Univ. Tokyo 4, 129-148, 1971-1972.
- [40] J.H. Wilkinson, Rounding Errors in Algebraic Processes, Prentice Hall, Englewood Cliffs, 1964.
- [41] J.H. Wilkinson, The Algebraic Eigenvalue Problem, Clarendon, Oxford, 1965.
- [42] O.C. Zienkiewicz, The Finite Element Method, 3rd edition, McGraw Hill, London, 1977.

This report was done with support from the Center for Pure and Applied Mathematics. Any conclusions or opinions expressed in this report represent solely those of the author(s) and not necessarily those of The Regents of the University of California, the Center for Pure and Applied Mathematics or the Department of Mathematics.