SIAM J. MATRIX ANAL. APPL.
Vol. 13, No. 1, pp. 121–137, January 1992

010

PREDICTING THE BEHAVIOR OF FINITE PRECISION LANCZOS AND CONJUGATE GRADIENT COMPUTATIONS*

A. GREENBAUM† AND Z. STRAKOS‡

Dedicated to Gene Golub on the occasion of his 60th birthday

Abstract. It is demonstrated that finite precision Lanczos and conjugate gradient computations for solving a symmetric positive definite linear system $Ax = b$ or computing the eigenvalues of A behave very similarly to the exact algorithms applied to any of a certain class of larger matrices. This class consists of matrices $\hat{A}$ which have many eigenvalues spread throughout tiny intervals about the eigenvalues of A. The width of these intervals is a modest multiple of the machine precision times the norm of A. This analogy appears to hold, provided only that the algorithms are not run for huge numbers of steps. Numerical examples are given to show that many of the phenomena observed in finite precision computations with A can also be observed in the exact algorithms applied to such a matrix $\hat{A}$.

Key words. conjugate gradient, Lanczos, finite precision arithmetic

AMS(MOS) subject classifications. 65F10, 65F15

1. Background and introduction.

The Lanczos algorithm for computing eigenvalues and eigenvectors of a symmetric matrix and the conjugate gradient algorithm for solving symmetric positive definite linear systems were introduced in the early 1950s by Lanczos [12] and by Hestenes and Stiefel [11], respectively. It was recognized at that time that the algorithms often failed to behave as they would in exact arithmetic due to the effect of rounding errors. Engeli, Ginsburg, Rutishauser, and Stiefel [5], for example, applied the conjugate gradient method (without a preconditioner) to the biharmonic equation and demonstrated that, for a matrix of order n, convergence did not occur until well after step n (although exact arithmetic theory guarantees that the exact solution is obtained after n steps). For this and other reasons, the algorithms did not gain widespread popularity at that time.

With the idea of preconditioning in the conjugate gradient method, interest in this algorithm was revived in the early 1970s, with several important papers appearing, including work by Reid [16] and by Concus, Golub, and O'Leary [4]. Due largely to the personal efforts of Gene Golub and those that he influenced, news of the effectiveness of the conjugate gradient method as an iterative technique spread quickly throughout the scientific computing community, and the algorithm soon became the most popular method for solving symmetric positive definite linear systems. Although the effect of rounding errors on the conjugate gradient algorithm was not well understood, it was observed numerically that (with a good preconditioner) either the method converged before rounding errors had any significant effect on the iterates, or, whatever the effect of roundoff, it was not catastrophic.

Further attempts were made to understand the effect of rounding errors on these two mathematically equivalent algorithms, and why, if they were run for enough steps

* Received by the editors January 2, 1991; accepted for publication (in revised form) July 29, 1991.

† Courant Institute of Mathematical Sciences, 251 Mercer Street, New York, New York 10012 (greenbaum@nyu.edu). This author's work was supported by the Applied Mathematical Sciences Program of the U.S. Department of Energy under contract DE-AC02-76ER03077.

‡ Czechoslovak Academy of Sciences, Institute of Computer Science, Pod vodarenskou vezi 2, 182 07 Praha 8, Czechoslovakia (na.strakos@na-net.ornl.gov). Part of this work was performed while this author was a visitor at the Courant Institute of Mathematical Sciences, New York, NY.

to be significantly affected by roundoff, the effects were not disastrous. Wozniakowski [20] considered a special version of the conjugate gradient (CG) algorithm and showed, essentially, that a finite precision implementation converged at least as rapidly as the method of steepest descent. More precisely, if the linear system to be solved is $Ax = b$, if $e^k = x - x^k$ denotes the error in the kth iterate, and if κ is the condition number of A, then the A-norm of the error at step k satisfies

$$\|e^k\|_A \leqq (1+2\varepsilon)\left(\frac{\kappa-1}{\kappa+1}\right)\|e^{k-1}\|_A + O(\varepsilon), \tag{1}$$

where ε is the unit roundoff of the machine and $O(\varepsilon)$ denotes terms involving the product of ε with the norm of various powers of A, x^k, and a constant. Wozniakowski also gave a bound on the ultimately attainable accuracy using this special version of the CG algorithm:

$$\begin{aligned} \limsup_{k\to\infty} \frac{\|e^k\|_2}{\|x^k\|_2} &\leqq \varepsilon\kappa^{3/2}C, \\ \limsup_{k\to\infty} \frac{\|b-Ax^k\|_2}{\|x^k\|_2} &\leqq \varepsilon\kappa\|A\|C. \end{aligned} \tag{2}$$

Cullum and Willoughby [3] proved a result similar to (1) for a more standard version of the CG algorithm.

The bound (1) is a large overestimate of the actual error, however. If the CG algorithm really converged as slowly as the method of steepest descent, it would seldom be used. Methods such as the Chebyshev algorithm or Richardson's method would be far superior. The bound (1) was derived by considering *individual* steps of the CG algorithm—assuming only that a particular step k is implemented accurately, it follows that the error at step k is reduced at least as much as it would be by a steepest descent step. Yet, an example due to Crowder and Wolfe [2] shows that unless one considers *all* steps of the CG algorithm, one cannot hope to establish much faster convergence than this. That is, if the initial search direction is chosen incorrectly but all other steps of the algorithm are implemented exactly, then convergence may be almost as slow as the method of steepest descent.

The following simple 3 by 3 example presented in [2] illustrates this phenomenon:

$$A = \begin{pmatrix} .1 & & \\ & 1 & \\ & & 1 \end{pmatrix}, \quad r^0 = \frac{1}{\sqrt{6}}\begin{pmatrix} 1 \\ -\sqrt{5} \\ 0 \end{pmatrix}, \quad p^0 = \frac{1}{4\sqrt{30}}\begin{pmatrix} -10\sqrt{5} \\ 14 \\ -3\sqrt{6} \end{pmatrix}.$$

If r^0 is the initial residual for a linear system with coefficient matrix A, and if, instead of taking the initial search direction p^0 to be r^0 we set p^0 as above, then if the remaining CG formulas,

$$\begin{aligned} r^k &= r^{k-1} - \frac{r^{k-1^T}p^{k-1}}{p^{k-1^T}Ap^{k-1}}Ap^{k-1} \\ p^k &= r^k - \frac{r^{k^T}Ap^{k-1}}{p^{k-1^T}Ap^{k-1}}p^{k-1}, \quad k = 1, 2, \cdots, \end{aligned}$$

are implemented exactly, then r^k will satisfy

$$r^k = \tfrac{3}{5} Q r^{k-1}, \quad k = 1, 2, \cdots, \quad \text{where } Q = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1/5 & -2\sqrt{6}/5 \\ 0 & 2\sqrt{6}/5 & -1/5 \end{pmatrix}.$$

Thus, the residual vector is rotated at each step through the angle $\arccos(-1/5)$ and reduced in size by a factor of $\frac{3}{5}$. Similarly, the A-norm of the error is reduced by a factor of $\frac{3}{5}$ at each iteration. This is somewhat faster than the steepest descent bound,

$$\frac{\kappa - 1}{\kappa + 1} = \frac{9}{11},$$

but it is slower than the Chebyshev method, which would converge at an asymptotic rate

$$\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \approx .52.$$

While most work on the CG algorithm was focusing on individual steps, the effects of rounding errors on the Lanczos algorithm were being studied from a more global point of view—considering the effects of roundoff over the entire course of the computation. Paige [13] wrote the Lanczos recurrence equations in matrix form along with the matrix of perturbations resulting from finite precision arithmetic. Using this formulation, he analyzed the loss of orthogonality among the Lanczos vectors. He showed that loss of orthogonality is only in the direction of converged Ritz vectors. From this it followed that at least one eigenpair must converge by step n. Paige later showed also that Ritz values "stabilize" only to points near eigenvalues of A [14]. The implications of these results as far as the rate of convergence of the Lanczos or CG algorithm were not so clear.

Parlett and Scott [15] used Paige's results to suggest a "fix" for the Lanczos algorithm—selective orthogonalization. This requires saving the Lanczos vectors and orthogonalizing against Ritz vectors as they converge. Further work on reorthogonalization strategies, as well as on understanding the behavior of the algorithms without reorthogonalization, was carried out by Simon [17]. He showed that until approximate orthogonality is lost, the tridiagonal matrix generated by a finite precision Lanczos computation is, indeed, the approximate projection of the matrix A onto the span of the Lanczos vectors (which may or may not be the desired Krylov space). Grcar [8] attempted a forward error analysis of the conjugate gradient algorithm. He showed that under a certain assumption, called the "projection property," the coefficients generated in a finite precision CG computation are within about ε of those that would be generated by the exact algorithm, as long as the vectors remain within about $\sqrt{\varepsilon}$ of the exact ones. Thus the initial deviation from exact arithmetic can be analyzed as if the coefficients were given rather than computed at each step.

In [10] a form of backward error analysis was developed for the Lanczos and CG algorithms. There it was shown that finite precision computations, run for no more than some number J steps, generate the same tridiagonal matrices at each step as the exact algorithms applied to a larger matrix $\bar{A}$, having possibly many more eigenvalues than A, but whose eigenvalues all lie within tiny intervals about the eigenvalues of A. A bound on the size of these intervals was derived in terms of the machine precision ε and the bound J on the number of steps. Using this analogy it was possible, in some cases, to derive more interesting bounds on the convergence rate of the CG and Lanczos algorithms.

For example, assuming that the widths of the intervals containing the eigenvalues of $\bar{A}$ are much smaller than the smallest eigenvalue of A, it follows that the condition number of $\bar{A}$ will be approximately the same as that of A. Hence any exact arithmetic error bound in terms of the condition number κ of A will hold, to a close approximation, for finite precision computations; e.g.,

$$\frac{\|e^k\|_A}{\|e^0\|_A} \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k. \tag{3}$$

This error bound was conjectured in [20], but it could not be proved with the approach used there. The bound derived in [10] on the size of these intervals was, unfortunately, a large overestimate. Thus, while bounds such as (3) were established if the machine precision ε was small enough, in many realistic problems the bound on the interval size was too large to provide useful information. A procedure was given to actually compute the matrix $\bar{A}$, however, and numerical computation of this matrix for many examples indicates that the eigenvalues of $\bar{A}$ are actually contained in much smaller intervals than the proven bound would suggest.

While it was proved in [10] that the behavior of finite precision Lanczos and CG computations is *identical* to that of the exact algorithms applied to a *particular* matrix $\bar{A}$, in this paper we demonstrate numerically that it is also very *similar* to that of the exact algorithms applied to *any* matrix, say, $\hat{A}$, which has many eigenvalues spread throughout tiny intervals about the eigenvalues of A. Thus, the qualitative behavior of a finite precision computation can be understood by understanding the behavior of the exact algorithms applied to such matrices $\hat{A}$. The size of the intervals is a modest multiple of the machine precision. It is not clear if this similarity is maintained if the algorithms are run for huge numbers of steps (say, $\gg 10^5$), but for more realistic computations, the similarity is demonstrated. For test problems, we consider a class of matrices introduced in [18]. There the behavior of the finite precision computations was compared with exact arithmetic theory and shown to give surprising results. In this paper we show why this behavior is to be expected. This similarity can also be used to explain the differences observed in [19] between the actual behavior of incomplete Cholesky and modified incomplete Cholesky preconditioners and that predicted by exact arithmetic theory.

Finite precision CG computations for solving an n by n symmetric positive definite linear system $Ax = b$ sometimes fail to converge after n steps, especially when n is small. In such cases, it is demonstrated that exact CG applied to the corresponding large linear system $\hat{A}\hat{x} = \hat{b}$ also requires more than n iterations to converge. More commonly, finite precision CG computations converge in far fewer than n steps, and the same holds for the exact CG algorithm applied to any matrix $\hat{A}$ whose eigenvalues are clustered in tiny intervals about the eigenvalues of A. Frequently, finite precision CG computations go through several steps at which there is only a modest reduction in the error and then at the next step there is a very sharp decrease in the error. This same behavior is observed in the exact CG algorithm applied to matrices $\hat{A}$ whose eigenvalues are distributed in n tight clusters about the eigenvalues of A.

Related to this phenomenon of slow convergence followed by a sudden drop in the error, is the phenomenon of multiple "copies" of eigenvalues appearing in finite precision Lanczos computations. Finite precision Lanczos computations frequently generate several close approximations to some of the eigenvalues of A before finding any close approximations to some of the other eigenvalues. Analogously, depending on how the clusters of the larger matrix $\hat{A}$ are distributed, the exact Lanczos algorithm applied to $\hat{A}$ may find several eigenvalues within some of the clusters before finding any in some of the other clusters. It is demonstrated that the rate of occurrence of multiple "copies" of eigenvalues

in finite precision Lanczos computations with matrix A is very similar to the rate at which the exact Lanczos algorithm applied to $\hat{A}$ finds different eigenvalues within the same cluster.

Sections 2–4 present numerical examples to demonstrate these phenomena. Implications of this analogy are discussed in § 5.

2. Description of numerical experiments. The matrices considered in this paper were introduced in [18] and have eigenvalues of the form

$$\lambda_i = \lambda_1 + \frac{i-1}{n-1}(\lambda_n - \lambda_1)\rho^{n-i}, \quad i = 2, \cdots, n, \quad \rho \in (0,1), \tag{4}$$

where n, λ_1, and $\kappa = \lambda_n/\lambda_1$ are fixed. For most of our experiments we have taken $n = 24$, $\lambda_1 = .1$, $\kappa = 1000$, and $\rho = .4, .6, .8, .9, 1.0$. These eigenvalues are plotted for each value of ρ in Fig. 1. Eigenvalues that are too close to be distinguished on the horizontal axis have been plotted vertically. For the smaller ρ-values, the eigenvalues are very tightly clustered at the lower end of the spectrum. The minimal difference, $\lambda_2 - \lambda_1$, corresponding to $\rho = .4, .6, .8, .9$, and 1.0 is $7.6e-9$, $5.7e-5$, .032, .43, and 4.3, respectively.

The algorithm used in these experiments for solving a symmetric positive definite linear system $Ax = b$ and computing the eigensystem of A is as follows [11], [12]:

Given an initial guess x^0, compute $r^0 = b - Ax^0$, and set $p^0 = r^0$.
For $k = 1, 2, \cdots$
Compute $\alpha_{k-1} = \dfrac{r^{k-1^T} r^{k-1}}{p^{k-1^T} A p^{k-1}}$.
Set $T_{k,k} = \dfrac{1}{\alpha_{k-1}} + \dfrac{\beta_{k-1}}{\alpha_{k-2}}$.
Take $x^k = x^{k-1} + \alpha_{k-1} p^{k-1}$.
Compute $r^k = r^{k-1} - \alpha_{k-1} A p^{k-1}$.
Compute $\beta_k = \dfrac{r^{k^T} r^k}{r^{k-1^T} r^{k-1}}$.
Set $T_{k,k+1} = T_{k+1,k} = \dfrac{\sqrt{\beta_k}}{\alpha_{k-1}}$.
Take $p^k = r^k + \beta_k p^{k-1}$.

The tridiagonal matrix T generated at step k will be denoted $T^{(k)}$, and its eigenvalues are taken as approximate eigenvalues of A (Ritz values). The eigenvectors of A can also be approximated if the previous residual vectors, $r^0, \cdots, r^{k-1}$, have been saved, but we will not discuss the computation of eigenvectors here. When solving linear systems, we will refer to this algorithm as the conjugate gradient method, or CG, while when using it to compute eigenvalues we will refer to it as the Lanczos algorithm. The equivalence of this method to the usual Lanczos process [12] in exact arithmetic is well known, and arguments in [3], [10] establish similar behavior in finite precision arithmetic as well. Numerical experiments with other variants of this algorithm have yielded similar results, as described in [18].

The above algorithm was applied to matrices A of the form

$$A = U \Lambda U^T,$$

where U is a random orthogonal matrix and $\Lambda = \operatorname{diag}(\lambda_1, \cdots, \lambda_n)$ is defined in (4). In all cases, a random right-hand side vector and a zero initial guess were used. Experiments were carried out on a Sun Sparcstation using double precision arithmetic (about 16

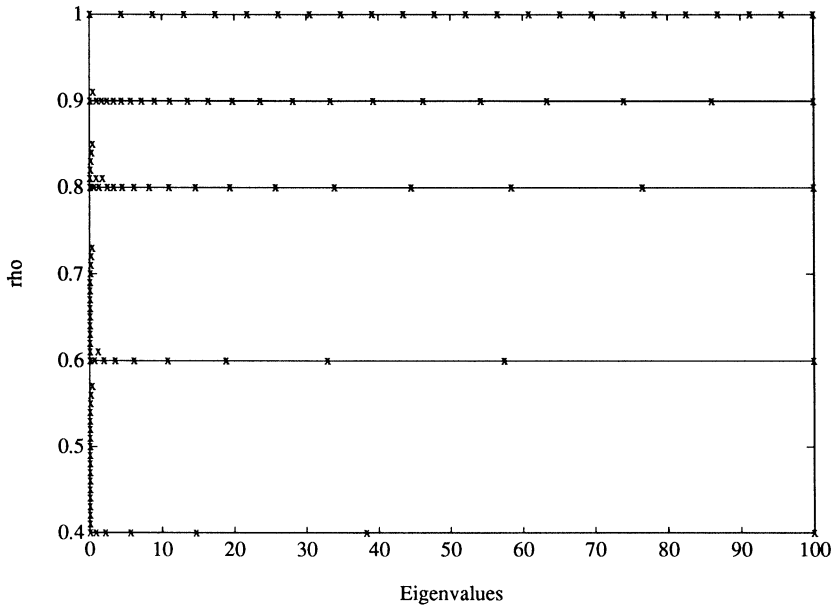

FIG. 1. *Eigenvalue distributions for different ρ-values.*

decimal digits). Most of the experiments were performed using *MATLAB*, making the algorithms relatively simple to implement.

We first compare finite precision computations for solving $Ax = b$ or computing the eigenvalues of A to the exact arithmetic algorithms applied to the same problem. "Exact arithmetic" was simulated by saving all residual vectors and using full reorthogonalization at each step. That is, the formula for r^k becomes

$r^k = r^{k-1} - \alpha_{k-1} A p^{k-1}$
For *kount* = 1, 2,
  For $j = 1, k-1$,

$$r^k = r^k - \frac{r^{k^T} r^j}{r^{j^T} r^j} r^j$$

  Endfor
Endfor

It is shown in [14] that the iterates generated using this modified algorithm do, indeed, resemble those that would be generated by the exact algorithm applied to a slightly different matrix (of the same order) with a slightly different initial vector. Until the size of the vector r^k approaches $\varepsilon \|A\| \max_{j=1,\ldots,k} \|x^j\|$, where ε is the machine precision, the recursively updated r^k is very close to the true residual, $b - Ax^k$ [10].

We also compare the finite precision computations involving the matrix A to "exact arithmetic" (full reorthogonalization) computations involving a larger matrix $\hat{A}$. The matrix $\hat{A}$ was taken to have a total of $11n$ eigenvalues, with eleven eigenvalues uniformly distributed throughout each of n tiny intervals about the eigenvalues of A. Several of the experiments were also performed with a matrix $\hat{A}$ having 21 eigenvalues evenly distributed in each of these same intervals, and the results were indistinguishable when presented in plots such as Figs. 1–9. Most of the experiments were performed with intervals of width 10^{-12}, or approximately

$$50\varepsilon \|A\|,$$

where $\varepsilon = 2^{-52}$ is the unit roundoff of the machine. Some experiments were performed with different size intervals to see the effect of the interval width on the behavior of the algorithms. The finite precision computation for $Ax = b$, with initial guess x^0, or initial residual r^0, was compared to the exact arithmetic computation for $\hat{A}\hat{x} = \hat{b}$, with initial guess $\hat{x}^0$ and initial residual $\hat{r}^0$, where

$$(5) \qquad \hat{A} = \operatorname{diag}(\lambda_{1,1}, \cdots, \lambda_{1,m}, \lambda_{2,1}, \cdots, \lambda_{2,m}, \cdots, \lambda_{n,1}, \cdots, \lambda_{n,m}),$$

$$\lambda_{i,j} = \lambda_i + \frac{j - \frac{m+1}{2}}{m-1} \cdot \delta, \quad j = 1, \cdots, m, \quad m = 11, \quad \delta = 10^{-12} \approx 50\varepsilon \|A\|,$$

$$\hat{r}^0 = \hat{b}, \quad \hat{b}_{i,1} = \hat{b}_{i,2} = \cdots = \hat{b}_{i,m}, \quad \text{and} \quad \sum_{j=1}^{m} (\hat{b}_{i,j})^2 = (U^T b)_i^2, \quad i = 1, \cdots, n,$$

where $\lambda_1, \cdots, \lambda_n$ are the eigenvalues of A and U is the orthonormal matrix of eigenvectors of A.

3. Results of CG computations. Here we show the remarkable similarity between a finite precision CG computation to solve $Ax = b$, with initial residual r^0, and the exact CG algorithm applied to the larger linear system $\hat{A}\hat{x} = \hat{b}$, with initial residual $\hat{r}^0$.

Figure 2(a) shows the convergence of finite precision CG computations applied to the linear system $Ax = b$, for the five different ρ-values listed in the previous section. The right-hand side vector b was taken to have random components, uniformly distributed between 0 and 1, and the initial guess x^0 was taken to be zero. The A-norm of the error at each iteration divided by the A-norm of the initial error

$$\frac{\langle x - x^k, A(x - x^k)\rangle^{1/2}}{\langle x - x^0, A(x - x^0)\rangle^{1/2}}$$

is plotted. (The "exact" solution x was computed as $U\Lambda^{-1}U^T b$.) Note that although exact arithmetic theory ensures that the correct solution is obtained after $n = 24$ steps, the finite precision computation requires significantly more than n steps for some of the ρ-values. For certain values of ρ the computation seems to be considerably more affected by rounding errors than for other values. Convergence slows as ρ goes from .4 to .6 to .8, but then improves as ρ reaches .9 and 1.

For comparison, Fig. 2(b) shows the convergence of the exact CG algorithm applied to the same linear system, with the same initial guess. In contrast to the finite precision computation, there is little difference between the results for $\rho = .8, .9$, and 1.0, with the slowest exact arithmetic convergence rate occurring for $\rho = .9$.

The behavior of the finite precision computations much more closely resembles that of the exact CG algorithm applied to the linear system $\hat{A}\hat{x} = \hat{b}$ (defined in (5)), shown in Fig. 2(c). Here the $\hat{A}$-norm of the error at each iteration divided by the $\hat{A}$-norm of the initial error is plotted. As with the finite precision CG computations, convergence slows as ρ goes from .4 to .6 to .8, but then improves for $\rho = .9$ and 1. The qualitative convergence behavior in Fig. 2(c) is also similar to that of the finite precision CG computations, in that, for certain ρ values, both go through stages at which little improvement is made for several steps and then a sharp drop in the error is seen at a subsequent step.

To see the effect of the interval size on the convergence rate of the exact CG algorithm applied to a matrix $\hat{A}$ with eigenvalues clustered in these intervals, we tried several different interval sizes for the case $\rho = .6$. That is, we considered matrices $\hat{A}$ whose eigenvalues were clustered in intervals of width $\delta = 10^{-13}, 10^{-12}, 10^{-10}$, and 10^{-6} about the eigenvalues

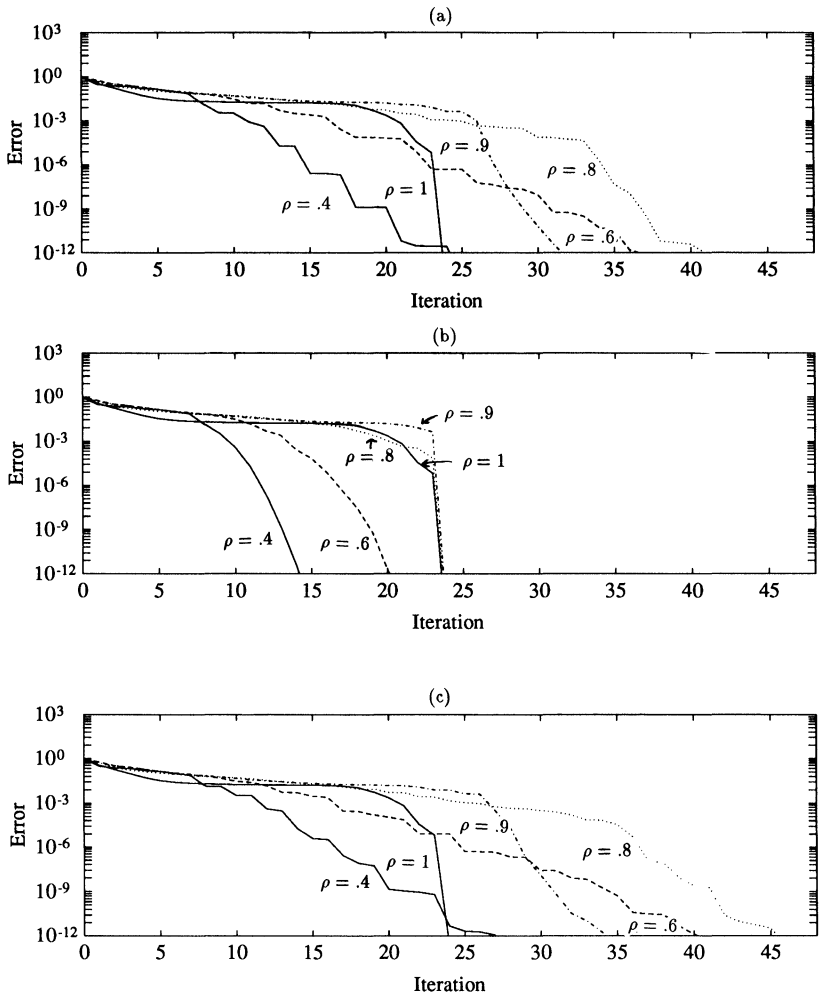

FIG. 2. (a) *Finite precision* CG *for* $Ax = b$. (b) *Exact* CG *for* $Ax = b$. (c) *Exact* CG *for* $\hat{A}\hat{x} = \hat{b}$.

of A. As before, each matrix $\hat{A}$ was taken to have eleven eigenvalues in each interval, uniformly distributed, and the initial residuals $\hat{r}^0$ were set according to (5). Figure 3 shows the convergence of the exact CG algorithm applied to the different problems $\hat{A}\hat{x} = \hat{b}$, along with that of the finite precision computation for $Ax = b$. While the exact computation with the matrix of interval width 10^{-13} most closely resembles the finite precision computation, similar qualitative behavior is reflected in all of these computations, except perhaps the one with interval width 10^{-6}, which is considerably slower to converge. Thus, it appears that a precise estimate of this interval width is not even necessary to predict the qualitative behavior of finite precision CG computations. They resemble exact CG computations for any matrix $\hat{A}$ with eigenvalues spread throughout small intervals about the eigenvalues of A, and the interval size can be anywhere within a rather wide range. The remainder of the CG comparisons will use 10^{-12} as the interval width for the eigenvalues of $\hat{A}$.

While most of our experiments have been performed with very small matrices ($n = 24$), similar phenomena can be observed with larger matrices, for which the CG

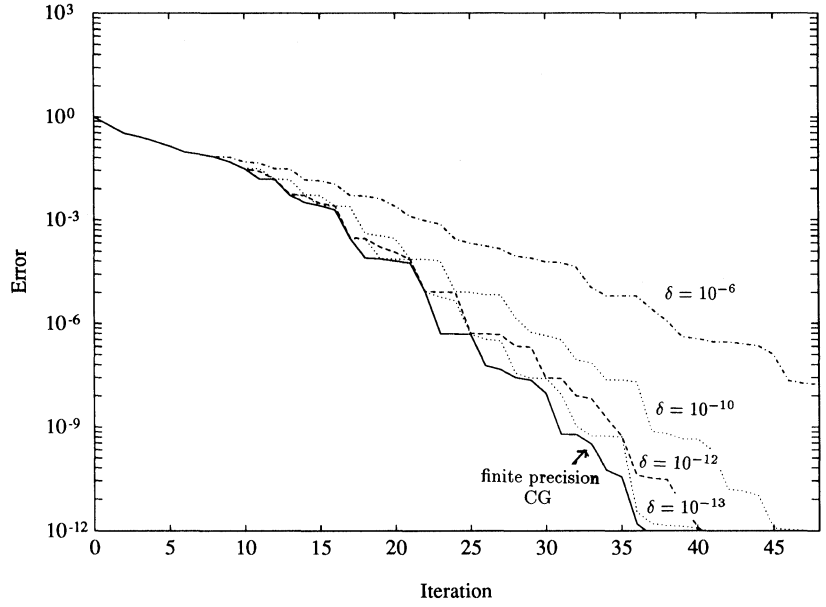

FIG. 3. *Exact* CG *for different interval widths and finite precision* CG *for* $Ax = b$ ($\rho = .6$).

algorithm is more often used. For large matrices, our comparisons with exact arithmetic computations for $\hat{A}$ become quite time- and storage-consuming, however, since $\hat{A}$ is eleven times as large as A and it is necessary to save all residual vectors and reorthogonalize at every step. Still, we have performed one experiment with a matrix A of order 100. The eigenvalues of A are still defined by formula (4), with $n = 100$, $\lambda_1 = .1$, $\kappa = 1000$, and we took $\rho = .8$. Figure 4 shows the convergence of finite precision CG for solving $Ax =$

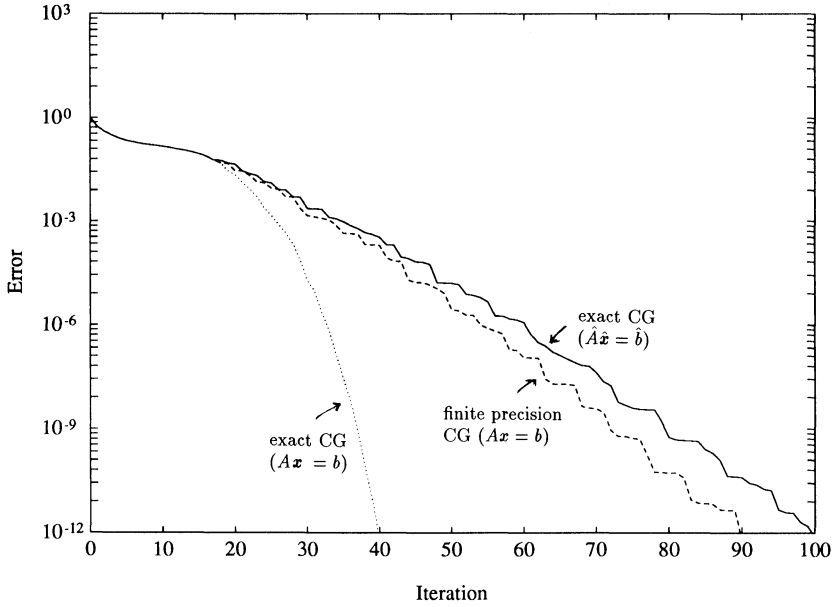

FIG. 4. *Exact and finite precision* CG ($n = 100$, $\rho = .8$).

b, exact CG for solving $Ax = b$, and exact CG for solving the larger problem $\hat{A}\hat{x} = \hat{b}$. Note the close resemblance between the first and last of these curves and the significant differences from the exact CG computation for $Ax = b$. Although this type of behavior is frequently seen in practice, it may not be realized that rounding errors are significantly affecting the convergence rate, since there is no exact arithmetic computation with which to compare.

4. Results of Lanczos computations. Here we show the similarity between the eigenvalue approximations generated at each step of a finite precision Lanczos computation with matrix A and initial vector r^0 and those generated at each step of an exact Lanczos computation with the larger matrix $\hat{A}$ and initial vector $\hat{r}^0$, defined in (5).

Figures 5(a), 6(a), and 7(a) show the eigenvalue approximations generated by a finite precision Lanczos computation with matrix A, for the cases $\rho = .6, .8, 1.0$. Similar results were observed with the other ρ-values. In order to distinguish clustered eigenvalues, we have plotted on the vertical axis not the actual eigenvalues, but the index of each eigenvalue of A, from 1 to n. An approximate eigenvalue that lies, say, $1/r$ of the way between eigenvalue i and eigenvalue $i + 1$ of A, will be plotted on the graph at y-value $i + \frac{1}{r}$. Eigenvalue approximations that are too close to be distinguished on the vertical axis have been plotted horizontally. For clarity, we have plotted the eigenvalue approximations only at every fourth step. In most cases, we see multiple copies of the larger eigenvalues appearing before any close approximations to the smaller, clustered eigenvalues appear. It should be remembered, however, that for the smaller ρ-values, these small eigenvalues are very tightly clustered, and there may be a close approximation to the cluster, even though the individual eigenvalues have not been identified. Only in the case $\rho = 1$ does the finite precision computation find all n eigenvalues by step n.

For comparison, Figs. 5(b), 6(b), and 7(b) show the eigenvalue approximations generated every fourth step of the exact Lanczos algorithm applied to the same matrices, with the same initial vectors. Here no "multiple copies" are observed, and all of the eigenvalues are identified after n steps.

The eigenvalue approximations generated by the finite precision computations much more closely resemble those generated by the exact Lanczos algorithm applied to the matrix $\hat{A}$, with initial vector $\hat{r}^0$, shown in Figs. 5(c), 6(c), and 7(c). In these figures we again see multiple close approximations to the larger eigenvalues appearing before step n, except in the case $\rho = 1$. The rate of appearance of these multiple copies also appears to be similar to that in the finite precision computations with matrix A.

We point out that for $\rho = 1.0$, the effect of roundoff on the Lanczos process is minimal (cf. Figs. 7(a), (b); also Figs. 2(a), (b)). After n iterations the process is "restarted" and it computes all eigenvalues twice in $2n$ iterations. For $\rho < 1.0$, the "restarting" is more frequent and multiple copies of large eigenvalues are computed simultaneously with single approximations to small eigenvalues. It can be observed that if a finite precision Lanczos computation generates a close approximation to each eigenvalue of A at some step, then the error in the corresponding finite precision CG computation drops dramatically at that step. See Figs. 6(a) and 2(a) ($\rho = .8$), between 32 and 36 iteration steps.

Again, to see the effect of the interval width on the eigenvalue approximations generated by an exact Lanczos computation for a matrix $\hat{A}$ with eigenvalues clustered in these intervals, we tried several different interval sizes, for the case $\rho = .6$. That is, we considered matrices $\hat{A}$ whose eigenvalues were clustered in intervals of width $\delta = 10^{-13}$, 10^{-10}, and 10^{-6} about the eigenvalues of A. As before, each matrix $\hat{A}$ was taken to have eleven eigenvalues in each interval, uniformly distributed, and the initial residuals $\hat{r}^0$ were set according to (5). Figures 8(a–c) show the eigenvalue approximations generated

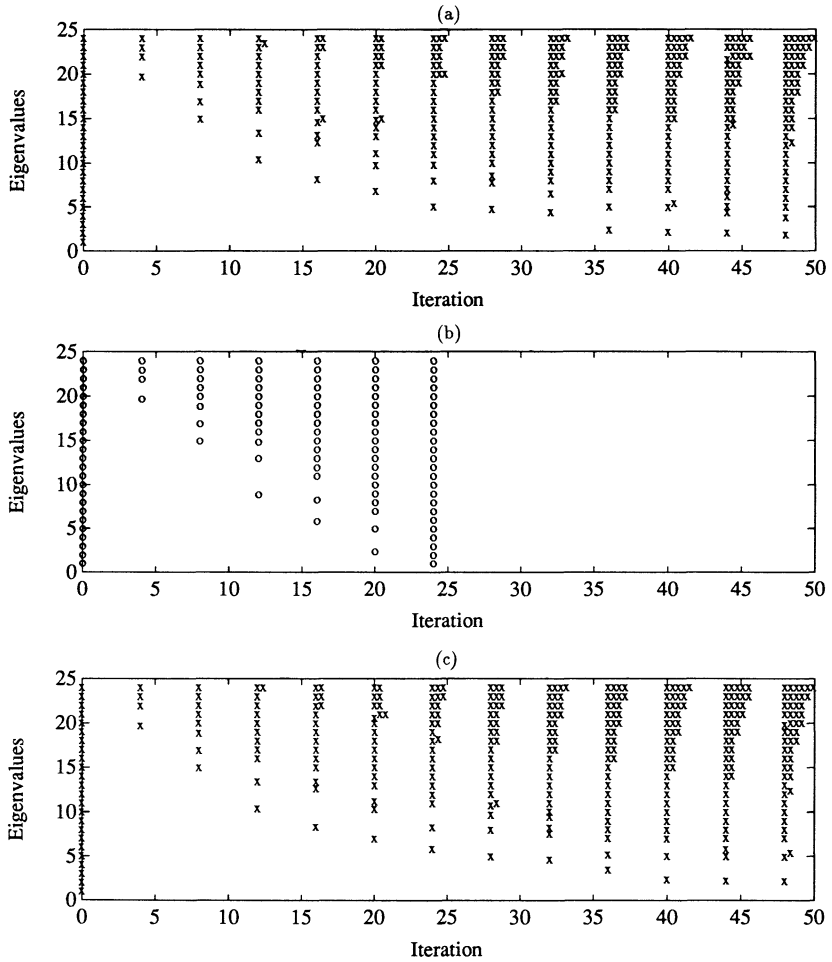

FIG. 5. (a) *Finite precision Lanczos on A* ($\rho = .6$). (b) *Exact Lanczos on A* ($\rho = .6$). (c) *Exact Lanczos on* $\hat{A}$ ($\rho = .6$).

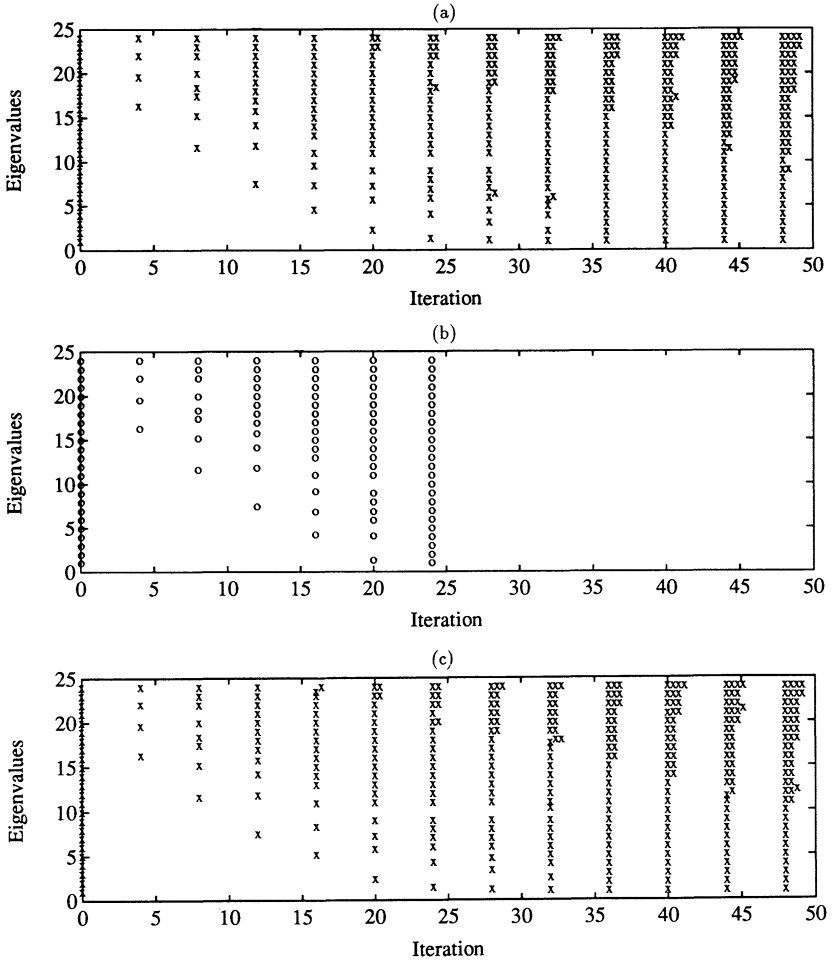

FIG. 6. (a) *Finite precision Lanczos on A* ($\rho = .8$). (b) *Exact Lanczos on A* ($\rho = .8$). (c) *Exact Lanczos on* $\hat{A}$ ($\rho = .8$).

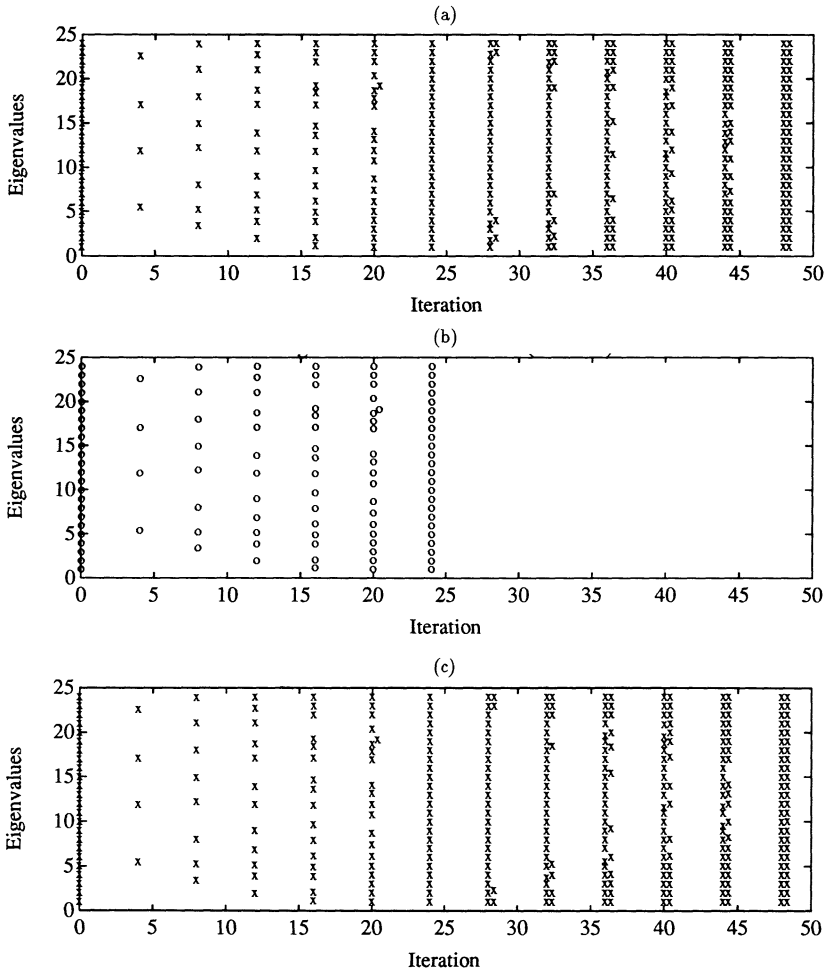

FIG. 7. (a) *Finite precision Lanczos on A* ($\rho = 1$). (b) *Exact Lanczos on A* ($\rho = 1$). (c) *Exact Lanczos on* $\hat{A}$ ($\rho = 1$).

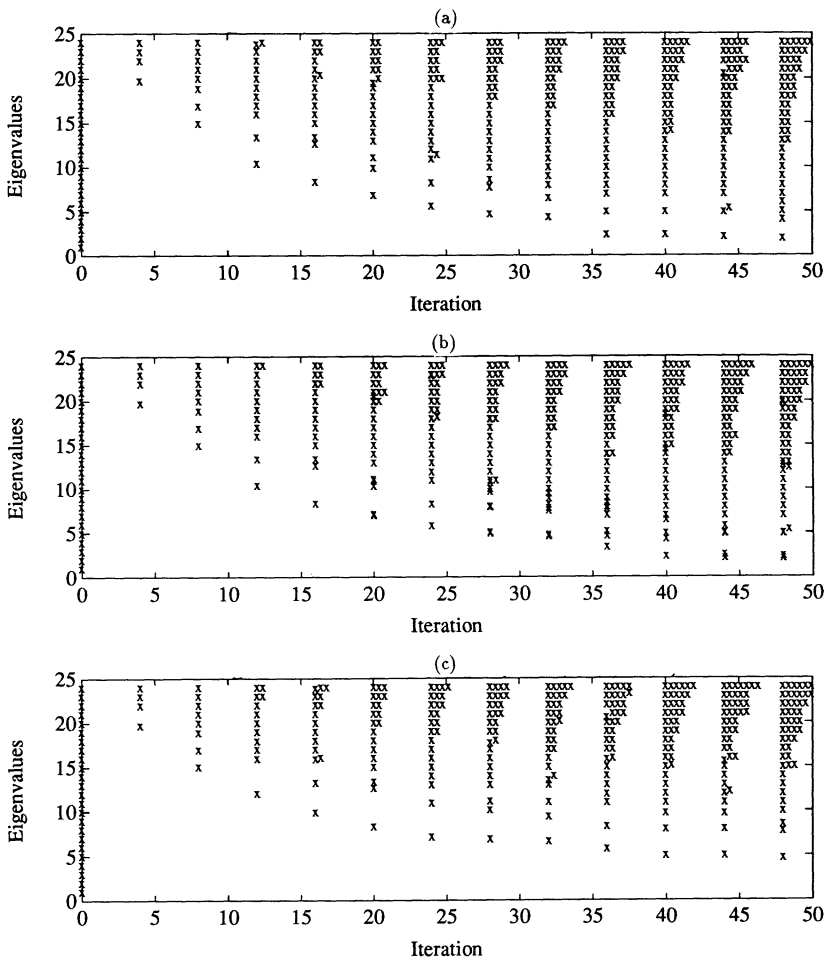

FIG. 8. (a) *Exact Lanczos on* $\hat{A}$ ($\rho = .6$, $\delta = 1.d - 13$). (b) *Exact Lanczos on* $\hat{A}$ ($\rho = .6$, $\delta = 1.d - 10$). (c) *Exact Lanczos on* $\hat{A}$ ($\rho = .6$, $\delta = 1.d - 6$).

at every fourth step by the exact Lanczos algorithm applied to each of these matrices $\hat{A}$. Note the similarity between each of these figures (as well as Fig. 5(c) with $\delta = 10^{-12}$) and Fig. 5(a), showing the eigenvalue approximations generated by a finite precision Lanczos computation for the matrix A. Figure 8(a) ($\delta = 10^{-13}$) shows the closest resemblance to the finite precision computation, while Fig. 8(c) ($\delta = 10^{-6}$) has somewhat more copies of the larger eigenvalues and somewhat fewer approximations to the smaller ones.

Finally, in Figs. 9(a–c), we have plotted results from a larger problem, with $n = 100$, $\rho = .8$. Again, note the similarities between the eigenvalue approximations generated by the finite precision Lanczos computation for the matrix A (Fig. 9(a)) and those generated by the exact arithmetic Lanczos computation for the matrix $\hat{A}$ (Fig. 9(c)). Unlike the exact Lanczos computation for A (Fig. 9(b)), these procedures both generate multiple close approximations to some of the larger eigenvalues before finding any close approximations to some of the smaller ones. The rate at which these multiple approximations appear is also similar in Figs. 9(a) and 9(c).

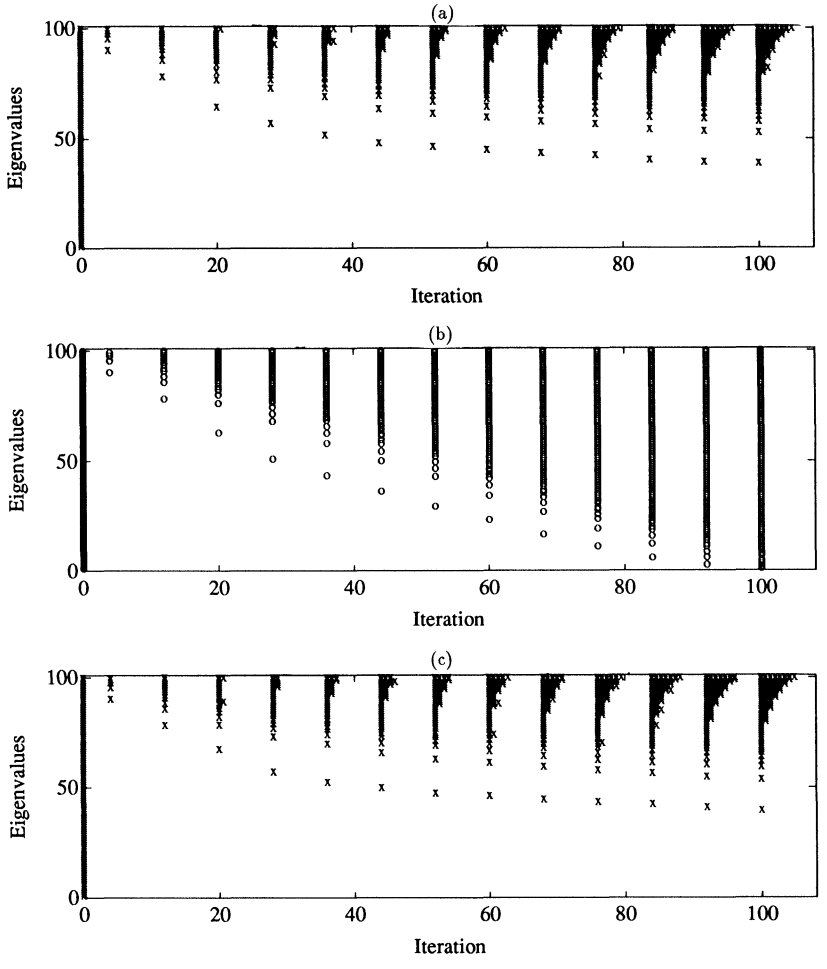

FIG. 9. (a) *Finite precision Lanczos on A* ($\rho = .8$). (b) *Exact Lanczos on A* ($\rho = .8$). (c) *Exact Lanczos on* $\hat{A}$ ($\rho = .8$).

5. Further discussion and open questions. We have demonstrated numerically that the behavior of finite precision Lanczos and CG computations with a matrix A closely resembles that of the exact algorithms applied to matrices $\hat{A}$ that have many eigenvalues spread throughout tiny intervals about the eigenvalues of A. In [10] it was proved that the eigenvalue approximations generated by a finite precision Lanczos computation are *identical* to those generated by the exact algorithm applied to a certain larger matrix $\bar{A}$, and that the A-norm of the error in the equivalent finite precision CG computation is reduced at approximately the same rate as the $\bar{A}$-norm of the error in the exact algorithm. Eigenvalues of the matrix $\bar{A}$ lie in tiny intervals about the eigenvalues of A (provided that the finite precision computation is not run for too many steps), but $\bar{A}$ might have many or only a few eigenvalues in some of these intervals.

Using these analogies, the problem of estimating and bounding the convergence rates of these algorithms in finite precision arithmetic reduces to a problem of estimating or bounding the convergence rates of the exact algorithms applied to certain classes of matrices. If an error bound can be established for the exact algorithm applied to *every* matrix whose eigenvalues lie within these intervals, then it will hold (at least to a close approximation) for the finite precision computation. If an error estimate is good for the exact algorithms applied to every matrix with many eigenvalues spread throughout these intervals, then it will also be a good estimate for the finite precision computation. We will not derive such bounds and estimates here, but it is not difficult to see how they might be derived (as, for example, in (3)), and some examples are given in [10].

A question that is frequently asked is whether a finite precision Lanczos computation eventually finds all eigenvalues of a matrix, or, at least, all well-separated eigenvalues and at least one close approximation to multiple or tightly clustered eigenvalues. Using the analogy developed in [10] between finite precision Lanczos computations run for no more than J steps and the exact algorithm applied to a matrix whose eigenvalues lie within intervals of width, say, δ_J, about the eigenvalues of A, this question can be translated as follows: Is there a J such that the exact Lanczos algorithm applied to every matrix $\tilde{A}$ whose eigenvalues lie within intervals of width δ_J about the eigenvalues of A—with initial vector $\tilde{r}^0$ satisfying

$$\sum_l (\tilde{r}^0, \tilde{u}^{i,l})^2 \approx (r^0, u^i)^2, \qquad i = 1, \cdots, n, \tag{6}$$

where $u^1, \cdots, u^n$ are the eigenvectors of A and $\tilde{u}^{i,l}$, $l = 1, \cdots$, are the eigenvectors of $\tilde{A}$ corresponding to the eigenvalues clustered about λ_i, $i = 1, \cdots, n$—finds at least one eigenvalue from each cluster within J steps? We know of no simple and general sufficient conditions for the existence of such a number J, so this question remains open.

Of course, if the interval widths δ_J could be bounded (with a suitably small bound) independent of J, then it would follow that a finite precision Lanczos computation would eventually find every eigenvalue of A whose eigenvector contained a nonnegligible component in the initial vector. (That is, it would find at least one close approximation to every well-separated eigenvalue and every eigenvalue cluster with a nonnegligible component in the initial vector.) For (6) implies, in this case, that the interval about each eigenvalue has a nonzero weight. From Favard's theorem [6] it would follow that the characteristic polynomials of the tridiagonal matrices generated by a finite precision Lanczos computation were the orthogonal polynomials for a certain measure whose support lies in the union of intervals $\cup_{i=1}^{n} [\lambda_i - \delta, \lambda_i + \delta]$, where δ is the bound on the interval size. But the roots of the orthogonal polynomials are known to converge to all weighted points (see, for example, [1]), so they would converge to at least one point in

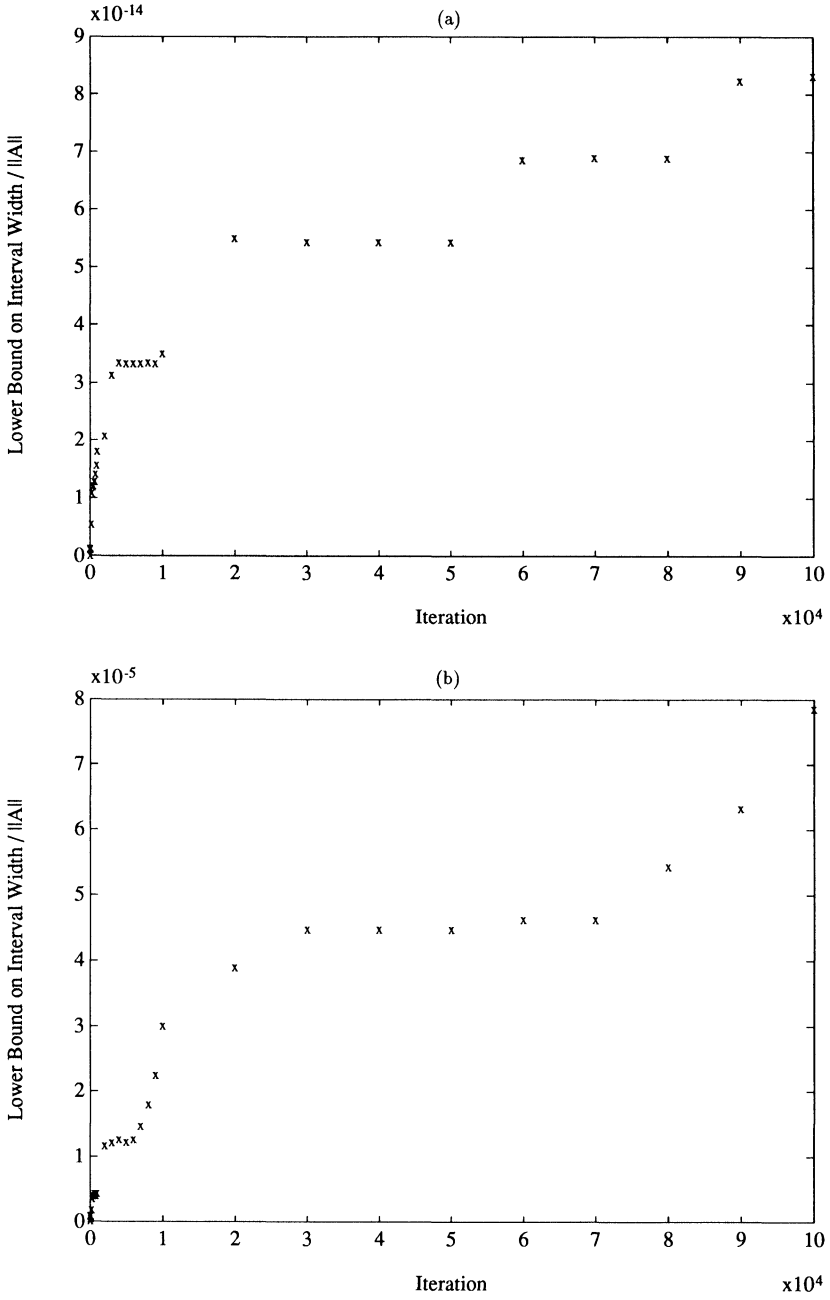

FIG. 10. (a) *Double precision Lanczos* (ρ = .6). *Run for* 10**5 *steps*. (b) *Single precision Lanczos* (ρ = .6). *Run for* 10**5 *steps*.

each of these intervals. Thus, we could then conclude that a finite precision Lanczos computation would eventually find an approximation within δ of each eigenvalue of A.

Whether this interval size can be bounded by a small number δ independent of J is an open question. (Of course, there is some bound that is independent of J. From Gershgorin's theorem and the formulas for the elements of the tridiagonal matrices, it

can be seen that all approximate eigenvalues generated by a finite precision Lanczos computation lie in the interval $[\lambda_{\min} - 2\lambda_{\max}, 3\lambda_{\max}]$. Hence the measure for which the characteristic polynomials are orthogonal has its support in this set. But this is not a very interesting bound.)

To try and determine whether such a bound δ exists, we have run a finite precision Lanczos computation for the case $\rho = .6$, $n = 24$, for 10^5 steps, and we have computed the spread of the eigenvalue approximations clustered about the largest eigenvalue of A. That is, we have taken all eigenvalue approximations that are closer to the largest eigenvalue of A than to any other eigenvalue of A, and we have computed the difference between the largest of these and the second smallest of these. By the interlace theorem, every future tridiagonal matrix will have an eigenvalue greater than the largest of these approximations and an eigenvalue between each pair of these approximations, and so, this difference gives a lower bound on the interval size δ in which the weighted points lie.

Results using double and single precision arithmetic are plotted in Figs. 10(a) and 10(b). For these experiments, we used the standard formulation of the Lanczos algorithm, rather than the CG form presented earlier, to avoid problems with underflow. As can be seen from the figure, this lower bound continues to grow with J at least out to $J = 10^5$, but it is growing *very* slowly. Whether it stops growing at some value significantly less than the distance to the next largest eigenvalue, or whether these eigenvalue approximations would eventually fill the entire Gershgorin interval, we cannot say. This remains an open question.

6. Conclusions. We have found the analogy between finite precision CG/Lanczos computations and the exact algorithms applied to a larger matrix with nearby eigenvalues to be useful in understanding and predicting the behavior of such computations. The proven identity between finite precision computations for A and exact computations for $\bar{A}$ enables one to prove results about finite precision CG/Lanczos computations. The demonstrated similarity between finite precision computations for A and exact computations for matrices $\hat{A}$ enables one to estimate the actual behavior of finite precision CG/Lanczos computations. It provides a nice explanation of the phenomena observed in [18], for example. The proven bound on the size of the intervals containing the eigenvalues of $\bar{A}$ is far from optimal, however, and it is hoped that this might be improved upon. Interesting open questions remain about whether a finite precision Lanczos computation eventually finds all eigenvalues and about how the algorithm behaves if run for huge numbers of steps.

REFERENCES

[1] T. CHIHARA, *An Introduction to Orthogonal Polynomials*, Gordon and Breach, New York, 1978.

[2] H. P. CROWDER AND P. WOLFE, *Linear convergence of the conjugate gradient method*, IBM J. Res. Develop., 16 (1972), pp. 431–433.

[3] J. CULLUM AND R. WILLOUGHBY, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Vol. I. Theory*, Birkhäuser, Boston, 1985.

[4] P. CONCUS, G. H. GOLUB, AND D. P. O'LEARY, *A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations*, in Sparse Matrix Computations, J. R. Bunch and D. J. Rose, eds., Academic Press, New York, 1976.

[5] M. ENGELI, T. GINSBURG, H. RUTISHAUSER, AND E. STIEFEL, *Refined Iterative Methods for Computation of the Solution and the Eigenvalues of Self-Adjoint Boundary Value Problems*, Birkhäuser-Verlag, Basel, Stuttgart, 1959.

[6] J. FAVARD, *Sur les Polynomes de Tchebicheff*, C.R. Acad. Sci. Paris, 200 (1935), pp. 2052–2053.

[7] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, Second Edition, The Johns Hopkins University Press, Baltimore, MD, 1989.

[8] J. F. GRCAR, *Analyses of the Lanczos algorithm and of the approximation problem in Richardson's method*, Ph.D. thesis, Department of Computer Science, University of Illinois, Urbana, IL, 1981.

[9] A. GREENBAUM, *Comparison of splittings used with the conjugate gradient algorithm*, Numer. Math., 33 (1979), pp. 181–194.

[10] ———, *Behavior of slightly perturbed Lanczos and conjugate-gradient recurrences*, Linear Algebra Appl., 113 (1989), pp. 7–63.

[11] M. R. HESTENES AND E. STIEFEL, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Standards, 49 (1952), pp. 409–436.

[12] C. LANCZOS, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Res. Nat. Bur. Standards, 45 (1950), pp. 225–280.

[13] C. C. PAIGE, *The computation of eigenvalues and eigenvectors of very large sparse matrices*, Ph.D. thesis, Institute of Computer Science, University of London, London, U.K., 1971.

[14] ———, *Error analysis of the Lanczos algorithm for tridiagonalizing a symmetric matrix*, J. Inst. Math. Appl., 18 (1976), pp. 341–349.

[15] B. N. PARLETT AND D. S. SCOTT, *The Lanczos algorithm with selective orthogonalization*, Math. Comp., 33 (1979), pp. 217–238.

[16] J. K. REID, *On the method of conjugate gradients for the solution of large sparse linear systems*, in Large Sparse Sets of Linear Equations, J. K. Reid, ed., Academic Press, New York, 1971, pp. 231–254.

[17] H. D. SIMON, *The Lanczos algorithm with partial reorthogonalization*, Math. Comp., 42 (1984), pp. 115–136.

[18] Z. STRAKOS, *On the real convergence rate of the conjugate gradient method*, Linear Algebra Appl., 154/156 (1991), pp. 535–549.

[19] H. VAN DER VORST, *The convergence behavior of preconditioned* CG *and* CG-S *in the presence of rounding errors*, in Preconditioned Conjugate Gradient Methods, O. Axelsson and L. Yu Kolotilina, eds., Lecture Notes in Mathematics 1457, Springer-Verlag, Berlin, 1990.

[20] H. WOZNIAKOWSKI, *Roundoff error analysis of a new class of conjugate gradient algorithms*, Linear Algebra Appl., 29 (1980), pp. 507–529.