JOURNAL OF COMPUTATIONAL PHYSICS 82, 218–228 (1989)

A New Iterative Scheme for Obtaining Eigenvectors of Large, Real-Symmetric Matrices

RAMESH NATARAJAN* AND DAVID VANDERBILT

Department of Physics, Harvard University, Cambridge, Massachusetts 02138

Received February 17, 1988; revised July 28, 1988

A rapidly convergent block-iterative scheme for the computation of a few of the lowest eigenvalues and eigenvectors of a large dense real symmetric matrix is proposed. The method is especially applicable to matrix eigenvalue problems that arise from the discretization of self-adjoint partial differential equations. One such application to certain symmetric matrices that arise in solid-state band structure calculations is considered in detail. The most time-consuming parts of the present algorithm are a matrix multiplication and a Gauss–Siedel relaxation step which are performed on each iteration. These two parts can, however, be very efficiently implemented on a vector or parallel processing computer. © 1989 Academic Press, Inc.

1. INTRODUCTION

In this paper we are interested in obtaining the lowest few solutions of the eigenvalue problem

$$A\mathbf{x} = \lambda \mathbf{x} \tag{1}$$

where A is an $n \times n$ square symmetric matrix. We are interested in this problem in the context of the solution of a linear partial differential equation using a Fourier expansion. (In our solid-state physics applications it is the quantum-mechanical Schrödinger equation in three dimensions.) Thus, $\mathbf{x}$ represents the unknown amplitudes in the expansion of the solution (in our case, the wavefunction) in a Fourier basis set. Typically the desired lowest solutions are composed primarily of low Fourier components. Thus, if the basis vectors ("plane waves" in three dimensions) are ordered by magnitude, the resulting matrix A is dense and is dominated by diagonal elements A_{kk} for large k, and a leading submatrix of A can be solved to obtain a first approximation to the desired solutions. The method proposed here is more generally intended for any matrices of this type, i.e., dense, diagonally dominant matrices that arise in applications where one can choose a basis in which the first few basis vectors almost span the desired solutions. In our case, a sufficiently accurate approximation to the wavefunction requires the dimension of A to be on

* Present address: P. O. Box 704, IBM T. J. Watson Research Center, Yorktown Heights, NY 10598.

0021-9991/89 $3.00

the order of 2000, of which typically the 100 lowest eigenvalue-eigenvector pairs are required. Other discretization schemes, such as finite differences, lead to a sparse matrix, which, however, must be much larger to obtain the same accuracy of discretization.

The Eispack algorithm (Garbow *et al.* [1]) for computing the eigenspectrum of (1) uses a two step procedure. In the first step, a series of Householder similarity transformations reduces A to a tridiagonal symmetric matrix. Following this, the QL algorithm is used on the resulting tridiagonal matrix to determine the eigenvalues and eigenvectors. This procedure, which is implemented in the Fortran subroutines TRED2 and TQL2, is not very useful when the matrix A is large. One difficulty is the large $O(n^3)$ operation count required by these algorithms in order to obtain the eigenvalues and eigenvectors even if, as in our application, only a fraction of the eigenvalues and eigenvectors are desired. Another difficulty is the $O(n^2)$ storage requirement, since efficient processing using the Eispack routines (particularly TRED2) requires that the entire A matrix be held in fast random access memory.

Parlett [2] has reviewed many of the direct iterative schemes that are applicable to the eigenvalue problem (1). Of these, inverse iteration (with shift) provides a rapid linearly convergent procedure for computing the eigenvector corresponding to the eigenvalue that is closest to the chosen shift. However, the factorization of a dense matrix that is required in this algorithm has an $O(n^3)$ operation count and this diminishes its utility for large matrices. On the other hand, efficient out-of-core algorithms exist for the matrix factorization. These algorithms are especially designed to minimise data transfer or paging when the matrix A must be stored on a secondary memory storage device such as disk. Because of this, the inverse iteration procedure is preferable to the Eispack procedures for large matrices, where the cost of I/O can dominate the actual computational cost. In addition, if the matrix A is sparse or banded, then a number of specialized matrix factorization techniques with reduced operation counts can also be used.

Other iterative schemes, some of which are discussed below, have also been proposed for the computation of a portion of the eigenspectrum of a symmetric matrix. These schemes are designed to have a lower $O(n^2)$ operation count than the usual inverse iteration method while retaining the fast rate of convergence of the latter. These fast convergence characteristics derive from the use of the minimax property of the Rayleigh quotient

$$\rho(\mathbf{x}) = \frac{(\mathbf{x}, A\mathbf{x})}{(\mathbf{x}, \mathbf{x})} \tag{2}$$

to recompute the shifts and basis vectors variationally in each iteration. In particular, given a subspace $\mathscr{E}^l$, $l \leqslant n$, the Rayleigh–Ritz principle applied to (2), can be used to determine the most optimal approximation to the l lowest eigenvalues and eigenvectors from $\mathscr{E}^l$. These estimates are obtained by exactly solving the l-dimensional eigenvalue problem that results form projecting the matrix A into the

subspace $\mathscr{E}^l$. The accuracy of the results can be refined in either of two ways. The first way is to increase the dimension of the approximating subspace until the desired convergence is obtained. The second way is to keep the subspace dimension fixed but to compute iteratively an improved approximating subspace, i.e., one that more effectively spans the eigenspace of the desired eigenvalues. These two strategies can be combined to accelerate the convergence. Selective convergence on the eigenspace of higher eigenvalues of A is also possible with the Rayleigh–Ritz procedure by using an approximating subspace that is orthogonal to the eigenspaces of lower eigenvalues (assuming that the latter have somehow been accurately determined).

The most straightforward of the $O(n^2)$ iterative schemes is subspace iteration (Jennings, [3]), which is used to compute the eigenvectors corresponding to the dominant eigenvalues (in absolute magnitude) of A. In this method, a subspace of dimension l, where l is larger than the number of desired eigenvectors, is chosen to initiate the computation. At each iteration, the basis vectors of this subspace are premultiplied by A to yield an l dimensional approximating $\mathscr{E}^l$. This premultiplication has the effect of amplifying the components of the dominant eigenvectors in the original basis—much as in the usual Power method (Wilkinson [4]). The matrix A is diagonalized after projecting into $\mathscr{E}^l$ and the resulting eigenvalues and backtransformed eigenvectors are tested for convergence. If convergence is achieved, then the backtransformed eigenvectors provide an updated basis and the cycle starting with the premultiplication by A is repeated. The most computationally intensive step is the matrix multiplication which has an operation count of $O(ln^2)$. This step, however, can be efficiently implemented on a computer with vector or parallel processing capabilities (Natarajan, [5]).

Lanczos [6] has proposed another $O(n^2)$ scheme, in which at the ith iteration the approximating subspace consists of the Krylov subspace of A of order i. The choice of an orthogonal basis set for this subspace leads to the useful property that the intermediate matrices to be diagonalized are tridiagonal; hence, this diagonalization can be very efficiently carried out using Sturm sequence techniques (such as implemented in the Eispack routine, TSTURM). However, the Lanczos algorithm is known to be susceptible to roundoff error growth. Nevertheless, modifications to the basic algorithm, such as intermediate selective reorthogonalization of the Krylov subspace basis vectors to the already converged eigenvectors (Scott, [7]) are quite effective in overcoming the roundoff problem. A block version of the original Lanczos method is required to treat the case when the complete eigenspaces of multiple eigenvalues need to be determined. In this case, the resulting intermediate matrices are block-tridiagonal rather that tridiagonal. Further details on the Lanczos method may be found in Cullum and Willoughby [8].

Davidson [9] has proposed an algorithm that provides a faster convergence than the Lanczos method especially when the desired eigenvalues are closely spaced. Ideally, in the case of closely spaced eigenvalues, an inverse iteration with shift would be used to provide better convergence than the methods that rely on a premultiplication with A. However, as discussed earlier, the complete inverse

iteration step is prohibitively expensive for such large matrices. In the Davidson procedure therefore, the exact matrix factorization is replaced by one step of a Jacobi relaxation. For diagonally dominant matrices of the kind which arise in the current application, this enables the components of the unwanted eigenvectors to be damped out from the approximating subspace basis in a relatively inexpensive way.

A generalized block version of the Davidson algorithm has been developed by Liu [10] for simultaneously determining the k lowest eigenvalues and eigenvectors of A. The details of this algorithm are as follows. A subspace $\mathscr{E}^l$, of dimension $l > k$, whose basis consists of the orthonormal vectors $\{\mathbf{b}_j^{(1)}\}$ $(j = 1, ..., l)$, is chosen to initiate the computation. These basis vectors define an $n \times l$ orthogonal matrix $Q^{(1)}$ in the form

$$Q^{(1)} = [\mathbf{b}_1^{(1)}, \mathbf{b}_2^{(1)}, ..., \mathbf{b}_l^{(1)}]. \tag{3}$$

The matrix $\bar{A}$ is formed by projecting A into this subspace as follows

$$\bar{A} = Q^{(1)\mathrm{T}} A Q^{(1)} \tag{4}$$

and the eigenvalues $\lambda_j^{(1)}$ $(j = 1, ..., l)$ and corresponding eigenvectors of $\bar{A}$ are determined. These eigenvalues and the backtransformed eigenvectors $\mathbf{x}_j^{(1)}$ are examined for convergence by checking the residuals $\mathbf{q}_j^{(1)}$ of the lowest k desired eigenpairs

$$\mathbf{q}_j^{(1)} = (A - \lambda_j^{(1)} I)\, \mathbf{x}_j^{(1)}. \tag{5}$$

If convergence has not occurred, the inverse iteration equations are formulated but instead of an exact solution, a single step of a Jacobi relaxation is carried out to yield a set of updates in the form

$$\delta \mathbf{x}_{k,j}^{(1)} = -(A_{kk} - \lambda_j^{(1)})^{-1}\, \mathbf{q}_{k,j}^{(1)}. \tag{6}$$

The set of update vectors are orthonormalized to each other and to the original basis of $\mathscr{E}^l$ to yield the orthonormal set $\{\mathbf{b}_j^{(2)}\}$ $(j = 1, ..., l + k)$. This set when added to the original basis of $\mathscr{E}^l$ leads to a basis for a new subspace $\mathscr{E}^{l+k}$. This basis also defines a new $n \times (l + k)$ orthogonal projection matrix into this subspace

$$Q^{(2)} = [\mathbf{b}_1^{(1)}, ..., \mathbf{b}_l^{(1)}; \mathbf{b}_1^{(2)}, ..., \mathbf{b}_k^{(2)}] \tag{7}$$

which is used in the next iteration. We note that this procedure generates a sequence of subspaces of dimension l, $l + k$, $l + 2k$, ... in which the eigenvalue problem for $\bar{A}$ must be solved exactly. In general $k \ll l \ll n$ so that the time consumed in this portion of the calculation, although $O(l^3)$, is small relative to the matrix multiplication required to compute the right-hand side in (5). However, if the desired convergence is not rapidly attained, the dimension of the approximating subspace may grow to become very large. When this happens, the calculation is usually stopped and restarted using only the basis computed from the most recent iteration. Modified versions of the Davidson and Davidson–Liu methods have been discussed by Davidson [11], Kosugi [12], and Morgan and Scott [13].

Bendt and Zunger [14] (see also Wood and Zunger [15]) have presented an algorithm with some modifications to the usual Davidson method (as presented above, with $k=1$). They suggest the use of a different expansion basis for x in (1) that consists of vectors that are good approximations to the eigenvectors of A. These vectors can be obtained for example by diagonalizing a leading submatrix of A. The matrix A expressed in this basis is more likely to be diagonally dominant and this leads to a better justification for using a Jacobi relaxation procedure. The use of such a nonorthogonal basis set results in a generalized eigenvalue problem to be solved on each iteration. In addition, they suggest the use of an approximating subspace whose basis consists of only the initial guess to the desired eigenvector and the set of updates computed on each iteration. This leads to faster execution for the diagonalization step since the order of the problem on the ith iteration is only $i+1$, as opposed to $i+I$ in the usual single-vector Davidson method. This modification is especially useful when many iterations are required for convergence. In this case, the order of the matrix to be diagonalized in Eq. (4) is eventually so large that the time taken for this step is comparable to the dominant matrix multiplication step. However, the method does not incorporate some of the attractive features of the block Davidson–Liu procedure, especially the ability to solve for complete eigenspaces of degenerate eigenvectors.

2. Present Scheme

The present work is primarily directed towards obtaining an algorithm for matrix eigenvalue problems that arise from the discretization of self-adjoint partial differential equations. In such problems, it is often useful to start with a coarse mesh discretization; for example, one can diagonalize in the space of low Fourier components using standard Eispack routines, to provide initial guesses to the desired eigenvalues and eigenvectors. The accuracy of the coarse mesh estimates can then be improved by making use of one of the Rayleigh quotient-based iterative techniques, using the matrix obtained from the fine mesh discretization (e.g., the full set of Fourier basis vectors). We propose a modified Rayleigh quotient-based technique which is particularly suited to such problems, and which has the advantages of being simple yet robust and efficient.

The changes that are made to the Davidson–Liu algorithm are as follows. First, a Gauss–Seidel update is used instead of the Jacobi update on each iteration, as suggested by Morgan and Scott [13] (see also Nex, [16]). That is, instead of using (6) on the ith iteration, we solve the lower triangular set of equations

$$(A_L - \lambda_j^{(1)} I)\, \delta \mathbf{x}_j^{(i)} = -\mathbf{q}_j^{(i)}, \tag{8}$$

where

$$A_{Lk,j} = \begin{cases} A_{k,j} & \text{if } k \geqslant j \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

Tests (see Section 4) show that the use of the Gauss–Seidel update provides a faster convergence. The Jacobi procedure has the advantage of an $O(n)$ operation count as well as lower I/O costs if the diagonal entries of A can be stored in core memory. However, the ostensibly greater $O(n^2)$ operation count and I/O requirements of the Gauss–Seidel procedure can be disposed of by effectively combining the calculations in (9) with the matrix multiplication step (5) (Froyen, [17]). Note that further improvements involving retention of a full leading block of A in (8) are possible (Nex, [16]) but have not been implemented here.

Second, we modify the form of the approximating subspace as follows. On each iteration, the approximating subspace $\mathscr{S}^{2l+m}$ is that spanned by: (i) the l trial vectors from the previous iteration; (ii) the l Gauss–Seidel updates; and (iii) the first m unit vectors. Clearly, any scheme that retains only partial information from the previous iterates, like the present one, cannot be more effective than a scheme that retains information from all previous iterates (witness the improved convergence properties of the Lanczos method in contrast to the power method, Cullum and Willoughby, [8]). However, in the present case this deficiency is ameliorated by the inclusion of the unit vectors, (iii), which add variational freedom. In fact, they correspond to a restriction of the matrix A in (1) to a coarse or truncated basis set expansion; hence, these vectors are expected to have a large projection on the eigenspace of the lower eigenvalues. Thus it is particularly worthwhile to determine these leading (low-Fourier) components exactly on each iteration, as done here.

The resulting projection matrix, on the ith iteration, is

$$K = [\mathbf{e}_1, ..., \mathbf{e}_m; \mathbf{b}_1^{(i-1)}, ..., \mathbf{b}_l^{(i-1)}; \mathbf{b}_1^{(i)}, ..., \mathbf{b}_l^{(i)}], \tag{10}$$

where $\mathbf{e}_j$ denotes the usual jth unit vector. Note that the matrix K, unlike Q in (3), is not orthogonal. The projection of A into this subspace transforms (1) into a $(2l+m)$-dimensional generalized eigenvalue problem

$$\bar{A}\mathbf{y} = \lambda \bar{B}\mathbf{y}, \tag{11}$$

where

$$\bar{A} = K^{\mathrm{T}}AK, \qquad \bar{B} = K^{\mathrm{T}}K, \qquad \mathbf{x} = K\mathbf{y}. \tag{12}$$

One advantage of the present approach over the Davidson–Liu approach is that the generalized eigenvalue problem that must be solved exactly on each iteration is of constant size (i.e., $2l+m$). This feature simplifies programming considerably.

3. Computational Details

The present algorithm has been implemented in the form of a Fortran subroutine ITEIG, whose details we summarize below. To begin with, we need initial guesses

for the lowest l eigenvalues and corresponding eigenvectors. These may be provided by the user; otherwise an option is provided to have them generated within the program by using the usual Eispack procedures to diagonalize an appropriate leading submatrix of A. We take these initial guesses in the form

$$(\lambda_1^{(0)}, \mathbf{x}_1^{(0)}), (\lambda_2^{(0)}, \mathbf{x}_2^{(0)}), ..., (\lambda_l^{(0)}, \mathbf{x}_l^{(0)}). \tag{13}$$

On the first iteration, we compute

$$z_j^{(0)} = A\mathbf{x}_j^{(0)} \qquad \text{for} \quad j = 1, ..., l \tag{14}$$

and then execute the following loop for $k = 0, 1, ...$ until convergence:

(i) Calculate the residuals

$$\mathbf{q}_j^{(k)} = -\mathbf{z}_j^{(k)} + \lambda_j^{(k)}\mathbf{x}_j^{(k)} \qquad \text{for} \quad j = 1, ..., l \tag{15}$$

and check the norms of these residuals for convergence in the desired lowest eigenpairs. The loop is terminated upon detecting convergence.

(ii) Calculate the updates to the vectors in the basis set by solving the lower triangular system

$$(A_L - \lambda_j^{(k)} I)\, \delta\mathbf{x}_j^{(k)} = \mathbf{q}_j^{(k)} \qquad \text{for} \quad j = 1, ..., l. \tag{16}$$

(iii) Premultiply the updates by the matrix A to obtain

$$\delta\mathbf{z}_j^{(k)} = A\, \delta\mathbf{x}_j^{(k)} \qquad \text{for} \quad j = 1, ..., l. \tag{17}$$

(iv) Form the entries of the matrices $\bar{A}$ and $\bar{B}$ in (12), with

$$K = [\mathbf{e}_1, ..., \mathbf{e}_m; \mathbf{x}_1^{(k)}, ..., \mathbf{x}_l^{(k)}; \delta\mathbf{x}_1^{(k)}, ..., \delta\mathbf{x}_l^{(k)}] \tag{18}$$

and solve the generalized eigenvalue problem (11) using the relevant Eispack routines to obtain the solutions $(\bar{\lambda}_j^{(k)}, \mathbf{y}_j^{(k)})$. For use with the present program, we have rewritten and vectorized the original Eispack routines. Details of these modifications along with relative performance measurements can be found in Natarajan [5].

(v) Update the estimates for the l lowest eigenvalues and corresponding eigenvectors of A by back-transforming the solutions obtained in (iv) above:

$$\lambda_j^{(k+1)} = \bar{\lambda}_j^{(k)}, \qquad \mathbf{x}_j^{(k+1)} = K\mathbf{y}_j^{(k)} \qquad \text{for} \quad j = 1, ..., l. \tag{19}$$

The vectors $\mathbf{z}_j^{(k+1)}$ required to compute the residuals in (i) during the next iteration can also be obtained at this stage by performing the back-transformation

$$\mathbf{z}_j^{(k+1)} = A\mathbf{x}_j^{(k+1)} = AK\mathbf{y}_j^{(k)} \qquad \text{for} \quad j = 1, ..., l, \tag{20}$$

where

$$AK = [A\mathbf{e}_1, ..., A\mathbf{e}_m; \mathbf{z}_1^{(k)}, ..., \mathbf{z}_l^{(k)}; \delta\mathbf{z}_1^{(k)}, ..., \delta\mathbf{z}_l^{(k)}]. \tag{21}$$

(vi) Increment the iteration counter k and go to (i) for the next iteration.

Note that in step (ii), the matrix $|A_{ii} - \lambda_j^{(k)}|$ is nearly singular whenever a trial eigenvalue nearly coincides with a root A_{ii} of A_L. In this case the ith component of $\delta\mathbf{x}_j^{(k)}$ cannot be reliably determined, and we set it to zero whenever $|A_{ii} - \lambda_j^{(k)}|$ is less than a preset cutoff value (taken to be of the order of the spacing between roots of A_L). This plays a role similar to the introduction of shifts in other methods. Actually, in our method the ith component of $\delta\mathbf{x}_j^{(k)}$ is set to zero whenever $i \leqslant m$, since the ith expansion vector $\mathbf{e}_i$ is included in the basis exactly anyway. Normally A will have been arranged so that its diagonal elements increase down the diagonal, in order that the first m basis vectors included exactly in step (iv) will be those most likely to have the largest projection on the desired eigenvectors. In this case, if m is chosen large enough so that $A_{mm} > \lambda_l^{(k)}$, the $|A_{ii} - \lambda_j^{(k)}|$ cutoff will never be encountered. Usually $m \geqslant l$ is sufficient to ensure this, and the method seems to work best in this case (see Section 4).

As mentioned in Section 2, the program actually combines steps (ii) and (iii), so that the total operation count for these two steps is n^2l instead of $\frac{3}{2}n^2l$. Moreover, the matrix elements of A are required only a column at a time within the outer loop; this is critical in order to minimise I/O costs when the matrix is too large to fit in central memory. In the latter case, the program provides a choice of two modes of operation. In the first, the matrix A is precomputed and stored so that the program pages through the matrix once per iteration. In the second mode, the matrix is not precomputed but rather its entries are computed as required through a user supplied routine MATROW. This routine, upon invocation, computes a desired row (or equivalently, a column) of the matrix A. Again, MATROW is only called once per row per iteration. The best choice between these options depends on whether it is less expensive to recompute the matrix or to page it from disk once.

4. Results

A test matrix of order 1459 (corresponding to a Fourier representation of the quantum-mechanical Schrödinger equation for an electron in a crystal) was generated in order to study the behavior of the present algorithm. It was desired to obtain converged solutions to the lowest 48 eigenvalues and corresponding eigenvectors. The convergence was monitored by computing the largest norm from among the residual vectors obtained from this set at each iteration. The computations were initialized by diagonalizing a leading submatrix of order 510 for each test case. All timings are reported in cpu seconds on the Cyber 205.

Figure 1 shows that the Gauss–Seidel update leads to a faster convergence. In order for the error to drop to a value below 5×10^{-3} only six iterations were

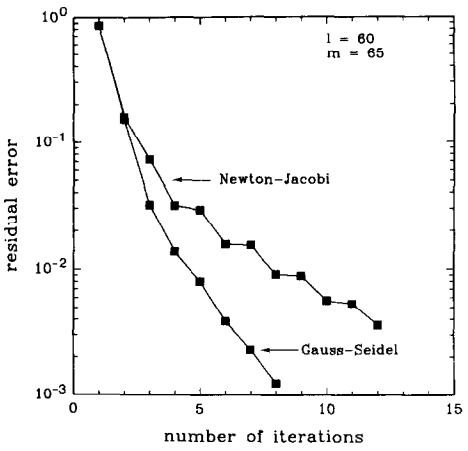

FIG. 1. The effect of the choice of iteration scheme (Gauss–Seidel versus Newton–Jacobi) on convergence. The residual error plotted is the largest from among the lowest 48 eigenvalue–eigenvector pairs. The blocksize is fixed at $l = 60$ and the number of exactly included basis vectors is fixed at $m = 65$. The Cyber 205 execution times per iteration were 4.2 and 4.9 s for the Newton–Jacobi and Gauss–Seidel methods, respectively.

required with the Gauss–Seidel update; the corresponding figure for the Jacobi update was 12 iterations. This difference in performance between these two schemes becomes sharper when the convergence criterion is decreased. For meaningful comparison, the cpu time taken in order to reduce the error to a constant value must also be considered. In the Gauss–Seidel case this was 29.4 s while in the Jacobi case it was 50.4 s.

In Fig. 2, we show the effect of varying the block size, i.e., the quantity l in Section 3, on the convergence. Again, we see that using a larger basis leads to a faster convergence. However, there appears to be a limit to the increase in performance that is attainable in this fashion since the separation between the convergence curves in increasing l from 66 to 72 is not as great as that obtained in increasing l from 54 to 60. Since increasing the magnitude of l also increases the cpu time, there is an optimal choice for this quantity. As a rule of thumb, we suggest choosing l about 25% larger than the number of desired eigenvectors.

The effect of including the m original basis vectors exactly to the approximating subspace in each iteration is considered in Fig. 3. Again a pronounced increase in the rate of convergence can be seen, especially in the early stages. Just as for the choice of the block size l, there is an optimal value of m that achieves a balance between faster convergence and the increased time per iteration. We have found that $m \cong l$ is typically a good choice.

For the test matrix studied, the choices $l = 60$ and $m = 65$ used in Fig. 1 were nearly optimum. The total diagonalization time using optimally vectorized standard techniques for a matrix of this size is approximately 240 s, so that an order-of-magnitude speedup was realized for this case. As with other iterative methods, the

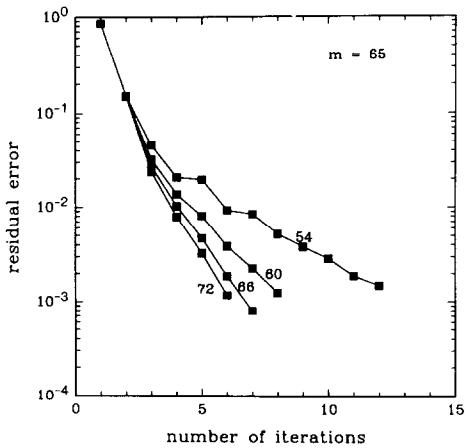

FIG. 2. The effect on convergence of varying the block size l (i.e., l is the number of iterated vectors). The residual error is the largest from among the lowest 48 eigenvalue–eigenvector pairs. The number m of exactly included basis vectors is fixed at 65. The Cyber 205 execution times per iteration were 4.3, 4.9, 5.6, and 6.2 s for the cases $l = 54$, 60, 66, and 72, respectively.

speedup is even more dramatic if the diagonalization step is part of an outer iterative loop as often occurs in "self-consistent" or "mean-field" calculations. In this case, the output eigenvectors from the previous pass through the outer loop provide good starting vectors for the algorithm so that, especially in the late stages of such calculations, only one or two update iterations may be needed to obtain convergence.

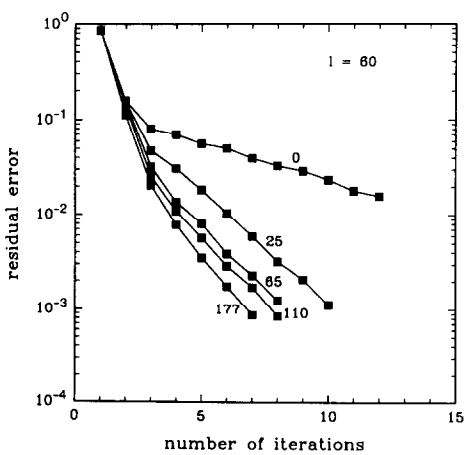

FIG. 3. The effect on convergence of varying the number m of exactly included low-order expansion basis vectors. The blocksize l is fixed at 60. The Cyber 205 execution times per iteration were 4.1, 4.4, 4.9, 5.7, and 7.5 s for the cases $m = 0$, 25, 65, 110, and 177, respectively.

5. SUMMARY

The method described here is intended to represent a reasonable compromise between simplicity, efficiency, and robustness. It is simpler than some of the previous algorithms because the subspace size does not increase with iteration number. This also allows us to avoid difficulties with overcomplete subspaces that sometimes occur in these other algorithms after several iterations. In addition, it is quite efficient in the sense that twofold or greater reductions in the residuals per iteration are not hard to obtain, even for matrices larger than the one tested above. The method has no difficulty in handling degenerate eigenvalues and can handle matrices that exceed the physical memory size without serious performance degradation. It is now being used routinely to diagonalize matrices of order 3000 and larger in a production environment.

ACKNOWLEDGMENTS

Computer time for this work was provided through the Harvard local allocation at the John von Neumann Center at Princeton, which is funded by the National Science Foundation. One of use (Vanderbilt) acknowledges support from NSF Grant DMR-85-14638. We are indebted to Sverre Froyen for important suggestions. We thank Kai-Ming Ho and D. C. Sorensen for useful discussions.

REFERENCES

1. B. GARBOW, J. M. BOYLE, J. J. DONGARRA, AND C. B. MOLER, *Matrix Eigensystems Routines—Eispack Guide Extension* (Springer-Verlag, New York, 1977).
2. B. N. PARLETT, *The Symmetric Eigenvalue Problem* (Prentice–Hall, Englewood Cliffs, NJ, 1980).
3. A. JENNINGS, in *Sparse Matrices and Their Uses*, edited by I. S. Duff (Academic Press, London, 1981).
4. J. H. WILKINSON, *The Algebraic Eigenvalue Problem* (Clarendon Press, Oxford, 1965).
5. R. NATARAJAN, IBM T. J. Watson Research Center Technical Report RC-13625, 1988 (unpublished).
6. C. LANCZOS, *J. Res. Nat. Bur. Stand.* **45**, 255 (1950).
7. D. S. SCOTT, in *Sparse Matrices and Their Uses*, edited by I. S. Duff (Academic Press, London, 1981), p. 139.
8. J. K. CULLUM AND R. A. WILLOUGHBY, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations. Vol. I. Theory* (Birkhauser, Boston, 1985).
9. E. R. DAVIDSON, *J. Comput. Phys.* **17**, 87 (1975).
10. B. LIU, in *Report on the Workshop "Numerical Algorithms in Chemistry: Algebraic Methods,"* edited by C. Moler and I. Shavitt (Lawrence Berkeley Lab., Univ. of California, 1978), p. 49.
11. E. R. DAVIDSON, in *Proceedings, NATO Advanced Study Institute on Methods in Computational Physics*, edited by G. H. F. Diercksen and S. Wilson (Reidel, Dordrecht, 1983), p. 95.
12. N. KOSUGI, *J. Comput. Phys.* **55**, 426 (1984).
13. R. B. MORGAN AND D. S. SCOTT, *SIAM J. Sci. Stat. Comput.* **7**, 817 (1986).
14. P. BENDT AND A. ZUNGER, Solar Energy Research Institute Technical Report TP-212-1698, 1982 (unpublished).
15. D. M. WOOD AND A. ZUNGER, *J. Phys. A* **18**, 1343 (1985).
16. C. M. M. NEX, *J. Comput. Phys.* **70**, 138 (1987).
17. S. FROYEN, Solar Energy Research Institute, Golden, CO, private communication (1987).