

AD-A127 972

THE SOFTWARE SCENE IN THE EXTRACTION OF EIGENVALUES
FROM SPARSE MATRICES(U) CALIFORNIA UNIV BERKELEY CENTER
FOR PURE AND APPLIED MATHEMATICS B N PARLETT MAR 83
PAM-132 N00014-76-C-0013

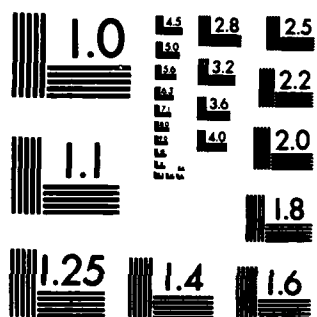
1/1

UNCLASSIFIED

F/G 9/2

NL

END
DATE FILMED
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

DA 127972



REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER <i>PA-132</i>	2. GOVT ACCESSION NO. <i>AD A127 972</i>	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) The software scene in the extraction of eigen- values from sparse matrices		5. TYPE OF REPORT & PERIOD COVERED Unclassified
7. AUTHOR(s) B.N. Parlett		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS University of California Department of Mathematics Berkeley, CA 94720		8. CONTRACT OR GRANT NUMBER(s) N00014-76-C-0013
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE <i>March 1983</i>
		13. NUMBER OF PAGES
		15. SECURITY CLASS. (of this report)
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report)		
DISTRIBUTION STATEMENT A Approved for public release; Distribution Unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)		
The class of users of programs in this area is discussed and split into the sporadic subsets. Available software for each group is reviewed and some current developments are outlined. This essay arose from a talk delivered at the Sparse Matrix Symposium held at Fairfield Glade, Tennessee in October, 1982.		

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

S-N 0102-LF-014-6601

URITY CLASS:

ON OF THIS PAGE(When Data Entered)

The Software Scene in The Extraction of Eigenvalues from Sparse Matrices

B. N. Parlett

ABSTRACT

The class of users of programs in this area is discussed and split into the intensive and the sporadic subsets. Available software for each group is reviewed and some current developments are outlined.

This essay arose from a talk delivered at the Sparse Matrix Symposium held at Fairfield Glade, Tennessee in October, 1982.

DTIC
ELECTE
MAY 10 1983
B

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
PER LETTER	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	

DTIC
ELECTE
MAY 10 1983

Research supported by Office of Naval Research Contract N00014-76-C-0013.

I. Introduction

Numerical analysts see their task as the development and analysis of methods for computing quantities which arise in science and engineering. Some assume that any good algorithm will soon find a user. Who knows if that assumption is warranted? In any case this essay will begin by taking a step back from software to consider who wants eigenvalues and eigenvectors of sparse matrices of large order. It is encouraging to see that some computing services are automatically monitoring the use of programs in their libraries (e.g. Sandia Labs in Albuquerque). Such practices will help at least a few people assess the actual usage of all those carefully written subroutines.

Our investigations though limited, all point to an important distinction between users that goes a long way towards explaining the present state of affairs in software development. That is the gist of Sections 3 and 4.

This essay may well be read by those who are not specialists in software for eigenvalue calculations and so there is a section which tells the EISPACK story. The very existence of this set of subroutines is intriguing and it continues to influence the development of mathematical software. Our account is too brief to do justice to its subject but it provides essential background material.

In Sections 6, 7, 8 we discuss the programs which were readily available in September 1982 and then, in Section 9 we describe some programs that are still under development.

Terminology

It is vital to science that one not be more precise than is necessary for the purpose in hand. In this essay words like small, large, and sparse play a key role. Yet their meaning will be relative to the user's computing system.

We say that an $n \times n$ matrix is small if two $n \times n$ arrays can be held in the

fast store, in addition to any programs that are needed, otherwise the matrix is large. This binary distinction should not be pressed too hard or it will crumble. The purists say that a matrix is sparse if it has only a few nonzero elements in each row, not more than 50 say. On this definition a matrix with 90% of its elements zero would not be sparse because $(1/10)n^2 > 50n$ for large enough n ($n > 500$).

Our definition suggests that it is reasonable to use similarity transformations to help find the eigenvalues of small matrices. Our definition also suggests that one should cherish the zero elements of sparse matrices. Where does that leave large, nearly full matrices? Answer: with the chemists - see Section 4.

The exploitation of sparsity in the execution of triangular factorization has given rise to a valuable and vigorous research area called sparse matrix technology. The reviews of Duff describe it well, [Duff, 1982]. It turns out that this technology is not directly relevant to eigenvalue problems. There are two classes of methods: those that employ explicit nondiagonal similarity transformations and those that do not. That is all. If it is attractive to use explicit similarity transformations then please use them. Our concern here is with the other class, whatever the character of the matrix.

This is the place to mention that G.W. Stewart gave a concise review of numerical methods for sparse eigenvalue problems in [Duff & Stewart, 1978]. That covers the background of most of the software we discuss here. Since then there have been important refinements to the Lanczos algorithm. Both Davidson's method and the ideas sketched in the section on recent developments are of more recent vintage.

2. EISPACK

EISPACK is an organized collection of some 40 FORTRAN subroutines which

compute some or all eigenvalues of a given matrix with or without the corresponding eigenvectors. This large number of subroutines reflects the yearning for efficiency. EISPACK has special techniques for real matrices, for symmetric matrices, for tridiagonal matrices ($a_{ij} = 0$ if $|i - j| > 1$), and for upper Hessenberg matrices ($a_{ij} = 0$ if $i - j > 1$).

The first issue of EISPACK appeared in 1974. The package was distributed by the code center of the Argonne National Laboratory, Argonne, Illinois, and the indispensable EISPACK GUIDE was published by Springer-Verlag, New York. To complete most eigenvalue calculations it is necessary to invoke more than one subroutine. For example, a nonsymmetric matrix B might be first "balanced" by a diagonal similarity transformation, $B \rightarrow DBD^{-1} =: C$; next C is reduced to Hessenberg form, $C \rightarrow P^t CP =: H$ with P orthogonal and then H is reduced to triangular form T (to within working accuracy), $H \rightarrow Q^t HQ =: T$ with Q orthogonal. The eigenvalues of B are found on the diagonal of T .

Such detailed knowledge of the process was more than some users wanted to learn. To meet this criticism some members of the EISPACK team wrote an interface called EISPAC which allowed the user to specify the type of matrix and the quantities to be computed. The interface then chose the appropriate subroutines and called them with the right parameter values. Unfortunately EISPAC was only available on IBM systems.

A second edition of EISPACK appeared in 1977 and a third is scheduled for early 1983. Each edition removed blemishes found in some subroutines and added new programs. We enlarge on this topic further on.

What many users of EISPACK do not know is that it represented a cooperative effort by most of the world's experts in matrix computations. It seems safe to assert that not one of the people involved in EISPACK, or its antecedents, became rich in the process. What a contrast to the software produced for

structural analysis! There the temptation to form a company and sell the software for profit was great. Consequently there are enumerable rival packages which provide similar services. The best known are NASTRAN, STRUDL, SAP, MARC, ADINA. This situation provides a nice, because rare, example of competition not leading to superior performance. Was the cooperation and sharing which characterized the numerical analysts due to their admirable sense of values or to the absence of a strong enough industrial demand for pure eigenvalue codes? The solution of this conundrum is left to the reader.

Another remarkable aspect of EISPACK is that its first version deliberately eschewed the production of new algorithms. The goal was "simply" to translate into FORTRAN some of the ALGOL programs in the famous "Handbook for Automatic Computation, volume 2, Linear Algebra", by J. H. Wilkinson and C. Reisch. The authors were the acknowledged leaders in the art of matrix computations. The Handbook appeared in 1971 and represented the truly international cooperation mentioned earlier. Not all the programs were written by Wilkinson or Reisch but each contribution was carefully scrutinized by them and, more often than not, improvements were incorporated into the final versions. Moreover most of the programs had already been published in *Numerische Mathematik* and so had been refereed, and tested, and available in the public domain.

Why mention these details? Because the exercise of translating these ALGOL programs into FORTRAN was far from trivial and, incredible as it may seem, blemishes were found in a number of the Handbook's programs. Of course, the EISPACK team was aiming high. It was not enough to have portable subroutines (for the 1966 standard version of FORTRAN) but these programs were not to suffer much when used on any of the operating systems available at that time. For example, there are virtual memory systems in which storage is split up into pages. The system's algorithm for fetching pages, when they are

not present, can have a strong effect on the time needed to execute a matrix computation. Fortunately it was possible to express the algorithms so that they would perform satisfactorily on all the major systems. This is the nature of work in mathematical software.

The next aspect of EISPACK that should be remembered is the massive effort at testing the programs, in an adversary manner, before their release. About 15 computing groups in the U.S.A. and Canada agreed to install and test the trial programs they received from Argonne. As bugs and blemishes were uncovered there were several iterations on the programs.

A nasty thought must be stirring in the reader's mind. How can an algorithm be published in *Numerische Mathematik* after careful refereeing, be subject to scrutiny and testing by Wilkinson and Reinsch prior to inclusion in the Handbook, be translated into FORTRAN by the EISPACK team with close attention to detail, be tested by various sites charged with that duty, and yet retain a bug or even a blemish? Either the people involved in this development are not truly competent or there is more to the production of software for the mass market than meets the eye. If the latter, then what precisely are the difficulties? We all await a succinct description of the intellectual problems that would establish mathematical software as a genuine subdiscipline of computer science.

Last, but not least, we should emphasize the effort expended by the team on the documentation and uniform coding style. It is a mistake to speak of "good" documentation because documentation which satisfies A may not be suitable for B. It is reasonable to speak of documentation being suitable for a given class of users. The EISPACK GUIDE has intimidated some users but, at the time it appeared, it glowed with virtue in comparison with other examples of documentation. Moreover nothing on this scale had been attempted before by the experts on mathematical software.

Inadvertently EISPACK has provided a common vocabulary (the names of its subroutines) to eigenvalue hunters. EISPACK was a highly visible distillation of what had been learnt by the matrix eigenvalue community during the previous 20 years. It was good for public relations. It predated LINPACK by 3 or 4 years and yet most people consider the eigenvalue problem to be significantly more difficult than the linear equations problem.

To numerical analysts EISPACK seemed to be the solution to the practical eigenvalue problem. What else remained?

1. EISPACK routines are not fast enough for so-called real time computations where the output is needed in microseconds.

2. EISPACK manipulates matrices as conventional two dimensional arrays or uses a conventional one dimensional representation of symmetric matrices. Except for the tridiagonal and Hessenberg form, EISPACK does not try to exploit zero elements in the original matrix. Moreover the stable similarity transformations used by most of the subroutines will quickly destroy any initial sparsity.

So the field is not dead by any means. As users become more sophisticated they are finding more and more need for eigenvalues and eigenvectors. So usage grows.

As the fast storage capacity of many computer systems continues to grow so does the domain of EISPACK. Sound advice to a casual user is to use EISPACK whenever possible, even if the matrix is 500 by 500 and sparse.

Who will then remain unsatisfied? The next section supplies an answer.

3. WHO WANTS EIGENVALUES OF LARGE, SPARSE MATRICES?

EISPACK subroutines have been used quite extensively and it has been assumed that the population of users is so large and so diverse that there is little point in examining the market more closely. Nevertheless it would be nice to

know how strong the demand is for programs which compute some eigenvalues and eigenvectors of a small nonsymmetric matrix.

Even less warranted is the natural assumption that there is a general need for an extension of EISPACK to sparse - and large - matrices. Conversations with a variety of users over several years have forced the author to the following surprising explanation of the current state of affairs.

The market for eigenvalue programs can be divided into two quite different camps: INTENSIVE users and SPORADIC users.

The intensive users already spend millions of dollars a year on the extraction of eigenvalues because spectral analysis is essential to their daily work. They need efficiency and have already crafted their programs to exploit the special features of their tasks. General purpose programs with meticulous code designed to cope with any difficulty which might arise are not cost effective for this group. Of course, the more enlightened intensive users will collaborate with experts as their special purpose software evolves to meet even more exacting demands. Actually the class of intensive eigenvalue hunters also splits into two quite distinct subclasses: those with small, dense matrices who need the output in real time (microseconds) and those who generate larger and larger matrices as they make their mathematical models more realistic. The "real time" users may be driven to solve their problems with hardware rather than software and we will concentrate here on the large matrix problem.

The SPORADIC user has neither the incentive nor the inclination to study matrix methods. The need for some eigenvalues arises and the user wants to obtain them with minimal fuss. Reliability is more important than efficiency and EISPACK is the answer to his prayers. We suggest that the number of sporadic users whose matrices are too large for EISPACK is, and will remain, very low.

The foregoing remarks do not lessen the value of developing good methods for all sorts of large eigenvalue problems. On the other hand the situation should make one hesitate before undertaking the painful chore of producing a general purpose reliable package for most sparse eigenvalue problems.

4. INTENSIVE USERS

The author would welcome corrections and/or extensions to the following list.

I. Structural Engineers

Most eigenvalue calculations arise at the heart of analyses of structures subject to small vibrations. There is an $n \times n$ global stiffness matrix K and an $n \times n$ global mass matrix M . Both are symmetric and real. K is positive definite. The usual task is to find all the eigenvalues λ_i in a given interval at the left end of the spectrum (i.e. near 0), together with their eigenvectors z_i .

$$(K - \lambda_i M)z_i = 0, i=1,2, \dots$$

The z_i determine the shapes of the fundamental modes of vibration of the structure and the λ_i determine the associated natural frequencies $2\pi/\sqrt{\lambda_i}$, $i = 1,2, \dots$.

The nonzero elements of K and M are integrals and more arithmetic operations are needed to form K and M than to compute a few eigenvalues.

In 1978 one company paid 12,000 dollars to obtain 30 eigenvalue/vector pairs for a problem with $n = 12,000$. This cost excludes program development. A good finite element package was used with the technique called subspace iteration for the eigenvalue extraction. Professor E. Wilson (Civil Engineering Department, University of California, Berkeley) estimates that structural engineers spent about 10 million dollars in 1978 on eigenvalue computations. A typical problem today will have $n = 500$ and 20 modes computed.

Nevertheless, there is continued demand to analyze more complicated structures in greater detail. Problems with $n > 10,000$ and 400 modes have been solved.

II. Quantum Chemists

The method of Configuration Interaction (CI) has become the preferred method for deducing observable properties of real and hypothetical molecules from their detailed electronic structure. The CI method approximates the solution of the clamped nuclei Schroedinger equation by expanding it in terms of orthogonal functions made out of products of so-called single and double electron spin orbitals. More details are given in [Shavitt,1977] and [Davidson,1982]. These papers show that interesting, difficult, special eigenvalue problems are being solved quite independently of the numerical analysis community. Great ingenuity goes into the calculation of the real symmetric matrix H (H for Hamiltonian). The nonzero elements of H are multiple integrals and constitute only 10% of the positions. Unfortunately they are scattered over the matrix in a way that precludes any simple structural template. Each eigenvector represents a wave function and its eigenvalue is the energy of the associated state.

In these chemical computations the determination of H requires 10 to 100 times more work than the extraction of the eigenvector belonging to the smallest (i.e. most negative) eigenvalue. Perhaps 100,000 dollars are spent per year in the U.S.A. on the actual eigenvalue/vector computation. Usually only the smallest pair is required. A typical calculation has the order $n = 10,000$ but this activity is expected to increase sharply. During the summer of 1982 the

ethylene molecule C_2H_4 was analyzed in joint work at Cambridge University and the University of California, Berkeley. For this problem

$$n = 1,046,758$$

This calculation demonstrated convincingly the need to include excited states in the expansion. In other words the simpler models in CI are not adequate for the detail required in current investigations.

It is sad that most numerical analysts have never heard of the eigenvalue methods invented by the chemists. These are described in [Davidson, 1982]. The general techniques favored by the numerical analysts are too inefficient for serious (i.e. specialized) tasks.

I have not identified a third group. In the 1960's there was considerable effort expended on the calculation of the criticality constant for nuclear reactors but that activity has subsided. There is much interest in the vibration of the piping systems in modern reactors but that work is part of structural analysis. One candidate for the third position is circuit analysis and control problems but at present that group seems to favor direct solution of their non-linear differential equations.

The intensive users have developed their own eigenvalue software. In fact the structural engineers discovered the method of simultaneous iteration for themselves but gave it a more descriptive name, subspace iteration. The method itself is quite obvious and what counts is the implementation. It is sad that the beautiful program RITZIT, developed by H. Rotishauser in 1968/69 and published in the handbook of Wilkinson and Reinsch, had no influence on the structural engineers working in the U.S.A., although in Britain the work of Jennings did employ some of the techniques embodied in RITZIT. By the time the RITZIT quality does creep into the finite element packages then subspace iteration will have been displaced by modern versions of the Lanczos algorithm. The

story is sad because numerical analysts should justify their professional support by communicating appropriate techniques to users. It is not sufficient to develop, analyze and publish. Moreover, for reasons both good and bad, the engineers will not read the numerical analysis journals and so the missionary work will have to be done by word of mouth.

All the knowledge acquired by the specialists in matrix computations played perhaps no role at all in 90% of the sparse, large eigenvalue extractions made in 1975 in the U.S.A., i.e. in the calculations of the structural engineers and the theoretical chemists. By 1985 the situation may have been reversed. Numerical analysts such as P.S. Jensen at Lockheed (Palo Alto) and J. Lewis at Boeing Computer Services (Seattle) and their colleagues are deeply involved in ambitious structural analyses and will adapt the best techniques they can find to produce really effective software. Portability is nice, it is particularly important for packages aimed at general users, but let us hope that it never becomes a commandment. The glamorous work, at the leading edge of our capacities, will have to exploit every feature that is special to the problem and the computer system. This is the way of Mary (see the next paragraph if you are not familiar with this expression).

Martha complained to Jesus that while she was busy preparing the meals her sister Mary just listened to him. Jesus condoned Mary's behavior and, in some traditions, the way of Martha has stood for the worthy, essential, but dull chores (editorial work, perhaps?). In contrast the way of Mary is "where the action is".

Those sporadic users for whom EISPACK is inadequate and who cannot develop programs for themselves, seem to be in hiding. We presume they exist and so the mundane but worthy task of providing robust, easy-to-use, portable software for standard sparse problem types devolves on the matrix specialists.

It is the way of Martha. We hope that the reader will forgive allusion to the New Testament [Luke 10:38-42].

5. HOW TO EXPLOIT SPARSITY.

It is possible to approximate an eigenvalue of a linear operator by "sampling" at well chosen points in its domain. Thus the simple power method samples the action of a matrix A on the sequence of column vectors $x, Ax, A^2x, A^3x, \dots$. The sequence of Rayleigh quotients of these vectors converges quite rapidly to the dominant eigenvalue of A . Recall that the Rayleigh quotient of a vector v is $v^t A v / v^t v$. Consequently there are methods that need only be given a subroutine, call it OP , which returns Av for any given vector v .

This is the only way in which A appears in the method. The structure and sparsity of A can be exploited by the user in writing code for OP . In other words, the buck is passed to the user! He, or she, is in the best position to take advantage of A 's characteristics to speed up the formation of Av . When v has 10,000 components one pays attention to this product. All the software we describe below actually ignores sparsity completely.

This is in stark contrast to the direct solution of linear equations where a number of clever devices are used to take advantage of zero elements. Some of those sparse techniques may be used by the subprogram OP , but none of that work appears explicitly in the eigenvalue codes. Nevertheless the development of methods based on the user-supplied subprogram OP represents a beautiful division of labor: the method is not obscured by details of A 's structure.

This is the place to mention a serious confusion that has arisen in the past in the assessment of methods. We focus on symmetric matrices for the rest of this section. At a certain level of abstraction the power method is identical to

inverse iteration. One technique works with A , the other with A^{-1} . Inverse iteration performs the useful task of finding the eigenvalue nearest to 0 but pays the significant price of factoring A (to LU) in order to form $\omega = A^{-1}v$ (by solving $Lu = v$ and $U\omega = u$). If a sparse eigenvalue program is used to compute eigenvalues of A it makes an ENORMOUS difference whether the subroutine OP delivers Au or $A^{-1}v$.

Subspace iteration finds eigenvalues close to σ by factoring $A - \sigma I$ and letting OP deliver $(A - \sigma I)^{-1}v$. When $\sigma = 0$ the eigenvalues near 0 are computed first and quite rapidly.

The Lanczos algorithm is somewhat different. It produces eigenvalues at both ends of the spectrum of the linear operator represented by OP , the more extreme ones emerging before the interior ones. Thus Lanczos offers the hope of computing the eigenvalues of A nearest 0 without the pain of invoking some sparse factorization of A . In this sense Lanczos has been compared with Subspace Iteration. The performance depends quite strongly on the matrix but as the order n grows ($n > 100$ say) Lanczos fares worse and worse and is soon eclipsed by Subspace Iteration. Lanczos will have computed perhaps 50 unwanted eigenvalues near ∞ for every eigenvalue near 0. This is because, in most given problems, the larger eigenvalues are much better separated than the small ones. (OP is approximating an unbounded operator.) Although Lanczos is optimal in certain important respects it is a hopeless task to compute the smallest eigenvalue without either factoring A , or solving $Au = b$ iteratively, or having a good approximation to the wanted eigenvector. This is certainly the case when $n > 1,000$.

The solution is easy. Give Lanczos the same OP subroutine as Subspace Iteration. Then the power of Lanczos reveals itself quickly. Both methods are then working with $(A - \sigma)^{-1}$ to find eigenvalues near σ .

6. SOFTWARE FOR THE MASSES

By the time a matrix expert has seen his program appear in a refereed journal he probably never wants to see it again. The program comes to dominate its creator. Dr. R.C. Ward (ORNL) is well aware of the consequent difficulty of getting good matrix programs into the hands of non-expert users. In conjunction with the Sparse Matrix Symposium of 1982 in Fairfield Glade, Ward, and his colleagues at Oak Ridge, have started a public catalog of software for sparse matrix problems, including eigenvalue extraction. This will help to focus the production of good software.

A non-expert will still be annoyed at seeing perhaps six programs all designed for the same task and all based on say the Lanczos algorithm. His annoyance is forgivable but unwarranted. He should be made to understand that the sparse eigenvalue problem is still a research topic. The experts do not yet know the best way to implement the Lanczos algorithm, (to block or not to block, to orthogonalize the Lanczos vectors a lot, a little, or not at all) or how to handle secondary storage. Thus it is good that there are several rival programs until a reasonable consensus is reached. Too many programs is better than none at all. For this reason there is no shame in the fact that the software available now is not up to EISPACK standards. In particular none of the codes has been subjected to widespread testing in a variety of computer systems. It is worth mentioning that Professor Ruhe had some difficulty in transferring his code STLM from the CDC 6600 on which it was developed in Sweden to an identical machine at Boeing. His code suddenly became much less efficient. Why? The operating system was different! The good ship Portability may well founder on the rock called Operating System.

All the programs mentioned below compute one or more eigenvalue/vector pairs of the symmetric problem $(A - \lambda B)z = 0$, unless the contrary is stated.

Brief

comments are given at the side. All programs are in FORTRAN, most in the ANSI standard version of FORTRAN 66. All are portable to some extent, some are completely portable. Between 30 and 60% of the lines are COMMENTS. A potential user should consult Ward's catalog for more information on the programs mentioned below.

A. Programs based on Lanczos

The modern versions of the Lanczos algorithm are not simple and most of them are described in Chapter 13 of [Parlett, 1980]. A brief summary will be given here, but please see the discussion in Section 8 for some important details.

Lanczos algorithms are iterative in nature. A starting vector (or block of vectors) is chosen and at each step a new vector is added to the sequence. In exact arithmetic these vectors are mutually orthogonal and of unit length. A characteristic and pleasing feature is that only the two most recent vectors are needed in the computation of the next one. In addition to these vectors the Lanczos method builds up a symmetric tridiagonal matrix T , each step adds a new row (and column) to T . Quite soon some eigenvalues of T begin to approximate some eigenvalues of the operator hidden in OP . Almost always it is the extreme eigenvalues which are well approximated.

Sometimes an extreme eigenvalue is approximated to full working precision (say 15 decimals) after only 30 Lanczos steps. It is rare that one can solve a linear system correct to 4 decimals after only 30 steps of the conjugate gradient algorithm (which is intimately related to the Lanczos algorithm), unless an excellent preconditioner is used. Thus it seems "easier" to find a few extreme eigenvalues and eigenvectors of A than to solve $Ax = b$!

The codes

All of the programs in Table A except for EA14, stand alone, they do not invoke other packages explicitly. This feature turns out to be attractive to new customers although it is irritating that EISPACK subroutines must be copied into the program instead of being invoked.

The only program which can be obtained, independently of the author, from the Argonne Code Center (now called NESC, the National Energy Software Center) is Scott's LAS02. Documentation is available on an accompanying file. This code has been used extensively at Oak Ridge National Laboratory on a variety of structural analysis problems.

Documentation for the Cullum and Willoughby codes is obtainable in book form (like the EISPACK Guide). Their programs use little or no reorthogonalization of the Lanczos vectors but have developed ingenious ways to identify which eigenvalues of the auxiliary tridiagonal matrix actually belong to the original matrix. Their code is shorter than the rival codes.

The Swedish program uses blocks but does little reorthogonalization. Its chief feature is a sophisticated mechanism for choosing the sifts σ in the spectral transform technique launched by Ruhe and described in Section 8. A careful comparison of LAS02 with STLM would be very interesting for the small band of specialists in matrix eigenvalue computations and would help in progress towards a preferred implementation of the Lanczos algorithm.

The program EA14 is much shorter than the others because it does not use blocks, does not do any reorthogonalization, and finds eigenvalues only, and not even their multiplicity. The user selects the interval to be explored. If the interval happens to be empty the code will report that fact in reasonable time. This is noteworthy because the code assumes that OP delivers A and so triangular factorization is not available. Thus the standard technique for checking

the number of eigenvalues in an interval is not available. Ian Gladwell is incorporating EA14 into a program which also computes eigenvectors, EA15. This will also go into the NAG library.

Table A

Symmetric Problems, Lanczos Based.

<i>Author</i>	<i>Name</i>	<i>Lines</i>	<i>Distributor</i>	<i>Comments</i>
J. Cullum & R. Willoughby	LMAIN LVMAIN	2100	authors (IBM)	$A - \lambda I$ only. no orthogonalization.
J. Cullum & R. Willoughby	BLMAIN	1000	"	$A - \lambda I$ only. block limited reorthog.
D.S. Scott	LAS02	3288	NESC	block. selective orthog.
T. Ericsson & A. Ruhe	STLM	8500	authors (Sweden)	Shift and Invent strategy. (See Section) no orthogonalization.
B.N. Parlett & J. Reid	EA14	648	Harwell	All eigenvalues in a given interval. No eigenvectors. No orthogonalization. $A - \lambda I$ only.

B. Programs not based on Lanczos.

There are a number of long, sophisticated, implementations of subspace iteration buried in Finite Element packages. As such they are beyond the limits of this survey. However the first three programs in Table B are available realizations of subspace iteration.

This paragraph sketches the method. An initial set of orthogonal vectors is chosen. The number of vectors in the set is the dimension of the subspace. Call it p . The proper choice of p is important and difficult. These vectors may be considered as the columns of an $n \times p$ matrix X^0 . At step j the subroutine *OP* is used to compute $Y := (A - \sigma B)^{-1} X^{j-1}$. Next the Rayleigh-Ritz approximations from the column space of Y are computed. These Ritz vectors go into the columns of the matrix X^j . They provide the best orthonormal set of approximate eigenvectors that can be obtained by taking linear combinations of the columns of Y . After a while some of the columns of X^j provide excellent approximations to some eigenvectors of the pair (A, B) . There are a number of variants of this scheme and several clever tricks in the implementation. See [Parlett, 1980, Chap. 14], [Jennings, 1981] or [Stewart, 1976] for more details.

The first three programs in Table B are realizations of subspace iteration. The Achilles heel of the technique is the selection of block size. The first two programs are by numerical analysts and it would be interesting to see how they compare with the codes imbedded in the finite element (FEM) packages mentioned in Section 2. It is likely that they are shorter, cleaner, more robust, and more efficient than their FEM rivals. On the other hand they have not been fine tuned for structural analysis problems and that might make a difference.

The entry TRACMN is the product of recent research. It is based on the fact that

$$\text{trace } [F^k (A - \sigma B) F]$$

is minimized over all $n \times n$ matrices F when, and only when, the columns of F are the eigenvectors of F belonging to the p eigenvalues closest to σ . At each step, the current F is adjusted in order to reduce the trace. We hope that the authors will compare it with LASO2 or some similar rival technique.

LOPSI is the only program offered in Ward's catalog for the nonsymmetric problem. The program has been published in TOMS (a severe test) and employs subspace iteration.

One version of Davidson's method is realized in DIAG and was listed in the catalog of the now defunct National Resource for Computation in Chemistry (LBL, Berkeley). The general reader is warned that Davidson's method has been developed for problems in Quantum Chemistry. The matrix must be strongly diagonally dominant in order for its perturbation technique to be justified. When all the diagonal elements of H are the same then Davidson's method reduces to Lanczos. The difference is as follows. At step j the Rayleigh-Ritz technique is used to produce the best approximation v_j to the fundamental eigenvector that can be obtained as a linear combination of the current basis vectors $b_1, \dots, b_j$. Let $\rho(x)$ be the Rayleigh quotient, $\rho(x) = x^t H x / x^t x$. The residual vector

$$r_j := (H - \rho(v_j))v_j$$

is proportional to the gradient vector $\nabla \rho$ at v_j .

Perturbation theory now says that if v_j is a good approximation then an even better one is

$$a_j := (\rho(v_j) - \text{diag}(F))^{-1} r_j$$

Davidson proposed to take as b_{j+1} the normalized component of a_j orthogonal to $b_1, \dots, b_j$.

We make three observations. 1. The "interaction" matrix or projection $B^t H B$ is not tridiagonal but full. 2. The method aims at a particular eigenvalue/vector pair. 3. When $\text{diag}(H) = h_{11}I$ then a_j is a multiple of τ_j and Lanczos is recovered. This last assertion is not obvious, since $v_j \neq b_j$, but it is true.

Observation 2 shows that Davidson is not really a rival to Lanczos. The methods address different tasks. Moreover Davidson exploits the fact that good starting vectors are available from chemistry.

Table B
Symmetric Problems, other methods.
None of the programs stands alone (TOMS = published in
the Transactions on Mathematical Software of the ACM).

<i>Author</i>	<i>Name</i>	<i>Lines</i>	<i>Distributor</i>	<i>Comments</i>
I. Duff	EA12	427	Harwell (England)	$A - \lambda I$ only. RITZIT inspired
P.J. Nikolai	SIMITZ	550	IMSL author	$A - \lambda B$ RITZIT inspired (in TOMS)
A. Jennings	SI	?	author (Ireland)	$A - \lambda B$ (in Int. Journ. Num. Methods in Engineering)
J.A. Wisniewski A.H. Sameh	TRACMN	586	authors	$A - \lambda B$ minimizes the trace of sections of $A - \lambda B$
A. Jennings & W. Stewart	LOPSI	?	authors (Ireland)	Subspace Iteration for NONSYMMETRIC PROBLEM. (in TOMS)
E.R. Davidson	DIAG	?	author (Univ. of Wash. Dept. of Chem.)	perturbation technique $A - \lambda I$

7. LANCZOS VERSUS SUBSPACE ITERATION

These two rival techniques are good and address the same task. A comparison of them is given in [Parlett, Nour-Omid, Taylor 1983] and the conclusion there is that a modern, properly used version of the Lanczos algorithm should be an order of magnitude (base 10) faster than a good subspace iteration program on problems with $n > 500$. The larger the number of modes wanted the greater the advantage of the Lanczos algorithm.

The fundamental theoretical advantage of the Lanczos is that it never discards any information whereas subspace iteration, at each step, overwrites a set of approximate eigenvectors with a better set. What is surprising is that the simple Lanczos algorithm needs only 5 or 6 n -vectors in the fast store at each step. Moreover, it can find multiple eigenvalues if selective orthogonalization is used.

There are enough poor implementations of Lanczos available to complicate comparisons. The engineers who have devoted themselves to steady improvements of subspace iteration are loyally defending the virtues of their codes. It will be interesting to see what happens.

8. WHAT HAVE WE LEARNT SINCE 1978?

Much could be said in answer to this question but two items suggest themselves.

1. The importance to new users of stand alone programs. This point was made in Section 5.

2. The Spectral Transformation. The Lanczos algorithm requires that a general linear problem

$$[K - \lambda M]x = 0$$

be reduced to standard form. There are several ways to accomplish this and it

is vital, for large problems, to choose a good one. Although the facts given below are elementary and have been known to the specialists in matrix computations for a long time it is fair to say that until A. Ruhe pointed out their implications in [Ericsson and Ruhe 1980] they were not given the proper emphasis. Of course, subspace iteration has employed spectral transformations from the earliest days.

Ruhe proposed that the original problem be rewritten in the standard form

$$[M^H(K - \sigma M)^{-1} M^H - \nu I]y = 0$$

with $y = M^H x$, and σ a shift into the interior of the interval which is to be searched for eigenvalues λ . This is quite different from the usual recommendation

$$[L^{-1}KL^{-1} - \lambda I]z = 0$$

where $M = LL^T$ and $z = L^T x$.

In the majority of large structural problems (with displacement formulation) M is singular as well as diagonal, perhaps 1/3 of its diagonal elements are zero. The shift σ is not fixed precisely so that there is no loss in assuming that $K - \sigma M$ is nonsingular and can be factored in a stable manner without any row and column interchanges. If $K - \sigma M = LDL^T$ then the product $y = M^H(K - \sigma M)^{-1} M^H u$ is formed in stages:

$$v \leftarrow M^H u, \text{ solve } L w = v, \text{ solve } DL^T x = w, y \leftarrow M^H x.$$

If the factorization of $K - \sigma M$ is too costly then in principle $(K - \sigma M)y = M^H u$ could be solved by an iterative method. The point is that the subprogram OP in Lanczos should return $M^H(K - \sigma M)^{-1} M^H u$ when given u . Sparse techniques can be used in factoring $K - \sigma M$. This reduction of the problem transforms the spectrum according to

$$\nu_i = \frac{1}{\lambda_i - \sigma}$$

and so the eigenvalues λ_i closest to σ turn into the eigenvalues ν_i closest to ∞ . These are the ones which will appear first in a run of Lanczos algorithm. For large n the reduction in the number of steps is so satisfactory that it will offset the cost of factorization. Without the spectral transformation it will often be necessary to take $\frac{1}{2}n$ or more steps of the Lanczos algorithm in order to approximate the smallest 10 eigenvalues. Yet the Lanczos algorithm is most effective when used with fewer than 200 steps.

9. WORK IN PROGRESS

It is surprising, at first, that the software reviewed above is so narrowly focussed. So it is natural that a few investigators are working to "fill the gaps", to provide robust programs for all the requests that might conceivably be made for eigenvalues of various types of large matrix.

Normal matrices enjoy a full set of orthonormal eigenvectors and most methods designed for real symmetric matrices extend naturally to normal ones. The most important subclass of normal but asymmetric matrices are those that are skew and Ward has developed efficient software for them, as described in [Ward, 1978].

Work on nonnormal sparse matrices is still in the experimental stage. The experts are exploring the problem and there is no consensus on a preferred technique. Y. Saad has adapted Arnoldi's method for sparse problems. This is the natural extension of the Lanczos idea but subject to the constraint of using orthonormal bases for the associated Krylov subspaces. The auxiliary matrix generated in the course of the algorithm is Hessenberg (i.e. $h_{ij} = 0$ if $i > j + 1$) rather than triadiagonal. This is not a serious feature provided that fairly short runs of Lanczos are used, say 50 steps at most. Saad is experimenting with

variations on this method in which orthogonality is given up in return for a banded structure in the Hessenberg matrix, see [Saad, 1980].

The Lanczos algorithm was generalized to the nonnormal case by Lanczos himself, the procedure is obvious. Unfortunately it can breakdown and, in fact, does so more often than not. In [Parlett and Taylor, 1981] it is shown how to restore a large measure of stability in return for modest increase in complexity. Some feature of the original, strict Lanczos algorithm must be discarded if the breakdown is to be avoided. The new method looks ahead at every step and decides whether to enlarge the Krylov subspace by one or two dimensions. The auxiliary matrix is not quite triangular, there is a bulge every time the dimension increases by two. Column and row eigenvectors are given equal status and condition numbers are automatically computed.

Axel Ruhe is experimenting with a two-sided Arnoldi process but this work is at an early stage.

What impedes the production of pleasing software for nonnormal problems is the potential ill-condition of the problem itself. Expectations have been set by the symmetric case where the theory is most satisfactory. In the general case it is difficult to generate useful error estimates, let alone error bounds, and this affects the selection of stopping criterion. Numerical analysts would want to deliver condition numbers along with the eigenvalues, but users do not want this information, especially if it increases the cost by a noticeable amount.

We terminate this digression with a reminder that Jennings' program LOPSI has been published and is available to the public. LOPSI adapts subspace iteration to the asymmetric case.

The most promising developments are close to the symmetric case. Many problems in dynamics arise in the form

$$M\ddot{x} + C\dot{x} + Kx = f$$

where $\dot{x}$ is the time derivative of the vector field $x(t)$. The damping matrix C is such a pain that it is usually ignored. The program [Scott & Ward, 1982] looks to the future and presents software for the associated quadratic eigenvalue problem

$$(\lambda^2 M + \lambda C + K)u = 0 .$$

Their method is based on the Rayleigh quotient iteration. It does not take the easy way out and reduce the quadratic problem to a linear one of twice the size.

The Lanczos programs in Ward's catalog work very well in combination with the spectral transformation discussed in Section 8. That approach is not possible when the user cannot, or will not, allow the solution of linear systems of the form

$$(K - \sigma M)v = \omega .$$

What can be done?

Scott observed, in [Scott, 1981], that when σ is close to an eigenvalue $\sigma + \delta$ of the pair (K, M) then the vector x belonging to the eigenvalue δ closest to zero for the standard problem

$$(K - \sigma M - \lambda I)x = 0$$

is an approximate eigenvector of (K, M) belonging to σ . He formulates an iterative method in which a standard sparse eigenvalue problem is solved at each step for (δ, x) and $\sigma + \delta$ converges monotonically to an eigenvalue of (K, M) .

10. CONCLUSION

The eigenvalue problem for large, sparse matrices is not yet understood well enough, in all its computational ramifications, to permit the rapid deployment of impeccable software to the masses in the style set by EISPACK. Indeed

it is doubtful that the public is impatient for the arrival of this facility. Those with an urgent need for such software prize efficiency above generality. They have developed their own programs independently of the numerical analysts and will continue to do so unless the matrix experts go to them and demonstrate that they can be useful.

The wide variety of computing environments is going to play havoc with naive concepts of portability. "Transmission of thought" is one of the two basic themes of science. The interesting and difficult task is to find the appropriate level at which to transmit.



REFERENCES

- E.R. Davidson, (1975), "The Iterative Calculation of a Few of the Lowest Eigenvalues and Corresponding Eigenvectors of Large Real-Symmetric Matrices", *Jour. Comp. Phys.* 17, pp. 87 - 94.
- E.R. Davidson, (1983), "Matrix Eigenvector Methods " to appear in the Proceedings of a NATO Advanced Study Institute on methods in Computational Molecular Physics.
- I. Duff (1983), "A Survey of Sparse Matrix Software", in *Sources and Development of Mathematical Software*, ed. W.R. Cowell, Prentice Hall Inc.
- T. Ericsson and A. Ruhe, (1980), "The Spectral Transformation Lanczos Method for the Numerical Solution of Large Sparse Generalized Symmetric Eigenvalue Problems", *Math. Comp.* 35, pp. 1251-1268.
- A. Jennings (1981), "Eigenvalue Methods and the Analysis of Structural Vibration", in *Sparse Matrices and Their Uses*, ed. I. Duff, Acad. Press, pp. 109-138.
- P.J. Nikolai (1979), "Algorithm 538. Eigenvectors and Eigenvalues of Real Generalized Symmetric Matrices By Simultaneous Iteration", *ACM Trans. Math. Softw.* 5, pp. 118-125.
- B.N. Parlett (1980), *The Symmetric Eigenvalue Problem*, (Prentice Hall, New Jersey).
- B.N. Parlett, B. Nour-Omid, and R.L. Taylor, "Lanczos Versus Subspace Iteration for Solution of Eigenvalue Problems", to appear in *Int. Jour. for Num. Meth.* in Aug., (1983).
- B.N. Parlett and D. Taylor (1981), Lookahead Lanczos Algorithm for Nonnormal Matrices, Report No. 43, Center for Pure and Applied Mathematics
- Y. Saad (1980), "Variations on Arnoldi's Method for Computing Eigenelements of Large Symmetric Matrices", *Lin. Alg. Appl.* 34, pp. 289-295.
- Problems Without Factorization", *SIAM J. Sci. Stat. Comp.* X, xxx-xxx. P. Saxe, D.J. Fox, H.F. Schaefer III, and N.C. Handy (1983), "The Shape-Driven Graphical Unitary Group Approach to the Election Correlation Problem. Application to the Ethylene Molecule", to appear.
- D.S. Scott (1981), "Solving Sparse Symmetric Generalized Eigenvalue Problems Without Factorization", *SIAM J. Num. Anal.* 19, pp. 102-110.
- D.S. Scott and R.C. Ward (1982), "Solving Quadratic λ -matrix Problems Without factorization", *SIAM J. Sci. Stat. Comp.* X, xxx-xxx.
- I. Shavitta (1977), "The Method of Configuration Interaction" in *Methods of Electronic Structure Theory*, ed. H.F. Schaeffer III Plenum Publ. Corp.
- G.W. Stewart (1976), "A Bibliographic Tour of the Large Sparse Generalized Eigenvalue Problem", in *Sparse Matrix Computations*, eds. J.R. Bunch and D.J.

Rose, Academic Press. pp. 113-130.

W.J. Stewart and A. Jennings (1981), "Algorithm 510. LOPSI: A Simultaneous Iteration Algorithm for Real Matrices," ACM Trans. Math. Softw. 7, pp. 230-232.

R.C. Ward and L.J. Gray (1978), "Eigensystem Computation for Skew-symmetric Matrices and a Class of Symmetric Matrices", Trans. on Math. Software 4, pp. 278-285.

