JOURNAL OF COMPUTATIONAL PHYSICS 69, 471–481 (1987)

# The Solution of Large Dense Generalized Eigenvalue Problems on the Cray X-MP/24 with SSD*

ROGER GRIMES

*Boeing Computer Services, M/S 7L-21, Engineering Technology Applications Division, P.O. Box 24346, Seattle, Washington 98124*

HENRY KRAKAUER

*Department of Physics, The College of William and Mary, Williamsburg, Virginia 23185*

JOHN LEWIS AND HORST SIMON

*Boeing Computer Services, M/S 7L-21, Engineering Technology Applications Division, P.O. Box 24346, Seattle, Washington 98124*

AND

SU-HAI WEI†

*Department of Physics, The College of William and Mary, Williamsburg, Virginia 23185*

Received: January 3, 1986; revised: May 9, 1986

Quantum mechanical bandstructure calculations often require the numerical solution of large, full generalized eigenvalue problems. The eigenvalue problems are definite and often symmetric, and in most cases only a few eigenvalues and eigenvectors are required. However, with a problem size ranging between 1000 and 2000 an in-core solution of these problems is still memory-limited on most of today's supercomputers. Here we investigate the use of the Lanczos algorithm for this type of eigenvalue problem. Numerical results demonstrate some of the advantages of the Lanczos algorithm. The Lanczos approach is generally faster and allows an out-of-core implementation for extremely large problems. 

## 1. INTRODUCTION

Quantum mechanical bandstructure calculations often require the numerical solution of large, full, generalized eigenvalue problems. The eigenvalue problems are definite and in many cases also symmetric of the form

$$Ax = \lambda Bx. \tag{1}$$

* This work is supported in part by NSF Grant DMR-84-16046 and a supercomputer grant of the Office of Advanced Scientific Computing at the NSF.

† Present address: Solar Energy Research Institute, Golden, Colorado 80401.



0021-9991/87 $3.00

Here $A$ and $B$ are real, symmetric $n$ by $n$ matrices and $B$ is positive definite. The definiteness of $B$ assures that the spectrum is real.

These eigenvalue problems are encountered after applying a Rayleigh–Ritz variational method (see, e.g., [8]) to solve the Schrödinger differential equations for the quantum mechanical wave function. For example, in [14] Wei and Krakauer report theoretical calculations of the pressure and volume at metallization of BaTe using such an approach. These calculations accurately reproduce recent experimental measurements of this transition [6, 7]. These calculations represent the first convincing demonstration of the ability of this type of calculations to predict band overlap metallization. The variational approach to obtaining the wave function is standard in quantum chemistry and solid state physics.

The efficient numerical solution of (1) is therefore of great interest to the community of computational physicists and chemists. In this report we discuss both the application of the Lanczos algorithm and a combination of the Householder reduction and bisection algorithm for (1). A discussion of both solution approaches as well as further references are given by Parlett [9]. A vectorized implementation of the Lanczos algorithm on the Cyber 205 and its application to large, sparse eigenvalue problems in structural engineering has been discussed by Parlett, Nour-Omid, and Natvig [10].

## 2. Properties of the Eigenvalue Problem

One main characteristic of the eigenvalue problem (1) is that typically only about 10% of the eigenvalues and corresponding eigenvectors are wanted. The desired eigenvalues are at the lower end of the spectrum. All negative eigenvalues and a few positive ones are needed for further calculations. Even though the number of required eigenvalues remains relatively constant, the problem size does not. There are some a priori rules of thumb dictating how many basis functions should be used in the Rayleigh–Ritz method, but the problem size may have to be increased for a given simulation. For the purpose of the discussion in this report, we assume a problem size of around 1000, which is fairly typical. Our largest numerical result is for $n = 1496$, and it is not unreasonable to consider calculations in the near future which would lead to problems of order 2000 to 3000.

Computing some eigenvalues of a 1000 by 1000 matrix is definitely a supercomputer problem, in particular in the context considered here, where one simulation run may require the solution of several hundred of these eigenvalue problems. The EISPACK 2 manual [4] gives as a benchmark for the solution of an 80 by 80 eigenvalue problem on the IBM 370/195 an execution time of 1.16 s. Based on this benchmark, and using the formulas given in [4] we estimate an execution time of about 18 min for our 1000 by 1000 matrix. Solving 200 such problems in a single calculation would require about 60 h. The results in this report show that the same computation can be carried out on a Cray X-MP in about 1 h. Hence an unfeasible

TABLE I

Approximately Largest Feasible Problem Size
for In-Core Solution Using EISPACK Routines

| Machine | Memory size | RSG | BISECT-path |
|---|---|---|---|
| Cray-1S | 2 Mword | 753 | 912 |
| Cray X-MP/24 | 4 Mword | 1110 | 1350 |

computational problem has been turned into a production type problem, which can be solved easily in an overnight run.

These large dense eigenvalue problems are, however, memory limited even on supercomputers such as the CRAY X-MP. The standard EISPACK driver RSG for the symmetric generalized eigenvalue problem requires about $3n^2$ memory locations for the computation of the complete spectrum. Combining some special EISPACK routines it is possible to compute only the $p$ eigenvalues and eigenvectors of interest. For this EISPACK path, using the subroutine BISECT for finding the eigenvalues of the tridiagonal matrix, and TINVIT for computing eigenvectors by inverse iteration, the storage requirements are $2n^2 + pn$ words. Neither of the two solution algorithms, however, can compute an in-core solution for the largest problems of interest. The largest problems solvable by the two approaches are given in Table I.

Since we were interested in solving problems up to $n = 2000$, some alternative options had to be investigated. The following approaches appeared to be feasible.

(1) The use of packed symmetric storage would reduce storage requirements by about one half. This approach is discussed in Section 3.

(2) The use of the Lanczos algorithm would reduce the storage requirements by about the same amount. In addition the Lanczos algorithm allows for an easy out-of-core solution of the eigenvalue problem, with the potential of using the solid-state-storage device on the CRAY X-MP. This approach is discussed in Section 4.

(3) A last option would be to develop an out-of-core version of the Householder reduction to tridiagonal form, and use it in connection with other out-of-core routines to develop an out-of-core version of EISPACK. This option is currently being investigated by some of the authors. Some remarks on an out-of-core EISPACK, and conclusions from our numerical results are given in Section 5.

## 3. Packed Symmetric Storage

Packed symmetric storage refers to a mode of storing the rows of the lower triangular part of the matrix one after the other in a one-dimensional array of length $n(n+1)/2$. This is equivalent to storing the columns of the upper triangular

part in the same fashion. The total storage requirement for computing $p$ eigenvalues and vectors is then approximately $n^2 + pn$. For $p = 30$ the largest feasible problem on the Cray 1-S is about of order 1304, and on the Cray X-MP/24 of order 1923. Hence packed symmetric storage allows us to perform the calculations we are currently interested in.

Unfortunately not all required subroutines for the generalized eigenvalue problem are implemented in EISPACK using packed symmetric storage. EISPACK only provides routines for the symmetric packed problem, but no subroutines which would reduce a generalized problem in packed form to a simple eigenvalue problem in packed form. In order to work only with packed matrices throughout the solution process, it was necessary to utilize some subroutines from LINPACK and to perform some additional coding.

The first step in solving (1) is a Choleski factorization of the positive definite matrix $B$,

$$B = LL^T. \tag{2}$$

Here we use the packed factorization subroutine SPPFA from LINPACK [2]. SPPFA computes the Choleski factor $L$ and overwrites $B$. A vectorized version of SPPFA has been used here.

The next step is the formation of the matrix $\tilde{A}$,

$$\tilde{A} = L^{-1}AL^{-T}. \tag{3}$$

This operation is not implemented in EISPACK. In our current program it has been newly implemented so that the lower triangle of $\tilde{A}$ overwrites the lower triangle of $A$. All operations can also be arranged so that the Cray Fortran compiler CFT vectorizes the inner loops. Because of the packed storage scheme this is not completely trivial.

The matrix $\tilde{A}$ is then reduced to tridiagonal form $T$ using the subroutine TRED3 from EISPACK. TRED3 is a version of the Householder reduction to tridiagonal form, where the coefficient matrix is given in symmetric packed form. Because of better vectorization on the Cray X-MP, the version of TRED3 from the latest EISPACK release [3] was chosen.

TABLE II

Execution Times for Packed EISPACK Routines

| Order | Execution time (s) |
|---|---|
| 219 | 1.370 |
| 667 | 14.729 |
| 992 | 37.330 |
| 1496 | 108.543 |

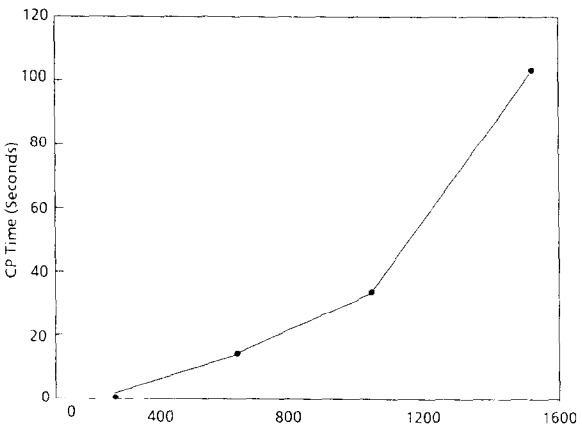


FIG. 1. Execution times for packed EISPACK routines.

The subroutine BISECT from EISPACK is used to compute all the eigenvalues of $T$ in a given interval. Then TINVIT, TRBAK3, and a triangular solve with $L^T$ must be applied to compute the corresponding eigenvectors of the $T$, transform them back to eigenvectors of $\tilde{A}$, and finally transform them to eigenvectors of (1). Operations in BISECT and TINVIT are essentially scalar and do not vectorize. These operations, however, only account for a small fraction of the total computational work. All eigenvector transformations vectorize well.

As an example for the computational performance of the packed EISPACK routines we consider a sequence of problems arising in surface calculations for tungsten. The same model was solved using an increasing number of basis functions. In all cases the computation of all eigenvalues in the interval $[-1.0, 0.30]$ was requested. All numerical tests were performed on the Boeing Computer Sevices Cray X-MP/24 using CFT 1.11. The execution times are given in Table II (see also Fig. 1).

## 4. LANCZOS ALGORITHM

In recent years the Lanczos algorithm has become the method of choice for large sparse generalized eigenvalue problems arising in structural engineering (dynamic analysis). A block shifted Lanczos algorithm for dynamic analysis problems has been recently implemented by Grimes, Lewis, and Simon [5]. The report [5] discusses the details of the implementation of this algorithm. For a discussion of the basic issues involving the practical use of the Lanczos algorithm for eigenvalue computations see [9]. The vectorization of the Lanczos algorithm and its performance on vector computers is discussed in [10].

Here we will only discuss some aspects of the block-shifted Lanczos algorihm, as

they are relevant for the understanding of the numerical results. The simple, unshifted Lanczos algorithm for the symmetric eigenvalue problem is related to the Householder reduction to tridiagonal form in the sense that it will also produce a tridiagonal matrix, which is similar to the original matrix. The main difference is, however, that the Lanczos algorithm can be terminated early, and nevertheless the truncated tridiagonal matrix will contain eigenvalue approximations to some of the eigenvalues of the original matrix. A second difference is that the Lanczos algorithm does not modify the original matrix directly. The only information needed is the computation of a matrix vector product for a given vector. This is quite easily accomplished, whether the matrices involved are sparse or dense, or whether they are stored in-core or out-of-core.

The basic Lanczos algorithm must be modified in several ways to handle generalized eigenvalue problems of the form (1). One modification is to switch to a block Lanczos algorithm. A block Lanczos algorithm operates on several vectors at the same time, and reduces the original matrix to block tridiagonal form. Even though convergence becomes slower for larger bocksizes, a block algorithm can become overall more efficient in terms of execution time, since the total number of accesses to the matrix and hence the number of I/O operations is reduced. A second reason for choosing a block algorithm is its better performance for multiple eigenvalues.

Another modification is the use of a shifted Lanczos algorithm. The shifted Lanczos algorithm is related to inverse iteration in the sense that it converges most rapidly to eigenvalues which are close to the chosen shift value. Unlike inverse iteration, which only computes one eigenvalue and vector at a time, the Lanczos algorithm will compute approximations to many eigenvalues near the shift. The shifted Lanczos algorithm, however, requires a factorization of the original matrix initially, and forward solve and back substitution operation with the factored matrix at each step instead of the matrix multiply. If the matrix is stored out-of-core, obviously corresponding out-of-core routines must be used.

The implementation of the block-shifted Lanczos algorithm [5] also employs an automatic shifting strategy. Based on a heuristic, user specified intervals are searched for eigenvalues by judiciously placing shifts. The use of inertia counts (see [9, 5]) assures that no eigenvalues will be missed.

The block-shifted Lanczos algorithm [5] has been developed originally for sparse problems arising in structural engineering applications. However, its modular structure made it easy to make the necessary modifications for a dense problem. All interaction between the coefficient matrices $A$ and $B$ and the code took place in only three subroutines. For these subroutines the corresponding LINPACK [2] subroutines for packed symmetric matrices were substituted. This involved subroutines for the symmetric packed factorization with inertia count (SSPCO) and the corresponding solve routine SSPSL. These routines had been vectorized previously. Since the LINPACK routines are in-core subroutines the total storage for the Lanczos algorithm was thus about the same as for the EISPACK implementation discussed above.

One advantage of the Lanczos algorithm is that potentially an out-of-core version is much easier to produce. It will only require out-of-core versions of the above LINPACK routines. Contrary to the situation with the Householder reduction, it is well understood how to solve linear systems out-of-core [13], and a complete out-of-core version of the Lanczos code can be developed easily.

A second advantage of the Lanczos algorithm is that the majority of operations involve long vectors of size $n$. The algorithm thus vectorizes quite naturally. This has been discussed previously in [10].

The current version of the Lanczos code keeps the coefficient matrices $A$ and $B$ in core. Only certain intermediate quantities produced by the Lanczos code are stored out-of-core. The Boeing Cray X-MP allows storing these internal files on the solid-state storage device (SSD). The SSD is a very fast secondary storage system, which

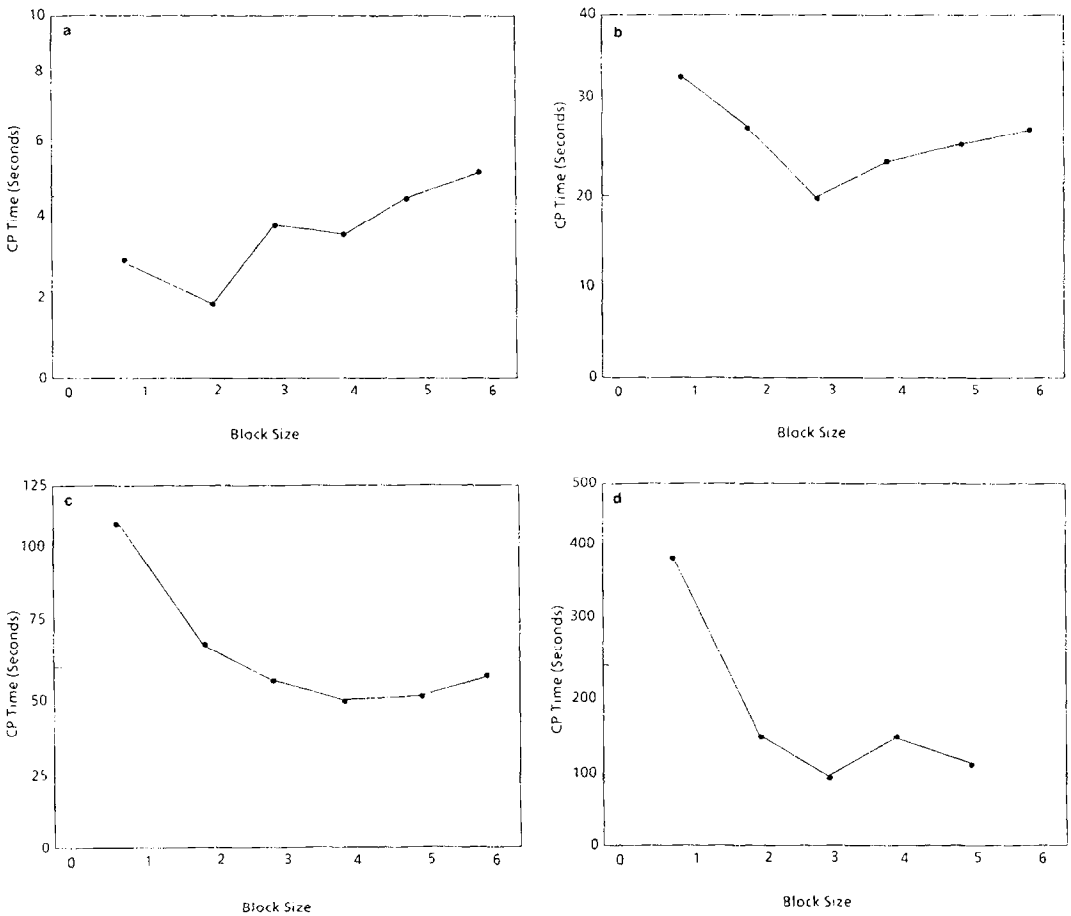


FIG. 2. Execution times for Lanczos code with SSD, tungsten surface, CP time: (a) order 219; (b) order 667; (c) order 992; (d) order 1496.

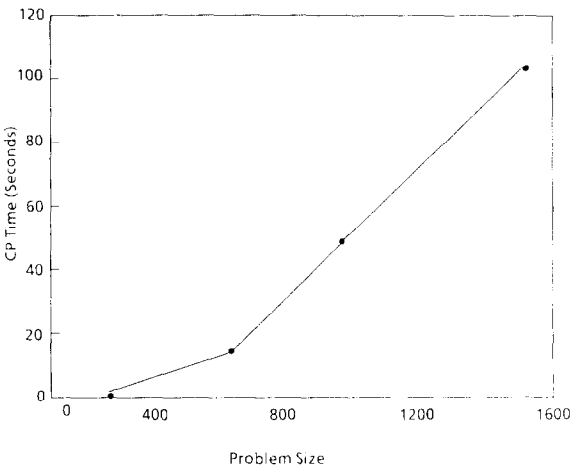


FIG. 3. Execution times for Lanczos code with SSD (optimal block sizes from results in Fig. 2).

when used optimally can sustain transfer rates only moderately slower than access to fast in-core memory [1].

In an initial test of the block-shifted Lanczos algorithm the four tungsten surface calculation problems were solved, with varying blocksize, and by storing intermediate quantities both on the SSD and on regular disks. The execution time with or without SSD did not vary significantly in most of the cases. Where they did differ, this could be attributed to the fact that a minor variation in the execution time caused the shift heuristic to make some different decisions. This insignificant difference can be explained with the fact that the amount of data transfered to and from the SSD in a single I/O request is comparatively small. The fast asymptotic transfer rate of the SSD does not come into play. We report here only the results with SSD.

A first version of the algorithm was required to compute the 30 smallest eigenvalues. The results of the first set of tests are summarized in Fig. 2. Several conclusions can be drawn from these results. First, blocksize three appears to be the

TABLE III

Execution Times for Block Shifted Lanczos

| Order | Execution time (s) |
|---|---|
| 219 | 1.48 |
| 667 | 6.49 |
| 992 | 14.62 |
| 1496 | 34.66 |

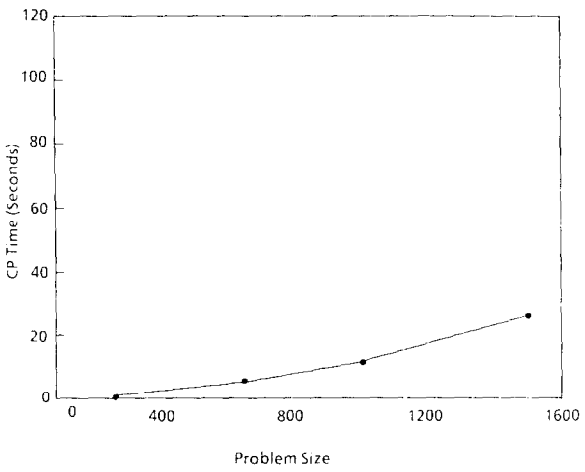


FIG. 4. Execution times for Lanczos code.

optimal choice for these dense problems. If the optimal execution times (see Fig. 3) are compared with the previous EISPACK results, they are slower by up to a factor of two. This is again due to the shifting strategy. This first implementation of the shifting heuristic is very conservative and employs more shifts and hence factorizations to ascertain that no eigenvalues have been missed. The Lanczos code actually solves a more difficult problem than the EISPACK code since it has no a priori information on the location of the eigenvalues.

Because of some inefficiencies the shifting strategy has been reformulated in a second version of the Lanczos code. This new version of the code requires for each problem only two or three factorizations and the results obtained (for blocksize three) are listed in Table III. The execution times are summarized in Fig. 4. Note that the Lanczos algorithm needs no priori knowledge of the interval.

## 5. CONCLUSIONS

In summary the performance of the optimal Lanczos code is considerably better than the EISPACK routines (see Fig. 5). However, there is a considerable performance variation within the Lanczos code depending on the choice of computational task and parameters. Besides a good performance of the Lanczos algorithm depends also on a good implementation of the shifting heuristic. Nevertheless the Lanczos algorithm appears to be the method of choice for the computation of some eigenvalues and vectors of large dense problems, especially when considering the fact that a full out-of-core version can be developed fairly easily. This conclusion confirms some theoretical results by Paige [11], who estimated that the Lanczos

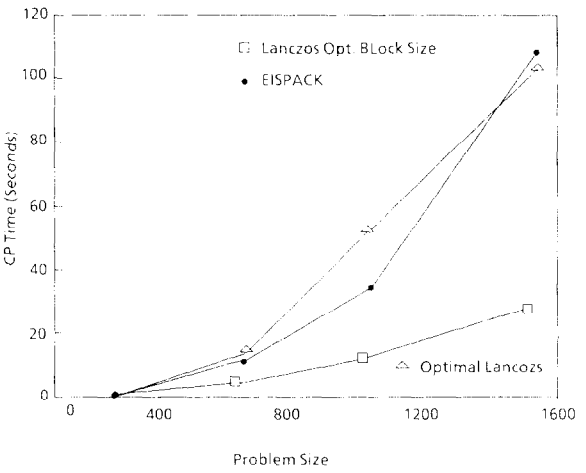


FIG. 5. Summary of execution times (overlay of Figs. 1, 3, and 4).

algorithm would be more efficient than Householder reduction and bisection, if less than a fourth of all the eigenvalues are required.

Will the whole question of out-of-core solutions for eigenvalue problems still be an issue when the next generation of supercomputers such as the CRAY 2 are considered? These machines provide a very large fast memory (up to 256 Mwords on the CRAY 2) and the calculations discussed in this report could be performed easily. The experience of the last decade has, however, shown that scientific problems under investigation grow at about the same rate as the computational power of supercomputers. Hence we believe that the solution of large dense eigenvalue problems in a memory limited environment will remain an interesting research question in scientific computing.

## REFERENCES

1. CRAY RESEARCH, INC., "Solid-state Storage Device Reference Manual," Publication HR-0032, 1984.
2. J. J. DONGARRA, C. B. MOLER, J. R. BUNCH, AND G. W. STEWART, *LINPACK User's Guide* (Soc. Indus. Appl. Math., Philadelphia, 1979).
3. J. J. DONGARRA AND C. B. MOLER, Report ANL/MCS-TM-12, Argonne National Laboratory, 1983 (unpublished).
4. B. S. GARBOW *et al.*, *Matrix Eigensystem Routines—EISPACK Guide Extension*, (Lecture Notes in Comput. Sci. No. 57, Springer-Verlag, Berlin/Heidelberg/New York, 1977).
5. R. GRIMES, J. LEWIS, AND H. SIMON, Report ETA-TR-39, Boeing Computer Services, 1986 (unpublished).
6. T. A. GRZYBOWSKI AND A. L. RUOFF, *Phys. Rev. B* **27**, 6502 (1983).
7. T. A. GRZYBOWSKI AND A. L. RUOFF, *Phys. Rev. Lett.* **53**, 489 (1984).
8. J. K. L. MACDONALD, *Phys. Rev.* **43**, 830 (1933).
9. B. PARLETT, *The Symmetric Eigenvalue Problem* (Prentice-Hall, Englewood Cliffs, N.J., 1980).

10. B. PARLETT, B. NOUR-OMID, AND J. NATVIG, in *Supercomputer Applications*, edited by R. W. Numrich (Plenum, New York, 1985).
11. C. C. PAIGE, Thesis, Univ. of London, 1971 (unpublished).
12. B. T. SMITH *et al.*, *Matrix Eigensystem Routines—EISPACK Guide* (Lecture Notes in Comput. Sci. No. 6, Springer-Verlag, Berlin/Heidelberg/New York, 1974).
13. "VectorPak Users Manual," Boeing Computer Services Document No. 20460-0501, 1985.
14. S. H. WEI AND H. KRAKAUER, *Phys. Rev. Lett.* **55**, 1200 (1985).