Conf-950439--11

PNL-SA-25658

PERFORMANCE OF A FULLY PARALLEL DENSE REAL SYMMETRIC EIGENSOLVER IN QUANTUM CHEMISTRY APPLICATIONS

G. I. Fann
R. J. Littlefield
D. M. Elwood

April 1995

Presented at the
High Performance Computing Symposium, Simulation
MultiConference 95
April 9-13, 1995
Phoenix, Arizona

Prepared for
the U.S. Department of Energy
under Contract DE-AC06-76RLO 1830

Pacific Northwest Laboratory
Richland, Washington 99352

DISCLAIMER

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

PERFORMANCE OF A FULLY PARALLEL DENSE REAL SYMMETRIC EIGENSOLVER IN QUANTUM CHEMISTRY APPLICATIONS

George I. Fann, Richard J. Littlefield and David M. Elwood
High Performance Computing Group
Environmental and Molecular Science Laboratories
Pacific Northwest Laboratory[1], Richland, WA
e-mail: gi_fann@pnl.gov

key words: symmetric eigensolvers, diagonalizer, parallel linear algebra

ABSTRACT

The parallel performance of a dense, standard and generalized, real, symmetric eigensolver based on bisection for eigenvalues and repeated inverse iteration and reorthogonalization for eigenvectors is described. The performance of this solver, called PeIGS, is given for two test problems and for three "real-world" quantum chemistry applications: SCF-Hartree-Fock, density functional theory and Møller-Plesset theory. The distinguishing feature of the repeated inverse iteration and orthogonalization method used by PeIGS is that orthogonalization may be performed across multiple processors as dictated by the spectrum. For each problem we describe the spectrum and the clustering of the eigenvalues, the most important factor in determining the execution time. For a spectrum that is well spaced, there is essentially no orthogonalization time. Most of the time is consumed in the Householder reduction to tridiagonal form. For large clusters, almost all of the time is consumed in the Householder reduction and in orthogonalization. Performance results from the Intel Paragon[2], and Kendall Square Research KSR-2 are reported.

1 INTRODUCTION

Bisection and inverse iteration is known to be a fast method for computing eigenvalues and eigenvectors of real symmetric tridiagonal matrices. In this paper we describe the performance of PeIGS, a parallel real symmetric and generalized symmetric eigensolver based on bisection for eigenvalues and repeated inverse iteration and orthogonalization for eigenvectors (Fann-Littlefield 1993). The performance of this solver for two test matrices with extreme clustering characteristics are presented. In addition, we present performance, spectrum and clustering characteristics for three quantum chemistry applications: SCF Hartree-Fock, density functional theory and Møller-Plesset theory.

PeIGS uses a flexible column distribution with packed storage for the real symmetric matrices. The user specifies the processor which stores each column. Basic linear algebra operations such as triangular matrix multiply are performed in a panel-block systolic fashion similar to that described in (Demmel et al. 1993, algorithm 4, p.16). Although this data distribution and algorithm does not scale as well as block submatrix distributions, we still obtain reasonable speedups. For the Hartree-Fock ZSM-5 quantum chemistry example, we obtain a speedup of 78 using 185 processors of the Intel Paragon and reduce the computing time from 2.5 hours to under 2 minutes (c.f. figure 12). This speedup suggests that useful speedups can be obtained in production runs. This method may perform even better for block matrix distributions.

Many parallel eigensolver algorithms use multi-section for isolating eigenvalues in a given interval and then use bisection to extract the eigenvalues. The eigenvectors are then computed using inverse iteration. Usually the currently iterated eigenvector is orthogonalized against the previously converged eigenvectors in a cluster (e.g., (Ipsen and Jessup 1990), (Jessup and Ipsen 1992), (Lo et al. 1987) to name just a few).

Our contribution is a more parallel method of extracting eigenvectors that must be orthonormalized because of clustered eigenvalues. The eigenvectors are computed using repeated inverse iteration and reorthogonalization (Fann-Littlefield 1993). This has been shown empirically to yield eigenvectors in time comparable to, and with residual and orthogonality of

[1]Pacific Northwest Laboratory is operated for the U.S. Department of Energy (DOE) by Battelle Memorial Institute under contract DE-AC06-76RLO 1830.

[2]Thanks to California Institute of Technology's CSCC for providing access to the Intel Paragon.

the same magnitude as LAPACK's `dstein` (Anderson et al. 1992).

In this paper we present empirical evidence that PeIGS scales well and performs well in real-world applications. We also present detailed information showing that Householder reduction and eigenvector extraction dominate the computation when large clusters are present, while Householder reduction dominates the computation when all of the clusters are small.

This paper is organized as follows. In Section 2, we describe the algorithms and the implementation strategy. In Section 3, we present the performance of PeIGS for various cluster sizes and for some chemistry applications. Finally, in Section 4, we summarize and present our conclusions.

2 ALGORITHM

PeIGS allows subsets of the allocated processors to perform the eigensystem computations. The processors not storing any of the eigenvectors or matrix elements are free to do other work. In the following all processors are assumed to be participating in the eigensystem computation.

2.1 Reduction

Let A and B be real symmetric matrices of dimension n. Assume that B is positive definite. The generalized eigenproblem $Ax = \lambda Bx$ is reduced to the standard eigenproblem $Cx = \lambda x$ by Choleski factorizing the symmetric positive definite matrix $B = LL^t$ and then setting $C = L^{-1}AL^{-t}$. The parallel Choleski factorization is performed using a submatrix algorithm (Heath 1986). Instead of doing two triangular solves with L to form the symmetric matrix C, we invert the lower triangular matrix L and perform a lower triangular and an upper triangular matrix multiply to form C. This approach is slightly less accurate than the standard method (Wilkinson 1965, pages 339-340) and those used in LAPACK (Anderson et al. 1992), but it exhibits more parallelism and less data dependence for large processor counts on distributed memory computers. In practice, we have not experienced any numerical problems as both methods of forming C are stable (Wilkinson, 1965, pages 339-340).

2.2 Standard Eigenproblem

The symmetric matrix C is reduced to tridiagonal form using Householder reduction, essentially a parallel version of EISPACK's `tred2` (Smith et al. 1976) (Chinchalkar and Coleman 1991) (Littlefield and Maschhoff 1993). Processor 0 then broadcasts the diagonal and subdiagonal of the tridiagonal matrix to all of the participating processors. Each processor then computes a set of eigenvalues using LAPACK's `dstebz`. For example, with p processors and $n = mp$, processor i computes eigenvalues $im + 1$ through $(i + 1)m$, where the eigenvalues are ordered in an increasing fashion. If the sets overlap, because of repeated or very close eigenvalues, then PeIGS performs redundant computation of the eigenvalues in a small interval containing the eigenvalues where the sets intersect.

Once the eigenvalue computation is finished, processor 0 collects all the eigenvalues and their corresponding block numbers and broadcasts this information to all of the processors participating in the eigenvector computation. The eigenvectors are extracted by repeated application of inverse iteration and reorthogonalization (Fann and Littlefield 1993). The eigenvectors are then back transformed by matrix multiplication using the Householder matrix to obtain the eigenvectors of the standard eigensystem problem. One more matrix multiply using L^{-1} produces the eigenvectors of the generalized eigensystem problem.

Two problems are commonly cited for an algorithm based on bisection for eigenvalues and inverse iteration for eigenvectors (Demmel et al. 1993, page 39):

1. load balancing the computation for certain distributions of eigenvalues

2. orthogonalization of eigenvectors stored on multiple processors for clustered eigenvalues.

Load balancing the computation of eigenvalues is not a problem when one starts with dense matrices and solves for all of the eigenvectors. For PeIGS, the total time in determining eigenvalues is, at present, less than 10% of the total execution time for well separated eigenvalues and much less for clustered eigenvalues. For well spaced eigenvalues (e.g., no clusters), most of the computation time is spent in the Householder reduction. For a large cluster, say all of the eigenvalues, most of the time is consumed in the eigenvector orthogonalization and Householder reduction phases of the computation. We will elaborate on this point in Section 3.

2.3 Eigenvectors

Finding orthogonal eigenvectors can be a time consuming problem if there are large clusters of eigenvalues whose eigenvectors must be orthogonalized. In this case parallel orthogonalization is used to ensure orthogonality of the eigenvectors.

Let T be a real, symmetric, tridiagonal matrix of order n with diagonal elements d_i and subdiagonal

- Perform cluster analysis:
  $k = 1$; $clustr(1) = \{1\}$
- **forall** $j = 2, m$
  if $\lambda_j - \lambda_{j-1} < 10^{-3}||T||_R$
  then $clustr(k) = clustr(k) \cup \{j\}$
  else $k = k + 1$, $clustr(k) = \{j\}$
- **end forall** j
- schedule cluster information for processors
- **forall** $k = 1$, number of clusters
  forall $j \in clustr(k)$
  initialize vector z_j
  with random vector
  end forall j
  for $i = 1, 2$
  forall $j \in clustr(k)$
  initialize iterate norm $\sigma_j = 0.0$.
  factor $T - \lambda_j = P_j L_j U_j$
  set $z_j = P_j z_j$
  loopA: until $\sigma_j \sqrt{10n} \geq 1$
  set $\delta = n||T||_R/||z_j||_1$
  set $\delta = \delta max(d_n/||T||_R, \epsilon)$
  scale $y_j = \delta z_j$
  solve $L_j U_j z_j = y_j$
  set $\sigma_j = ||z_j||_\infty$.
  end loopA
  normalize $z_j = z_j/||z_j||_2$.
  end forall j
  if $i = 1$ perform two iterations of
  panel blocked MGS on $clustr(k)$
  if $i = 2$ perform 1 iteration of
  panel blocked MGS on $clustr(k)$
  end for i
- **end forall** k

FIG. 1. *Algorithm for computing eigenvector*

elements e_i. Following standard practice, we denote T by $[e_i, d_i, e_{i+1}]$. We assume that the eigenvalues of T have been computed and are arranged in increasing order, and we define a *cluster* of eigenvalues to mean a maximal subset of monotonically increasing eigenvalues of T, $\lambda_j \leq \lambda_{j+1} \leq ... \leq \lambda_k$ such that for $j \leq l < k$, $\lambda_{l+1} - \lambda_l < 10^{-3}||T||_R$ where $||T||_R = max_i\{|e_{i-1}| + |d_i| + |e_i|\}$ and no larger subset of eigenvalues of T containing S satisfies these properties. This is the same definition of cluster that is used in LAPACK 1.0 (Jessup and Ipsen 1992) (Anderson et al. 1992).

Most of the current inverse iteration methods (Jessup and Ipsen 1992) (Anderson et al. 1992) perform inverse iteration on one vector in a cluster and then orthogonalize this vector against previously converged eigenvectors corresponding to eigenvalues in the cluster. This is serial in nature.

Inverse iteration is performed on all of the eigenpairs in a cluster in a perfectly parallel fashion in our repeated inverse iteration and re-orthogonalization method. Orthogonalization is then performed in parallel using all the processors storing the unconverged eigenvectors of the cluster. Usually two iterations of inverse iteration and orthogonalization are needed. The key here is that two iteration of modified Gram-Schmidt is used in the first orthogonalization (Fann-Littlefield 1993). We summarize the algorithm in Figure 1.

3 PERFORMANCE

In this section we describe the clustering of eigenvalues and the corresponding performance of our implementation on an Intel Paragon and on a Kendall Square Research KSR-2. The parallel performance of our algorithm is dominated by the Householder reduction of the standard eigensystem problem to tridiagonal form and by the orthonormalization of eigenvectors corresponding to clustered eigenvalues.

We present two test problems and three chemistry applications where all the eigenvalues and eigenvectors are required. Problems 1 and 2 provide an indication of the performance for the reduction steps as well as PeIGS's performance on problems with different cluster characteristics. The reduction steps from the generalized eigenproblem to the tridiagonal eigenproblem do not depend on the spectrum. The timings from problem 1 and problem 2 are typical for the reduction steps: choleski factorization, inverse computation, matrix multiplications to form C and Householder reduction. In the figures below, we plot the speed up curves for the two test problems, and the time consumed by each part of the code. We also plot the average communication time versus the completion time in Figures 6, 10, 11 (communication curve). The average communication time is defined by $(\sum_{i=0}^{p-1}(commtime(i)))/p$ where $commtime(i)$ is the time processor i spends in communication. All processors synchronize at the beginning and end of the computation so that the completion time is the same on all processors. The communication curves give us a view of the communication time versus the completion time. PeIGS has its own communication library and requires only blocking send and blocking receive. On the Paragon PeIGS requires only **csend** and **crecv** from the NX message passing library. On the KSR, PeIGS uses TCGMSG (TCGMSG 1995).

We compared PeIGS with comparable routines from LAPACK using one processor of the KSR-2.[3] PeIGS was 10% slower than the LAPACK solvers (using standard reductions from LAPACK and then calling **dstev** which calls **dstebz** and **dstein**) for the

[3]The compiler switches were **f77 -O2 -r8 -xfpu3** and **cc -O2 -r8 -xfpu3** on the KSR, and **if77 -O3 -Mr8 -Minline=100**, **icc -O3 -Mr8 -Minline=100** on the Intel Paragon. The Intel Paragon message passing software and operating system was OSF R1.2.4.

Example	$n =$	*clusters*
Example1	500	[1,21], [465,495]
Example1	1000	[1,94], [860,995]
Example1	2000	[1,415], [1405,1995]
W^+_{1001}	1001	500 clusters of 2 evals
W^+_{2001}	2001	[1,1992]
$SiOSi_6$	1687	[1,1313] and smaller ones
biphenyl	966	[1,939] and smaller ones
ZSM-5	2053	[1,108],[109,301],[309,416]
		[417,740],[741,936],[939,1590]
		[1591,2052] and smaller ones

FIG. 2. *Sizes and location of the clusters*

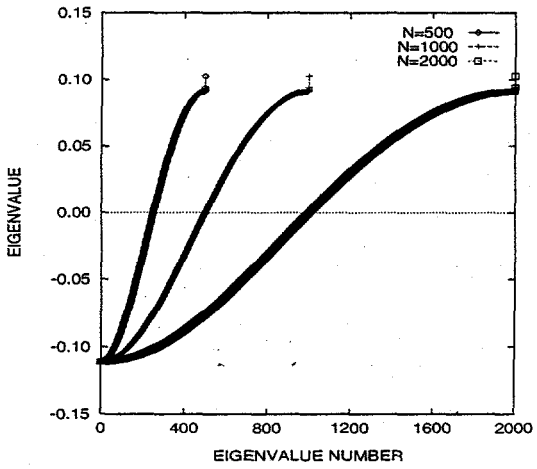

FIG. 3. *Problem 1 eigenvalues*

generalized eigenproblem using problem 1 with $n = 1000$.

In calculating the speedups on the Intel Paragon and the KSR-2 we used the observed one processor time as much as possible. For certain problems we were unable to obtain valid one processor times because of memory limitations. In those cases, we estimated 1-processor times by extrapolation from observed times with multiple processors. On the KSR-2, this was done with example 1 with $n = 2000$, and with ZSM-5. In both cases, we extrapolated from the two processor time by multiplication by 2. On the Intel Paragon, extrapolation was done with problem 1 for $n = 2000$, W^+_{2001}, $SiOSi_6$, and ZSM-5. For $SiOSi_6$ the extrapolated 1-processor time was obtained by multiplying the two processor time by 2; for the others we multiplied the four processor time by 4.

In the following we will assume that all of the eigenvalues are in increasing order. The clusters will be reported in terms of their location given this order.

3.1 Test Problems

The two test problems use matrices that

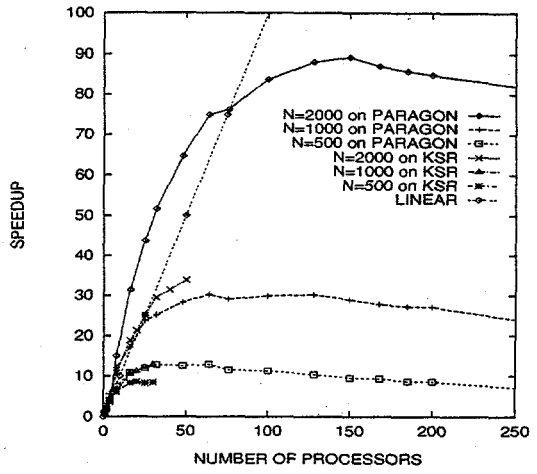

FIG. 4. *Problem 1 speedups*

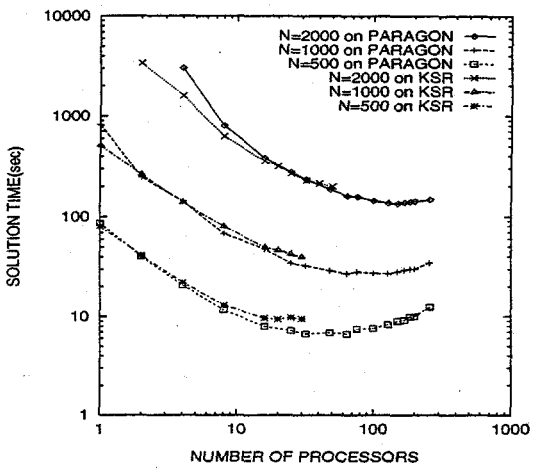

FIG. 5. *Problem 1 solution times*

represent the extremes of clustering. The first test problem has only two clusters, each containing from 5% to 20% of the eigenvalues, depending on n. (The clusters are relatively larger for larger n.) The second problem uses the well known Wilkinson's tridiagonal matrices W^+_{1001} and W^+_{2001}. The matrix W^+_{2001} has a cluster of size 1993. The largest cluster of W^+_{1001} is 2. The locations of the clusters are shown in Figure 2.

3.1.1 Problem 1 This is an example of a generalized eigensystem problem. The test matrix is a symmetric matrix A of dimension 2000 with diagonal elements $A(i,i) = 1/i$, $i = 1, 2000$ and $A(i, i-1) = -1$, $i = 2, 2000$. The matrix B is a symmetric tridiagonal matrix $[-1, 20, -1]$. This problem is characterized by clusters of eigenvalues whose sizes are small relative to n.

The speedup curves for $n = 500$, $n = 1000$ and $n = 2000$ are given in Figure 4. The cluster information is given in Figure 2, the spectra in Figure 3, the speedup

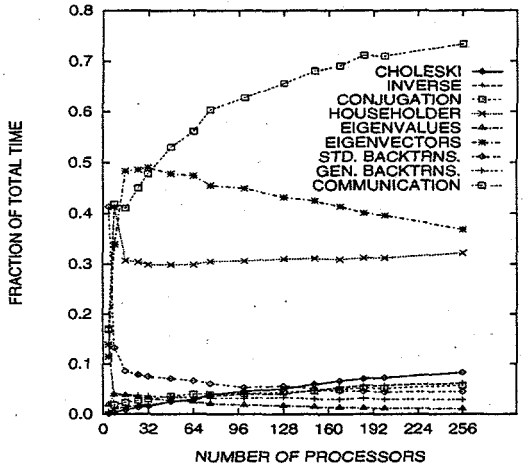

FIG. 6. *Time fraction for components on the Paragon for problem 1 for n=2000*

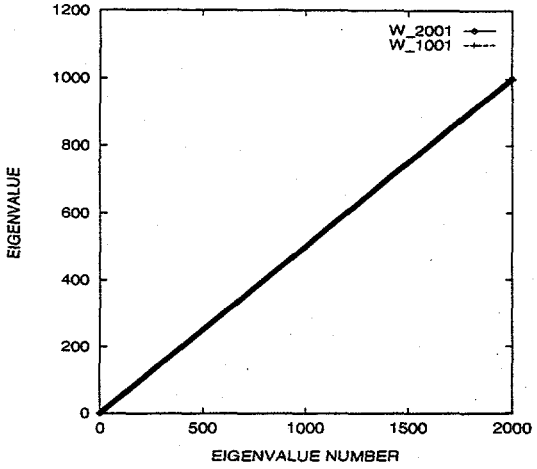

FIG. 7. W_{1001}^+ *and* W_{2001}^+ *eigenvalues*

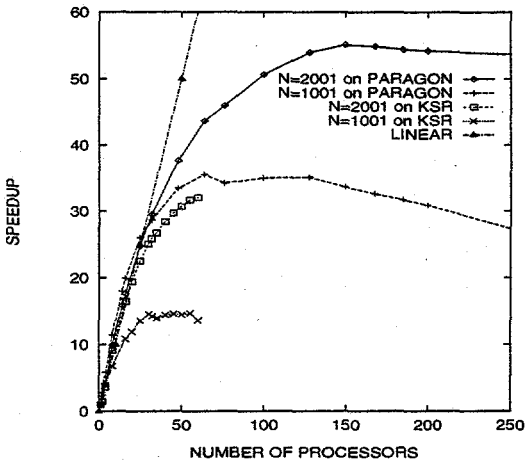

FIG. 8. W_{1001}^+ *and* W_{2001}^+ *speedups*

and time curves in Figures 4 and 5, and the fraction of time spent in various computations is given in Figure 6.

In Figure 6 the different curves are the fraction of total time spent choleski factoring B (curve `choleski`), computing L^{-1} (curve `inverse`), computing $C = L^{-1}AL^{-t}$ (curve `conjugation`), householder reduction (curve `householder`), eigenvalue computation (curve `eigenvalue`), eigenvector computation (curve `eigenvector`), back transformation for the standard eigenproblem (curve `std. backtrns.`), and back transformation for the generalized eigenproblem (curve `gen. backtrns.`). The communication time (curve `communication`) is the average of the total communication time for all of the processors divided by the wall clock time.

Figure 4 shows that for small clusters we attain a speedup of 89 using 150 processors for $n = 2000$. Thus, for $n/p < 14$ we were able to reduce the time from over 3000 seconds using four processors to under 135 seconds. The superlinear speedup is due to the small 16K cache on each of the Paragon's i860 nodes. For the $n = 500$ case, PeIGS achieves a speedup of little over 12 using 64 processors. Using 16 processors we already attain a speedup of over 10. In this case the computation is communication bound.

Figure 6 shows that for moderately large clusters the solution time is dominated by the eigenvector orthogonalization and Householder reduction. The other pieces of the algorithm (choleski factorization, inverse, conjugation, matrix multiplications) consume large fractions of the time for less than 16 processors but are not a factor for larger processor counts. Notice that the matrix multiplication using the Householder matrix and the two multiplications for conjugation start to become a factor as we use more than 190 nodes. However, the Householder reduction fraction curve is rising as fast as the matrix multiplication, so that the Householder would continue to be the bottleneck for even larger matrices.

3.1.2 Problem 2: W_{1001}^+ and W_{2001}^+ The second set of test matrices are Wilkinson's W_{1001}^+ and W_{2001}^+ matrices (Wilkinson 1965, p. 308). This test problem presents clusters which are on the extreme performance ends for our symmetric eigensolver.

The Wilkinson W_{m+1}^+ matrix is defined by

$$d_i = \begin{cases} 1000 + 1 - i & i = 1, ..., m/2 + 1 \\ i - 1000 - 1 & i = m/2 + 2, ..., m + 1 \end{cases}$$

$$e_i = 1 \quad i = 1, ..., m + 1.$$

For the W_{2001}^+ matrix the lowest 1993 eigenvalues form a single cluster. In contrast, for W_{1001}^+, there are 500 clusters of two eigenvalues each. For both matrices, the

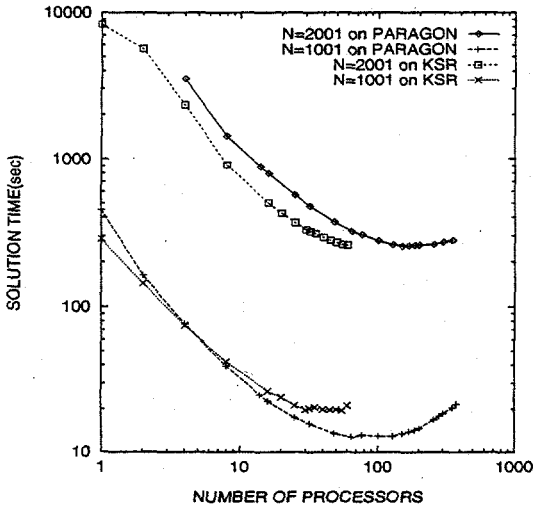

FIG. 9. W^+_{1001} *and* W^+_{2001} *solution times*

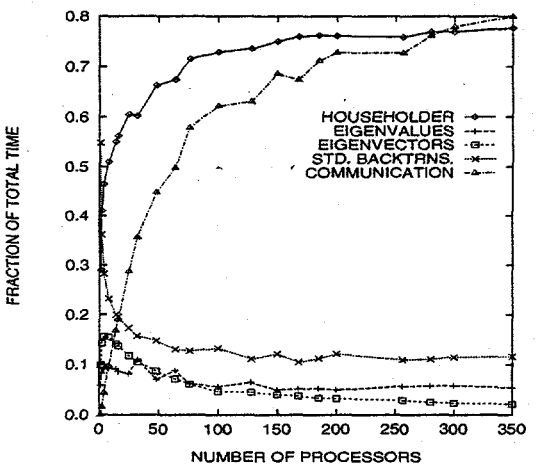

FIG. 10. *Time fraction for components on the Paragon for* W^+_{1001}

eigenvalues are almost uniformly spaced in pairs between 0 and $n/2$. Thus, the dramatic difference is an artifact of the heuristic definition of "cluster", rather than indicating any basic difference in spectral characteristics. Nonetheless, because of the sizes of the clusters under the definition used, these two matrices represent nearly the fastest and the slowest solution times for our algorithm.

A fast solution time can be achieved for W^+_{1001} as shown in Figure 8 and Figure 9. In Figure 8 the time for complete solution can be reduced to about 12 seconds with a speedup of 35 using 128 processors of the Paragon. For this example there are no clusters that span multiple processors, so the eigenvectors extraction is performed in a perfectly parallel fashion as shown in Figure 10. Notice that the eigenvalue computation consumes less than 10% of the total time and that the

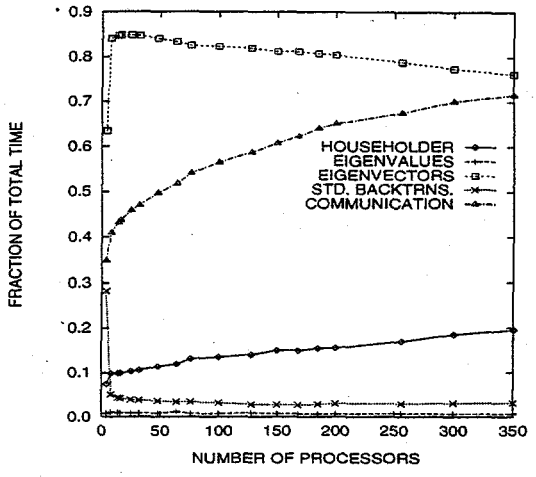

FIG. 11. *Time fraction for components on the Paragon for* W^+_{2001}

overall computation is dominated by the Householder reduction to tridiagonal form.

An example of a problem with the slowest solution time for our algorithm is W^+_{2001}. In this case the cluster is almost the whole spectrum and orthogonalization must be performed across all of the processors. However, better speedup is achieved than with W^+_{1001} because a larger fraction of the work, including orthogonalization, can be performed in parallel. This is illustrated in Figures 8 and 9. Figure 11 shows that more than 75% of the time is spent in eigenvector orthogonalization, and that the rest of the computation is dominated by Householder reduction. PeIGS attains a speedup of more than 53 using 150 processor of the Paragon with a solution time of 255 seconds. On the KSR-2 we achieve a speedup of 32 using 62 processors with a solution time of 260 seconds. Again, as n/p approaches 20 with n fixed and with increasing p, we notice that there is no more speedup and communication begins to dominate.

3.2 Examples from Chemistry

In this section we present performance number for real chemistry applications. We refer the development of the chemistry codes and models to the references (Guest et al. 1994) (Bernholdt and Harrison 1994) (Wong and Harrison 1994) because we cannot possibly do justice to their exposition.

The chemistry problems have cluster characteristics that are similar in size to those of Problem 1 and W^+_{2001}; although, the spectra are quit different. This suggests that the speedup curves should resemble those of Figure 4 and Figure 8.

3.2.1 $SiOSi_6$ Density functional theory methods

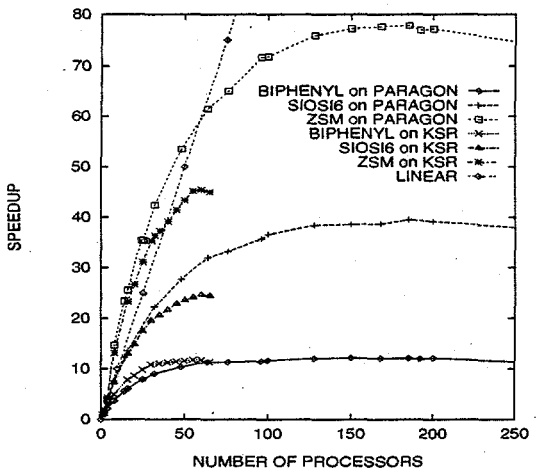

FIG. 12. *Quantum chemistry speedups*

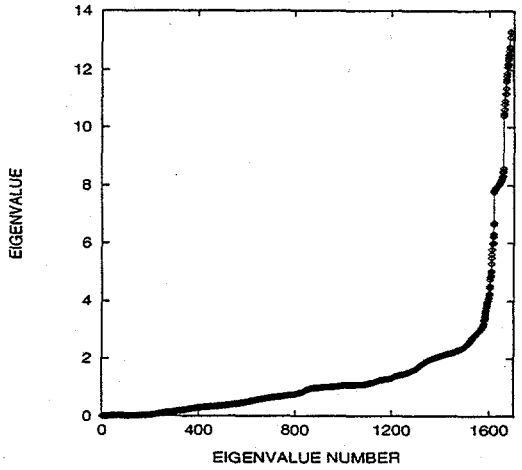

FIG. 14. *$SiOSi_6$ eigenvalues*

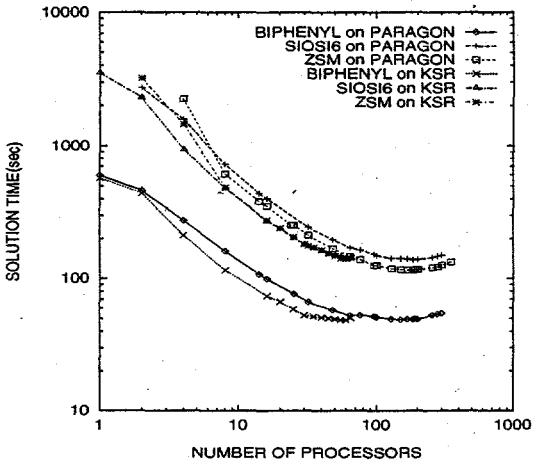

FIG. 13. *Quantum chemistry solution times*

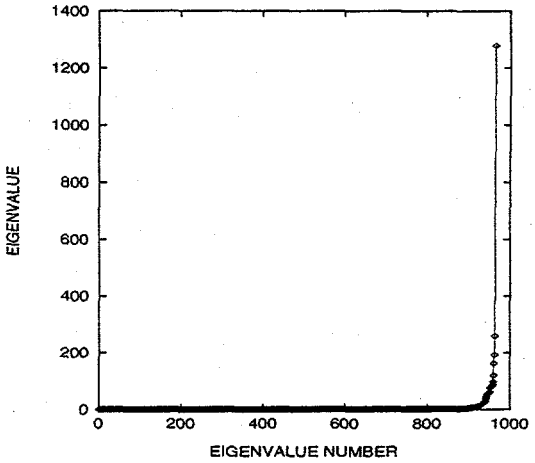

FIG. 15. *BiPhenl eigenvalues*

(Guest et al. 1994) produce certain types of matrices with a large cluster of eigenvalues. This is an example of a real symmetric eigensystem problem for determining bulk properties for the molecule $SiOSi_6$. The overlap matrix uses the deMon double zeta plus polarization orbital basis and deMon fitting auxiliary basis for charge density and exchange correlation (St-Amant and Salahub 1990). The overlap matrix is a real symmetric and positive definite matrix of size $n = 1687$. The spectrum of this matrix is shown in Figure 14. This matrix has one large cluster of 1313 eigenvalues in the low end of the spectrum. The eigenvalues in this cluster are contained in the interval [0.0012008200880054, 1.7029431674296136]. There are 32 smaller clusters ranging in sizes from 2 to 126 scattered throughout the upper end of the spectrum. There are two large clusters of size 126. The others are between sizes 2 to 11.

3.2.2 Biphenyl This is an example of a standard symmetric positive definite eigensystem problem occuring in Møller-Plesset theory in the modeling of biphenyl (Bernholdt and Harrison 1995). Biphenl is $2,2' - di(trifluoromethyl)biphenyl$. This matrix is the overlap matrix of dimension $n = 966$ with a large cluster consisting of the first 939 eigenvalues. There are four other clusters of sizes 2 to 4: 939-940(2), 947-949(3), 950-952(3), 954-957(4).

3.2.3 Zeolite ZSM-5 This is an example of a standard symmetric eigenproblem that is reduced from a generalized eigensystem problem. This eigenproblem appears in the application of self-consistent field(SCF) Hartree-Fock method for solving the non-linear Schrodinger problem using "Zeolite-fragment with embedded pyridine 6-31G basis set" (Wong and Harrison 1995). The spectrum is characterized by scattered clusters of eigenvalues of various sizes: 1-108(108), 109-301(193), 304-307(4), 309-416(108), 417-740(324), 741-936(196), 939-1590(652), 1591-2052(462).

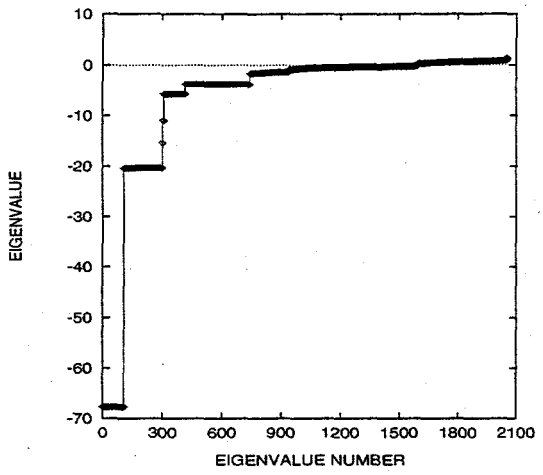

FIG. 16. *ZSM-5 eigenvalues*

4 CONCLUSION

We have described the parallel performance of a column based dense real symmetric generalized and standard eigensolver based on bisection for eigenvalues and on a repeated inverse iteration and reorthogonalization method for eigenvectors, on a number of extreme examples and on some real-world problems. We were able to obtain good speedup for matrices during production runs.

For inverse iteration and orthogonalization there may be better heuristics for determining clusters than the conventional one that we used. An alternative definition that produced smaller clusters would certainly reduce the amount of time spent orthogonalizing eigenvectors, but could also cause increased error or loss of orthogonality. A systematic study of this speed versus accuracy tradeoff would be valuable.

5 REFERENCE

Anderson et al. 1992. *LAPACK User's Guide*, SIAM, Philadelphia. LAPACK is publicly available via `netlib`.

Bernholdt, D., and R. Harrison. 1995. "Orbital Invariant Second Order Many-Body Perturbation on Parallel Computers: An Approach for Large Molecules." *J. Chem. Physics.* Accepted for publication.

Chinchalkar, S. and T. Coleman. 1991. "Computing Eigenvalues and Eigenvectors on a Dense Real Symmetric Matrix on the Ncube 6400." Technical Report CTC91 TR74. Cornell Theory Center, Cornell University, Ithaca, NY(Sept.).

Demmel, J.W., M.T. Heath and H.A. van der Vorst. 1993. "Parallel Numerical Linear Algebra." LAPACK Working Note 60., Computer Science Technical Reports CS-93-192, Univ. of Tennessee, Knoxville, TN(August).

Fann, G. and R.J. Littlefield. 1993. "Inverse Iteration and Re-orthogonalization." In *Proceeding of the Sixth SIAM Conference on Parallel Processing For Scientific Computing, Norfolk, VA, March 22-24.* R. Sincovec et. al., ed.:409-413, SIAM, Philadelphia.

Guest, M.A. et. al. 1995. "High Performance Computational Chemistry; NWChem and Fully Distributed Parallel Applications." In *High Performance Computing: Technology and Applications.* eds. L. Grandinetti, G.R. Joubert, J.J. Dongarra, and J. Kowalik, Elsevier, Amsterdam(in press).

Harrison, R. 1995. "TCGMSG manual." The latest version of `tcgmsg.4.05` is available via anonymous ftp from `ftp.anl.gov:pub/tcgmsg`.

Geist, G.A. and M.T. Heath. 1986. "Matrix factorization on a hypercube processor." In *Hypercube Multiprocessors,* M.T. Heath ed., SIAM, Philadelphia:161-180.

Ipsen, I. and E.R. Jessup. 1990. "Solving the tridiagonal eigenvalue problem on the hypercube," *SIAM J. Sci. Stat. Comput.,* vol. 11:203-229.

Jessup, E.R. and I. Ipsen. 1992. "Improving The Accuracy of Inverse Iteration," *SIAM J. Sci. Stat. Comput.,* vol. 13, no. 2:550-572.

Littlefield, R.J. and K.J. Maschhoff. 1993. "Investigating the performance of parallel eigensolvers for large processor counts," *Theor. Chim. Acta,*:457-473.

Lo, S.S., B. Philippe, and A. Sameh. 1987. "A Multiprocessor Algorithm for the Symmetric Tridiagonal Eigenvalue Problem." *SIAM J. Sci. Stat. Comput.,* vol. 8, no. 2(March).

St. Amant, A., D. R. Salahub. 1990. *Chem. Phys. Letter,* vol. 169:387.

Smith, B.T., et al. 1976. *Matrix Eigensystem Routines–EISPACK. Lecture Notes in Computer Science,* vol 6, 2nd ed., Springer-Verlag.

Wilkinson, J. H. 1965. *The Algebraic Eigenvalue Problem,* Clarendon Press.

Wong, A.T., and R.J. Harrison. 1995. "An Approach to Large SCF Calculations," *J. Comp. Chem.* Accepted for publication.