Application of a New Algorithm for the Symmetric Eigenproblem to Computational Quantum Chemistry

Inderjit Dhillon[*] George Fann[†] Beresford Parlett [‡]

Abstract

We present performance results of a new method for computing eigenvectors of a real symmetric tridiagonal matrix. The method is a variation of inverse iteration and can in most cases substantially reduce the time required to produce orthogonal eigenvectors. Our implementation of this algorithm has been quite effective in solving "degenerate" eigenproblems in computational chemistry. On a biphenyl example, the implementation is 46 times faster than an earlier PeIGS 2.0 code using 1 processor of the IBM SP. It reduces the time for computing eigenvectors of this 966×966 matrix to under 0.15 seconds using 64 processors of the IBM SP. We present performance results for calculations from the SGI PowerChallenge and the IBM SP.

1 Introduction

In computational chemistry, the solution of dense real symmetric standard eigensystem problems is often required to obtain energy states of Hamiltonians. The computational chemists require all or a certain fraction of the spectrum and its associated eigen-subspaces to predict information about potential energy surfaces. In practice, the solution to the eigenproblem is limited by available processing time and computer memory, which is why fast and compact schemes are of special interest. In general construction of the matrix is the time-consuming step. These considerations limit the type and size of problems that can currently be solved.

In current software implementations, it is common practice to reduce the generalized and the standard eigenproblems to a tridiagonal eigenproblem [6] [13]. One of the most flexible, general, and fast methods for solving the real symmetric tridiagonal eigenproblem is based on some form of bisection for the eigenvalues and inverse iteration for the eigenvectors. The time required to solve for the eigenvalues of a tridiagonal matrix is small in comparison to the time required to find the orthonormal eigenvectors, perform the Householder reduction to tridiagonal form, and perform the backsolve steps required to find the eigenvectors for the dense problem.

A historical problem with inverse iteration is the appearance of eigenvalues that are close together. Past implementations of the inverse iteration process are sensitive to the choice of the initial starting vector. If the eigenvalues are too close and the starting vectors for inverse iteration are not chosen well, the resulting eigenvectors will not be orthogonal and are likely have significant components in the same direction. A common remedy to ensure orthonormality of the eigenvectors

[*]Computer Science Division, EECS, University of California, Berkeley. Support provided by the EMSL construction project at PNNL and contract ARPA DAAL03-91-C0047 via a subcontract from the University of Tennessee.

[†]Environmental Molecular Sciences Laboratory, Pacific Northwest National Laboratory, Richland, WA 99351. Pacific Northwest is operated by Battelle Memorial Institute for the U.S. Department of Energy under Contract DE-AC06-76RLO 1830. Support provided by EMSL construction project at PNNL.

[‡]Department of Mathematics and Computer Science Division, University of California, Berkeley. Support provided by ONR contract N000014-90-J-1372.

is to orthonormalize the approximate eigenvectors in some fashion. Unfortunately, the need for orthogonalization introduces load balancing problems and possible bottlenecks in an otherwise perfectly parallel algorithm. Three examples from real world quantum chemistry simulations[8] using the parallel eigensolver PeIGS have shown that the definition of a cluster of degenerate eigenvalues from LAPACK's inverse iteration solver DSTEIN may be unnecessarily large. The parallel eigensolver PeIGS 2.0 uses DSTEIN's definition of a cluster of close eigenvalues and performs parallel orthogonalization of eigenvectors.

For example, for a 966×966 dense real symmetric matrix that arises in the modeling of a biphenyl, the serial LAPACK's tridiagonal inverse iteration was found to take nearly 80-90% of the total CPU time to solve the dense real standard eigenproblem using 1 processor of an IBM RS6000/590. On massively parallel processors the performance is similar, with orthonormalization consuming 75% of the total CPU time of PeIGS 2.0 on 64 processors of an Intel Paragon[1][8].

In this paper we focus on a new preliminary version of a fast and scalable algorithm for finding orthonormal eigenvectors of real symmetric tridiagonal matrices. For the sake of brevity we call it the Berkeley algorithm in this paper. It is preliminary because this paper contains work in progress, and components of the preliminary algorithm are expected to change in the near future.

The Berkeley algorithm shows promise in reducing the amount of required orthonormalization. The algorithm has as its impetus K.V. Fernando's solution to the Wilkinson problem[9]; how to choose the starting vector for inverse iteration. The Berkeley algorithm outlines one approach to solving the tridiagonal eigenproblem. It is based on an algorithm being developed by Dhillon and Parlett [4]. The main innovation lies in discarding the $2n - 1$ diagonal and off-diagonal elements of the symmetric tridiagonal matrix in favour of more "stable" bidiagonal factors from which the eigenvectors can be computed. Carefully chosen inner loops that do not compromise computational efficiency are used to obtain greater accuracy. As a result, "clusters" of eigenvalues in our algorithm are much smaller than those defined by the LAPACK routine DSTEIN. In chemistry applications this means that physical degeneracies are more closely aligned with numerical degeneracies, resulting in smaller clusters of eigenvectors requiring orthogonalization.

In the simulation of biphenyl discussed above, the Berkeley algorithm solves the eigenvectors for the symmetric tridiagonal eigenproblem 46 times faster than an earlier PeIGS 2.0 implementation when each is executed using one processor of an SGI PowerChallenge. Using 10 processors of the SGI PowerChallenge, the parallel Berkeley algorithm computes all of the eigenvectors in under 0.5 seconds. This is 32 times faster than the comparable subroutine for computing eigenvectors of symmetric tridiagonal matrices from parallel PeIGS 2.0 executing on 10 processors. If we replace the older symmetric tridiagonal solver in PeIGS 2.0 with the the Berkeley algorithm in the standard real symmetric eigensystem solver the solution time using 10 processors is reduced from 41 seconds to 14 seconds.

Using 128 processors of an IBM SP[2], the time required to find eigenvectors for the tridiagonal symmetric eigenproblem is now under 0.15 seconds for the Berkeley algorithm. Extensive spectral and timing results are presented in section 3.

2 The Berkeley Algorithm

We outline here a preliminary version of the Berkeley algorithm being developed by Dhillon and Parlett [3, 4] to compute orthogonal eigenvectors of a symmetric tridiagonal matrix. The Berkeley algorithm promises to be an $O(n^2)$, reliable and embarrassingly parallel method. This algorithm is still under development, so in this paper we present only the preliminary algorithm, based on some of the ideas in references [3, 4]. The preliminary version presented in this paper can be $O(n^3)$ in

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

[2]This research was performed in part using the Molecular Sciences Computing Facility in the Environmental Molecular Sciences Laboratory at the Pacific Northwest National Laboratory. Thanks to MSCF for providing early access to HPCS-1A for this work.

the worst case. However, the improvements in speed obtained with this preliminary version over current LAPACK inverse iteration software are substantial (see section 3). The algorithm presented here is also as accurate as the current LAPACK inverse iteration software.

In order to describe the algorithm, we must first define some important terms and conventions.

For our purposes, let T denote an n dimensional real symmetric tridiagonal matrix with diagonal d and lower diagonal e. Let us assume, for simplicity, that T consists of 1 block and that no entries of the upper diagonal are 0.

We define the absolute and relative gaps between the two eigenvalues λ_i and λ_{i+1} as

$$\begin{aligned} \mathrm{absgap}(\lambda_i, \lambda_{i+1}) &= |\lambda_{i+1} - \lambda_i| \\ \mathrm{relgap}(\lambda_i, \lambda_{i+1}) &= \frac{|\lambda_{i+1} - \lambda_i|}{\max(|\lambda_i|, |\lambda_{i+1}|)} \end{aligned}$$

The Berkeley algorithm consists of two main parts, identified as Algorithm 2.1 and Algorithm 2.2. The main steps in each of these algorithms can be summarized as follows.

ALGORITHM 2.1. **Computes the eigensystem of T.**

1. Choose μ such that $T + \mu I$ is positive definite.
2. Compute the Cholesky Decomposition :
$$T + \mu I = LL^t.$$
3. Compute the singular values of L, denoted by $0 \le \hat{\sigma}_1 \le \hat{\sigma}_2 \le \cdots \le \hat{\sigma}_n$, to high relative accuracy (this may be done by a bisection algorithm or by the **dqds** algorithm of Fernando and Parlett [10]).
4. Group the eigenvalues of LL^t, $\hat{\lambda}_i = \hat{\sigma}_i^2$, as :

   a.) isolated — $\hat{\lambda}_m$ is isolated if
$$\min(\mathrm{relgap}(\hat{\lambda}_m, \hat{\lambda}_{m+1}), \mathrm{relgap}(\hat{\lambda}_m, \hat{\lambda}_{m-1})) > \min(10^{-3}, 1/n).$$

   b.) loose clusters — $\hat{\lambda}_m, \hat{\lambda}_{m+1}, \ldots, \hat{\lambda}_{m+l}$ form a loose cluster if
$$\max(100, n)\epsilon \le \min(\mathrm{relgap}(\hat{\lambda}_i, \hat{\lambda}_{i+1}), \mathrm{relgap}(\hat{\lambda}_i, \hat{\lambda}_{i-1})) \le \min(10^{-3}, 1/n),$$
   for $m \le i \le m + l$, and the neighboring eigenvalues $\hat{\lambda}_{m-1}$ and $\hat{\lambda}_{m+l+1}$ are isolated.

   c.) tight clusters — $\hat{\lambda}_m, \hat{\lambda}_{m+1}, \ldots, \hat{\lambda}_{m+l}$ are tightly clustered if there are eigenvalues $\hat{\lambda}_j$, $\hat{\lambda}_{j+1}$, $m \le j < m + l$, such that
$$\mathrm{relgap}(\hat{\lambda}_j, \hat{\lambda}_{j+1}) < \max(100, n)\epsilon,$$
   and
$$\min(\mathrm{relgap}(\hat{\lambda}, \hat{\lambda}_{i+1}), \mathrm{relgap}(\hat{\lambda}_i, \hat{\lambda}_{i-1})) \le \min(10^{-3}, 1/n),$$
   for $m \le i \le m + l$, and the neighboring eigenvalues $\hat{\lambda}_{m-1}$ and $\hat{\lambda}_{m+l+1}$ are isolated.
5. For all isolated and loose clusters of eigenvalues, compute the corresponding eigenvectors by Algorithm 2.2. For a loose cluster, orthogonalize the corresponding eigenvectors by a process such as the modified Gram-Schmidt method.
6. For tight clusters, do "traditional" inverse iteration with reorthogonalization in the inner loop to compute the eigenvectors (as in LAPACK). Instead of a random starting vector, the starting vectors for inverse iteration can be chosen as e_r where r is chosen by steps 1 and 2 of Algorithm 2.2. Note that the choice of r might be different for each eigenvalue in the cluster.

□

Algorithm 2.2. **Computes an eigenvector z of LL^t given an eigenvalue estimate $\hat{\lambda}$.**

1. Compute $LL^t - \hat{\lambda} I = L_+ D_+ U_+ = U_- D_- L_-$.
2. Compute $\gamma_k = D_+(k) + D_-(k) - (LL^t)_{kk}$ for $k = 1, 2, \ldots, n$, and choose the index r where $|\gamma_k|$ is minimum.
3. Solve $(LL^t - \hat{\lambda} I) z = \gamma_r e_r$.

□

The "+" means to factor from top down and "-" means to factor from bottom up.

There are a number of significant differences between the above algorithm and LAPACK's implementation of inverse iteration. The most important ones can be summarized as follows;

1. Rather than working with the $2n-1$ diagonal and off-diagonal entries of the tridiagonal matrix T, Algorithm 2.1 chooses an alternative representation of T, namely the Cholesky factor L of $T + \mu I$, for an appropriate shift μ. The bidiagonal matrix L is a "better" representation of the tridiagonal matrix for our purposes [3, 4].
2. In our Berkeley algorithm, each individual eigenvector is computed by Algorithm 2.2, which in step 3 above can be thought of as doing the initial step of inverse iteration with starting vector e_r. The important point is that e_r is chosen in a deterministic way and is guaranteed to contain a large component in the desired direction. This is in contrast to LAPACK's inverse iteration, which chooses a random starting vector. Another important observation is that with the help of the triangular factorizations, the eigenvector computed in step 3 can be formed solely by multiplications and divisions; no additions are necessary. Additional details on this method, such as alternative formulae for γ_k that are better for computational purposes, can be found in [9, 5, 3].
3. The separation criterion used in Algorithm 2.1 to determine whether orthogonalization is needed is a relative criterion as opposed to the absolute one used in LAPACK, which orthogonalizes the iterates computed for λ_i, λ_{i+1} if

$$\frac{\operatorname{absgap}(\hat{\lambda}_i, \hat{\lambda}_{i+1})}{\|T\|} \leq 10^{-3} \tag{2.1}$$

   In contrast, Algorithm 2.1 orthogonalizes when

$$\operatorname{relgap}(\hat{\lambda}_i, \hat{\lambda}_{i+1}) \leq \min\left(\frac{1}{n}, 10^{-3}\right). \tag{2.2}$$

   Since

$$\operatorname{relgap}(\hat{\lambda}_i, \hat{\lambda}_{i+1}) \geq \frac{\operatorname{absgap}(\hat{\lambda}_i, \hat{\lambda}_{i+1})}{\|T\|},$$

   our algorithm orthogonalizes less often than LAPACK's DSTEIN. This difference is particularly significant if the eigenvalues are small in magnitude. For the quantum chemistry example presented in section 3, this difference buys substantial savings in time. Note also the additional factor $1/n$ in (2.2) that relaxes the orthogonality tolerance for large n, while ensuring that the desired residual and orthogonality are satisfied.
4. In Algorithm 2.1, we distinguish between *loose* and *tight* clusters. In LAPACK, all clusters are treated the same and all reorthogonalization is done in the inner loop. In contrast, for *loose* clusters, we need to orthogonalize only at the end to guarantee accuracy. For *tight* clusters, Algorithm 2.1 essentially does the same computations as LAPACK. The savings in time due to *loose* clusters can be significant.

As we have mentioned earlier, our algorithm can be much faster than existing implementations of inverse iteration. The major advance is in using the bidiagonal matrix L for computational purposes. Theory developed in [12, 2, 7, 14] implies that by using carefully constructed inner loops, we can compute very accurate eigenvectors of LL^t. The dot products between these computed eigenvectors are determined by the relative gaps between eigenvalues. It can be shown that if $\text{relgap}(\hat{\lambda}_i, \hat{\lambda}_{i+1}) = 10^{-d}$(i.e., $\hat{\lambda}_i$ and $\hat{\lambda}_{i+1}$ agree in d digits) then the eigenvectors z_i and z_{i+1} computed by Algorithm 2.2 have a dot product of $10^d\epsilon$. For example, if $\hat{\lambda}_1 = 1.0 \times 10^{-50}$ and $\hat{\lambda}_2 = 1.5 \times 10^{-50}$, then the corresponding eigenvectors computed by Algorithm 2.2 would have a dot product of 10^{-15} in double precision arithmetic (where $\epsilon \approx 10^{-16}$)! Note that no orthogonalization is required to obtain such vectors. The interested reader is referred to [3, 4] for more details about the theoretical foundations.

Although the algorithm presented in this paper does explicit orthogonalization that can lead to $O(n^3)$ in total work, our goal is to develop a reliable $O(n^2)$ algorithm. We hope to achieve this by using more "stable" representations, one for each cluster of eigenvalues, to compute the corresponding eigenvectors. We have had considerable numerical success in doing so. Currently, there is no theory that completely explains our numerical observations. We are in the process of developing the appropriate theory, and the patient reader is requested to wait for [3, 4].

3 Application and Results

Our parallel version of the Berkeley algorithm has been applied to produce eigenvectors for a number of tridiagonal matrices derived from quantum chemistry applications. As an illustration of the performance improvements, we have chosen to present results for an overlap matrix from a quantum chemistry application; resolution of the identity Moller-Plesset second order perturbation method (RI-MP2) applied to the modeling of a biphenyl (2,2'- ditrifluoromethyl) [1]. This example shows that improvements can be extremely beneficial and also have exciting implications.

The real symmetric tridiagonal biphenyl matrix is generated by Householder tridiagonalization of the overlap matrix of dimension 966. Due to the limitations of finite precision, we cannot hope to compute the true eigenvalues or ensure that the computed eigenvectors are exactly orthogonal. It can be shown that it is sufficient to find eigenpairs $(\hat{\lambda}_i, \hat{v}_i)$, $\hat{v}_i^T\hat{v}_i = 1$, $i = 1, 2, \ldots, n$, such that the residual norms are small, and the computed eigenvectors are numerically orthogonal. That is, the computed eigenpairs must satisfy the following conditions;

$$\|(T - \hat{\lambda}_i I)\hat{v}_i\| \leq n\varepsilon\|T\|, \tag{3.3}$$

$$|\hat{v}_i^T\hat{v}_j| \leq n\varepsilon, \quad i \neq j, \tag{3.4}$$

where ε is the machine precision.

In LAPACK's definition of a cluster, the biphenyl tridiagonal matrix has 1 large cluster consisting of the first 939 eigenvalues whose corresponding eigenvectors must be orthonormalized. For some of the older inverse iteration algorithms this is a bottleneck for computing eigenvectors on both serial and parallel processors. Using LAPACK's tridiagonal eigensolver, orthogonalization in DSTEIN uses more than 90% of the serial time to compute the eigenpairs. Using an older version of the parallel eigensolver PeIGS 2.0 more than 75% of the total solution time for the standard eigensolve on more than 64 processors of the Intel Paragon[8] was consumed by parallel modified Gram Schmidt(MGS) orthonormalization of the eigenvectors. The maximum parallel speedup obtainable was quite limited.

However, if we use the definition of a cluster from the Berkeley algorithm(Algorithm 2.1 step 4), there is no *tight* cluster of eigenvalues in this case. There are only *loose* clusters: one cluster of 9 eigenvalues, two of 8 and 66 of sizes 2 to 5. Therefore, only a large number of small orthonormalization are required.

For parallel computation we distributed the eigenvectors in panels of columns so that each processor computes essentially $\lceil n/p \rceil$ vectors in a natural order (i.e., processor 0 computes the first $\lceil n/p \rceil$ vectors, processor 1 computes the next $\lceil n/p \rceil$ vectors, etc.). For this preliminary implementation, no attempt was made to distribute and schedule the computation so that the small clusters would reside on only 1 processor, but we did add additional local synchronizations to prevent race conditions. All of the eigenvalues were calculated in a replicated fashion on each processor. All communication was performed using blocking send and blocking receives. Some additional performance may be gained using one way get/put mechanisms and asynchronous communication. In terms of orthogonalization, there are at most two processors involved in ortho-normalization of a cluster, so this step can be performed in an almost perfectly parallel fashion.

We tested the parallel variant of the Berkeley algorithm in PeIGS 3.0 using TCGMSG [3] on two parallel computers; an SGI PowerChallenge and an IBM SP[4]. We were able to achieve significantly better performance on uniprocessor and demonstrate parallel scalability. Results are presented here to describe the performance using large processor counts on the IBM SP. Refer to Figure 1 for the linear scalability of the PeIGS 3.0 performance on the SGI PowerChallenge[5].

method name	time to completion(sec)
Berkeley algorithm	6
bisection + inv it (LAPACK)	112
bisection + inv it + reorthog (PeIGS)	126
eispack	32
QR (LAPACK)	46
divide and conquer (LAPACK)	20

Table 1. Sequential time of tridiagonal eigensolver on an IBM RS6000/590 for n=966 biphenyl.

The Berkeley algorithm performs significantly better than PeIGS 2.0 and LAPACK's bisection and inverse iteration routine using only one processor on the biphenyl problem. The timing for the full solution of the real symmetric tridiagonal eigenproblem for the biphenyl calculation using the Berkeley algorithm is compared to the timing for other standard algorithms in Table 1. In Table 1, we see only a speedup of 21 between the PeIGS 2.0 performance and that of the Berkeley algorithm if we include the time for computing the eigenvalues. The reason for this is that the eigenvalues takes only 2-3 seconds and if we subtract this time then the speedup is 41 for computing the tridiagonal eigenvectors[6].

On a single processor of the IBM-SP, the Berkeley algorithm was over 46 times faster than the old PeIGS inverse iteration scheme for solving eigenvectors for the symmetric tridiagonal eigenproblem, solving for all of the eigenvectors in under 3 seconds. The residual of the two outputs are comparable and the orthogonality of eigenvectors produced by PeIGS is slightly better than that of LAPACK, $O(10^{-14})$ compared to $O(10^{-13})$. Further improvements in removing the orthogonalization step with the *tight* clusters are expected[3]. The improvements to the tridiagonal eigensystem problem are realized in the standard eigenproblem. The standard eigensystem problem for the n=966 biphenyl example can now be solved in 90 seconds using the Berkeley algorithm on 1 processor; the old PeIGS algorithm took 248 seconds.

Using 64 processors of the IBM SP, the Berkeley eigensolver can find all of the eigenvectors for the symmetric tridiagonal eigenproblem in under 0.12 seconds. This is a slight improvement over the 0.16 seconds solution time using 32 processors. There is essentially no further speedup obtained using

[3]PeIGS uses TCGMSG or MPI for communication.

[4]Thanks to the Molecular Sciences Computing Facility at PNNL for providing access to the IBM SP.

[5]The eigensolver PeIGS 3.0 uses the communication package TCGMSG[11] or MPI. On the IBM SP the PeIGS code was compiled using **xlhpf90 -Pv -O3 -qstrict** and **mpcc -O3 -qstrict** and used native MPI. On the SGI PowerChallenge the code was compiled using **f90 -O3** and **cc -O3**

[6]The difference between the thin node timings of the IBM SP (RS6000/390, 64K data cache, 55 mflops for linpack) and that of the IBM RS6000/590 (with 256K data cache, 131 mflops for linpack) is the smaller size of the data cache on the thin node and its effect on the performance for MGS for the eigenvectors.

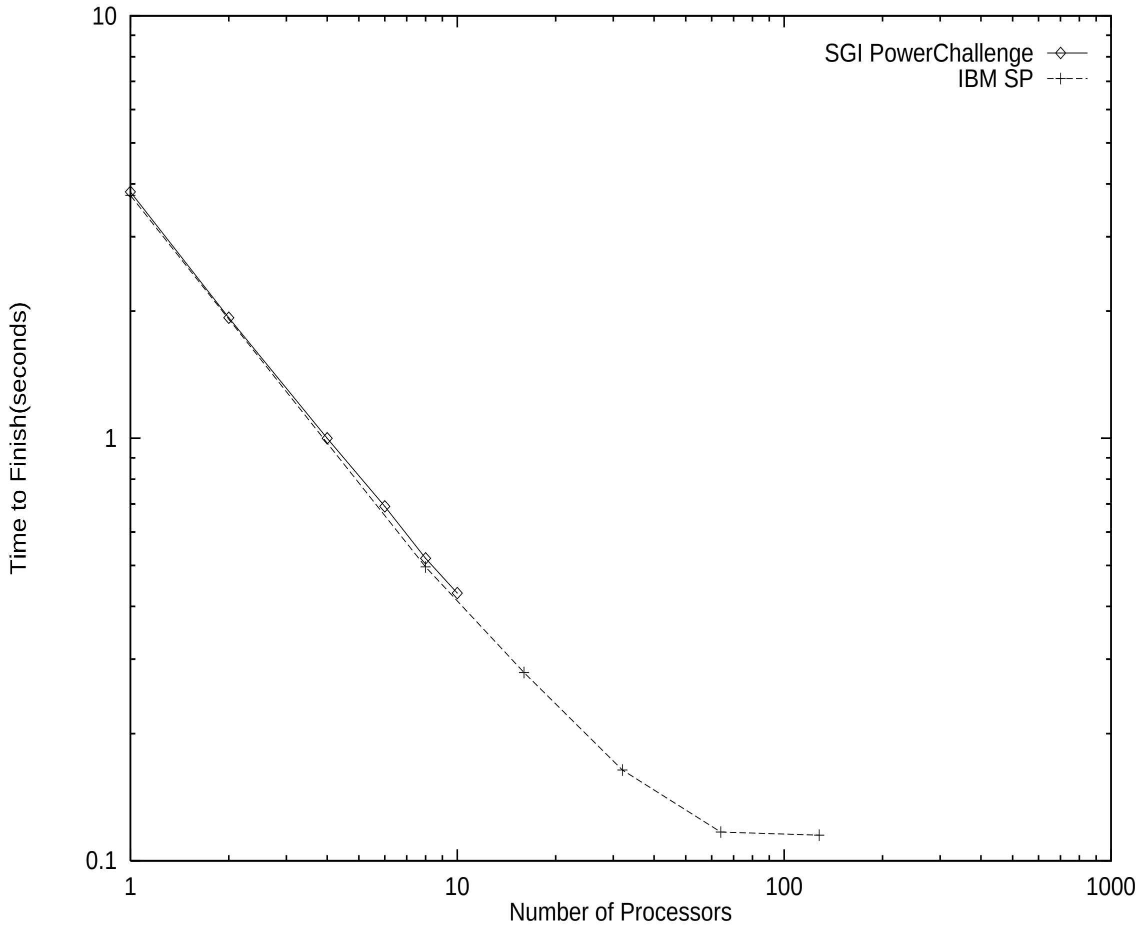

FIG. 1. *Time to compute eigenvectors of real symmetric tridiagonal matrices on the SGI PowerChallenge and the IBM SP.*

more than 32 processors; it is hard to overcome the latency with so little work on each processor. The solution of the full standard eigensystem problem can be solved in under 10 seconds using 32 processors. This is a parallel speedup of 9. In comparison, using 128 processors, PeIGS 2.0 takes 28 seconds to solve for the eigenvectors of the tridiagonal eigensystem problem. The performance of the tridiagonal eigensolver contributes to improved performance of the standard eigensolver. The standard eigensolver using the Berkeley tridiagonal algorithm is a factor of 3 faster than the older PeIGS 2.0 using 128 processors, solving the standard eigensystem problem in 11 seconds versus 33 seconds.

The main observation for large processor counts for the standard eigensystem problem is that the Householder reduction step and its backsolve now account for more than 90% of the total solution time on one processor, and more than 65% of the total time using 128 processors. The sequential tridiagonal eigenvalue solver accounts for 20% of the total time.

4 Conclusion

We have presented a preliminary version of the Berkeley algorithm for computing the eigenvectors of dense real symmetric tridiagonal This new algorithm defines a cluster of eigenvalues in a way that better reflects the degeneracies of the true eigenvalues. Using this definition of clusters and a two level approach, the Berkeley algorithm avoids the bottleneck of orthonormalization for close eigenvalues which has plagued past inverse iteration schemes. On some quantum chemistry simulations we were able to obtain significant speedup over existing serial implementations, as well as show reasonable parallel performance.

In a RI-MP2 simulation of a biphenyl, the Berkeley algorithm is faster than the older PeIGS 2.0 in computing eigenvectors of real symmetric tridiagonal matrices by a factor of more than 46 using one processor. This performance improvement reduces the solution time of the corresponding standard symmetric eigensolver time to 10 seconds using 128 processors of the IBM SP. This is a factor of 3 improvement over the PeIGS 2.0 solution time using 128 processors.

References

[1] D. Bernholdt and R. Harrison, *Orbital Invariant Second Order Many-Body Perturbation on Parallel Computers: An Approach for Large Molecules.* J. Chem. Phys. 102, 1995.

[2] J. Demmel and W. Kahan, *Accurate singular values of bidiagonal matrices.* SIAM J. Sci. Stat. Comput., 11(5):873–912, September 1990.

[3] I. S. Dhillon, *A Stable $O(n^2)$ Algorithm for the Symmetric Tridiagonal Eigenproblem.* Ph.D. thesis, University of California, Berkeley, California, Expected 1997, in preparation.

[4] I. S. Dhillon and B. N. Parlett, *Orthogonal Eigenvectors without Gram-Schmidt.* In preparation.

[5] B.N. Parlett and I.S. Dhillon, *On Fernando's method to find the most redundant equation in a tridiagonal system.* Linear Algebra and its Applications, 1996, to appear.

[6] B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler, *Matrix Eigensystem Routines – EISPACK Guide.* volume 6 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, 1976.

[7] S. Eisenstat and I. Ipsen, *Relative perturbation techniques for singular value problems*, SIAM J. Numer. Anal., 32(6), 1995.

[8] G. I. Fann, R.J. Littlefield, and D. M. Elwood, *Performance of a Fully Parallel Dense Real Symmetric Eigensolver in Quantum Chemistry Applications.* Proceedings of High Performance Computing '95, Simulation MultiConference, The Society for Computer Simulation 1995.

[9] K. V. Fernando, *On computing an eigenvector of a tridiagonal matrix*, to appear in SIMAX.

[10] K. Fernando and B. N. Parlett, *Accurate singular values and the differential qd algorithms.* Numerische Mathematik, 67:191-229, 1994.

[11] R. Harrison, *TCGMSG Manual.* The latest version of TCGMSG can be obtained via anonymous ftp from `ftp.pnl.gov:/pub/global/global2.2.tar.gz`.

[12] W. Kahan, *Accurate eigenvalues of a symmetric tridiagonal matrix.* Computer Science Dept. Technical Report CS41, Stanford University, Stanford, CA, July 1966 (revised June 1968).

[13] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide (second edition).* SIAM, Philadelphia, 1995, 324 pages.

[14] R.-C. Li, *Relative perturbation theory: (I) eigenvalue and singular value variations.* Technical Report UCB//CSD-94-855, Computer Science Division, Department of EECS, University of California at Berkeley, 1994(revised January 1996).

[15] J. H. Wilkinson, *The Algebraic Eigenvalue Problem.* Oxford University Press, Oxford, 1965.