A Parallel Algorithm for Solving the Complex Symmetric Eigenproblem on the Cray J-9x

Ilan Bar-On* Marcin Paprzycki*

Abstract

We consider the parallel performance of the recently proposed algorithm for the solution of the complex symmetric eigenproblem [5], on the Cray J-9x series of parallel shared memory vector computers. It is observed that implicit multitasking (based on the parallel BLAS kernels) does not lead to satisfactory performance and is similar to the parallel performance of the LAPACK routines for solving the complex Hermitian and complex general eigenproblems. We conclude that, although these algorithms are rich in matrix and vector operations, we have to apply more specific coarse grain techniques to achieve higher performance.

1 Problem Specification

A new generation of algorithms for studying chemical reaction modeling [15], [14] lead to the solution of a complex symmetric eigenproblem, i.e.

$$AV = V\Lambda, \quad V^T V = I, \quad \Lambda = Diag(\lambda_1 \cdots \lambda_N). \tag{1}$$

This problem is computationally expensive as the matrices are large (of order 7000 and more) and dense. In this paper we consider the direct solution of (1) by means of the new algorithm presented in [5]. The algorithm may be summarized as follows:

- Reduce A to a tridiagonal complex symmetric matrix T, i.e.

$$Q^T A Q = T, \qquad Q^T Q = I. \tag{2}$$

- Compute the spectral decomposition of T, i.e.

$$TW = W\Lambda, \quad W^T W = I, \quad \Lambda = Diag(\lambda_1 \cdots \lambda_N). \tag{3}$$

- Compute the original eigenvectors of A by backtransformation, i.e. $V = QW$.

In earlier work we have studied the sequential performance of this algorithm [3], [4]. The aim of this note is to discuss the initial results of our attempts at code parallelization. In Section 2 we shortly summarize the sequential performance of the new code and that of the related complex eigensolvers. Section 3 presents the results of our experiments on the Cray J-9x shared memory vector computers. Finally, Section 4 proposes the directions for future research.

*Dept. of Math and Comp. Sci., The University of Texas of the Permian Basin

2 Sequential performance

The complex Hermitian eigenvalue problem is one of the best known and most studied problems in numerical linear algebra. The related complex symmetric eigenvalue problem, on the other hand, has not received much attention till very recently. For example, Wilkinson in his comprehensive work on *The Algebraic Eigenvalue Problem* [16], comments only that:

COROLLARY 2.1. *An arbitrary square complex matrix* $A = (a_{ij})$ *is similar to a symmetric matrix.*

Proof. See Gantmacher [10][page 13].

Thus, he concludes, there is no reason to consider complex symmetric matrices separately. However, in many practical applications [2], [15], [14], one is often interested in diagonalizing very large dense complex symmetric matrices. The general eigensolvers routines that are currently used in computational practice are very time consuming (even on the fastest supercomputers). In an attempt to solve this problem, a new efficient algorithm that takes advantages of the symmetricity of the matrix has been developed [5]. The algorithm is similar to the real symmetric eigensolver with one significant difference. In the case of a breakdown in the reduction stage (2) additional recovery transformations have been introduced. Since in computational practice these additional transformations occur only rarely, the total complexity of the new algorithm remains similar to that for the complex Hermitian eigensolver. The new algorithm is therefore substantially faster than the general eigensolver.

In Table 1 we present the performance of the new algorithm on a single processor Cray J-916, and compare it to the performance of the standard methods, for matrices of order $n = 2200$. The first row represents separately the reduction (Reduce) stage of the algorithm (2), the computation of the eigenvalues (Eig), and eigenvectors(Vec) of the tridiagonal matrix (3), and the final calculation of the eigenvectors (Recover) of the original matrix. We report the running times (in seconds) as well as the Mflop rates (in parentheses) of each stage. In the second row we have combined the running times for computing the eigenvalues and the calculation of the eigenvectors. Finally, the last two rows present the speedup of the new algorithm over the LAPACK routines, CGEEVX (the general solver) and CHPEVX (the Hermitian solver).

TABLE 1
Complex Symmetric Eigensolver.

n=2200	Reduce	Eig	Vec	Recover
New	324(177)	163(14)	154(14)	489(172)
Total	487		1130	
General	~6.0		~6.3	
Hermitian	~0.6		~1.5	

Observe that the code reaches $\sim$ 175 Mflops, about $\sim$ 91% of the practical peak performance of the machine [13], on the most time consuming stages of the algorithm. However, due to the low performance of the middle stages, the overall performance is somewhat unsatisfactory. We note here, that this behavior may be traced to our use of old scalar codes by Cullum and Willoughby [7], CMTQL1 for the calculation of the eigenvalues, and INVERM1 for the calculation of the corresponding eigenvectors (the only

codes currently available, see [3], [4] for more details). Nonetheless, the speedup with respect to the general eigensolver is quite impressive (~ 6). Compared to the Hermitian solver, the algorithm performs similarly in the Reduce stage, and the low speedup (~ 0.6) when computing the eigenvalues is caused by the inefficiency of the Eig stage. However, surprisingly, the new algorithm outperforms the Hermitian solver by ~ 1.5 times when computing the complete eigensystem. This fact can be attributed to the implementation details of the Recover stage.

3 Parallel Performance

In our initial attempt at parallelization we have decided to use the automatic multitasking facility available on the Cray's FORTRAN compiling system. This decision was based on two assumptions. First, the new code has been implemented in such a way that **all** computationally intensive stages are performed by the BLAS routines [12], [9], [8]. Second, we wanted to compare the parallel performance of the new code with the corresponding LAPACK [1] routines and the parallelization of most of the LAPACK routies is based on the parallelism in the optimized BLAS kernels [6], [11].

The experimental data for the parallel performance was collected on three Cray J-9x machines; a 5 and a 16 processor computers in the Computation Center University of Texas in Austin and a 16 processor machine at the Supercomputing center in Ben Gurion University, Israel. Neither of the machines has been used in a benchmarking mode and their load has varied from very light, to relatively heavy. To establish the run-times reported here multiple runs of each experiments have been performed and a combination of *timef*, *second* and *mttimes* utilities has been used. Obviously, the results reported for the larger number of processors do not represent the best possible performance, but they are reliable enough to establish the general picture of the performance of the codes.

In the first series of experiments we have investigated the parallel performance of finding the eigenvalues only. The results for the matrix of size $n = 2200$ are summarized in Table 2 and presented in Figure 1.The first series of experiments with the Hermitian solver CHPEVX (HermPac) and the new code (New) indicated no parallelization. We have therefore experimented also with the unpacked Hermitian eigensolver routine CHEEVX(HermUnp). Finally, the results of experiments with the general solver CGEEVX are reported (General).

TABLE 2
Eigenvalue Calculation

n=2200	General	HermPac	HerUnp	New
1 procs.	2730	303	302	487
16 procs	1387	303	96	487
Speedup	1.968	1	3.14	1
Efficiency	12%	6%	19%	6%

We observe some speedup for the general solver, and for the unpacked version of the Hermitian solver. The packed Hermitian solver, as well as the new code do not parallelize at all. The reason for this is that both codes use the packed storage representation of the data and thus call the packed storage based BLAS kernels. These kernels have not been parallelized by the Cray [11]. Note however, that the cost of using the unpacked version is relatively high, e.g. as the matrices can easily reach the size 7000×7000 the

unpacked memory scheme costs $\sim 200MBytes$ of additional memory (assuming 8-byte double precision real numbers). Still, the results indicate that there exist at least two possible approaches for the parallelization of the new code. First, it is possible to use the unpacked storage representation and apply the parallelized BLAS kernels. Second, it is also possible to develop optimized parallel versions of the appropriate packed storage BLAS kernels. This latter approach could also benefit the packed storage version of the Hermitian solver.

In the second series of experiments we consider the parallel performance of these routines when computing the complete eigensystem. Table 3 presents these results for the general solver, the packed (HermPac) and unpacked (HermUnp) versions of the Hermitian solver and two results for the new code. The first result represents the performance of the whole algorithm (New), the second the performance of just the backtransformation stage(NewBack). As previously the speedup and efficiency are reported.

TABLE 3
Eigenvector Calculation

n=2200	General	HermPac	HermUnp	New	NewBack
1 procs.	6724	1533	1531	1143	495
16 procs	3406	1313	1079	769	104
Speedup	1.97	1.16	1.41	1.48	4.76
Efficiency	12%	7%	9%	9%	30%

As expected, when the complete eigensystem is calculated the general solver behaves quite similarly to the case when only the eigenvalues were sought. The results presented in Tables 2 and 3 and in Figure 1 show clearly that the general solver can successfully utilize only 2 processors and any further increase in the processor number does not lead to performance increases of either of its stages.

The unpacked Hermitian solver only slightly outperforms the packed version and both these solvers are much slower than the new code. This result indicates not only that the single processor performance of the backtransformation stage of the Hermitian solver is rather inefficient (see Table 1 above), but also that its parallel performance leaves a lot to be desired. As for the new code, parallel speedup is obtained only in the last, backtransformation stage. Here the performance is surprisingly good considering that the experiments have been executed on a relatively busy machine.

Finally, Figure 1presents the speedups of the eigenvalue calculation and the total execution are represented for the matrix of order $n = 2200$ and for $1, 2, \ldots, 16$ processors. The results of the eigenvalue calculation for the packed Hermitian solver and the New code are omitted as no speedup has been achieved. The results for the bactransformation stage of the new algorithm are included separately.

The results confirm that there are only two algorithms that lead to speedup. The unpacked version of the Hermitian eigensolver when only the eigenvalues of the system are calculated and the backtransformation stage of the new algorithm.

4 Conclusions

We have presented initial results of our attempts at parallelization of the new algorithm for solving the complex symmetric eigenproblem. Our results confirm that using implicit multitasking yields very little gain in terms of speedup and that more explicit parallel

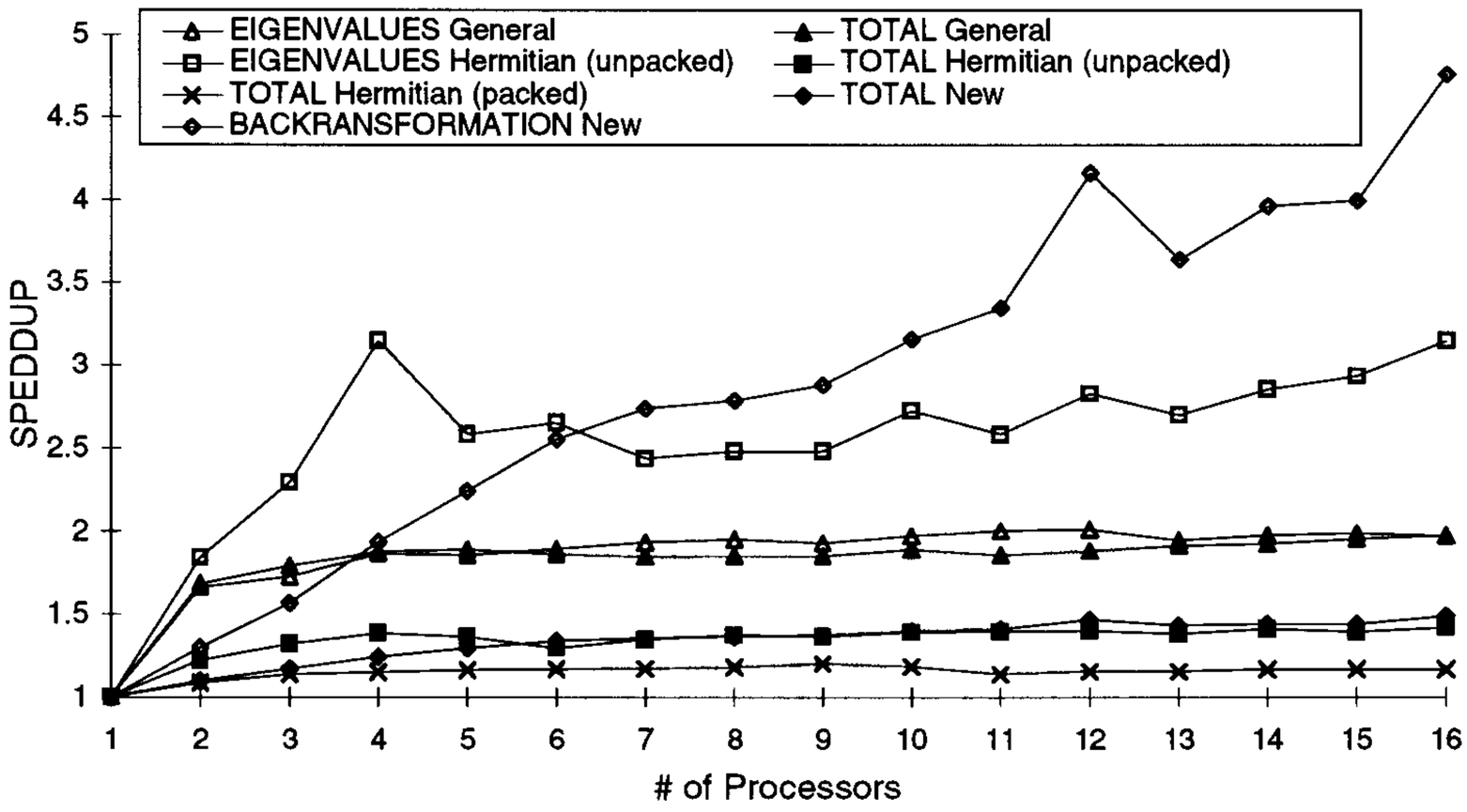

FIG. 1. *Multiprocessor performance comparison.*

techniques should be used. The basic directions of our research are currently as follows:

- experiments with unpacked versions of the new code
- development parallel BLAS kernels for the packed storage representation
- development of the new vector-parallel codes for the complex symmetric tridiagonal matrices
- experiments on scalar parallel shared memory computers
- introduction of explicit parallelization into the new code

Acknowledgments We thank Prof. Nimrod Moiseyev, from the Chemistry Department of the Technion, for the computer time grant.

References

[1] E. Anderson et. al., *LAPACK Users' Guide*, SIAM, 1992.
[2] Z. Bai, D. Day, and Q. Ye, *ABLE: an adaptive block lanczos method for non-Hermitian eigenvalues problems*, Tech. Rep. 95-04, University of Kentucky, Feb. 1996.
[3] I. Bar-On and M. Paprzycki, *An Efficient Algorithm for Finding Eigenvalues of Complex Symmetric Matrices.* submitted for publication, 1996.
[4] I. Bar-On and M. Paprzycki, *High performance solution of a complex symmetric eigenproblem.* submitted for publication, 1996.
[5] I. Bar-On and V. Ryaboy, *Fast diagonalization of large and dense complex symmetric matrices, with applications to quantum reaction dynamics*, Siam J. Sci. Stat. Comput., (1996). To appear.
[6] Cray Research, *UNICOS Math and Scientific Library Reference Manual*, (1995).
[7] J. Cullum and R. Willoughby, *Lanczos Algorithms for Large Symmetric Eigenvalues Computations*, Birkhauser Boston, 1985.
[8] J. J. Dongarra, J. Du Croz, I. Duff and S. Hammarling, *A Set of Level 3 Basic Linear Algebra Subprograms*, Technical Report ANL-MCS-TM57, Argonne National Laboratory, (1988)

[9] J. J. Dongarra, J. Du Croz, S. Hammarling and R. J. Hanson, *An Extended Set of FORTRAN Basic Linear Algebra Subprograms*, ACM Transactions on Mathematical Software, 14 (1988)

[10] F. Gantmacher, *Applications of the theory of matrices*, John Wiley & Sons, 1959.

[11] C. Hempel, Silicon Graphics Inc., *personal communication.*

[12] C. L. Lawson, R. J. Hanson, D. R. Kincaid and F. T. Krogh, *Basic Linear Algebra Subprograms for FORTRAN Usage*, ACM Transactions on Mathematical Software, 5, (1979)

[13] M. Paprzycki and C. Cyphers, *Multiplying matrices on the Cray - Practical considerations*, CHPC Newsletter, (1991).

[14] V.Ryaboy and N.Moiseyev, *Cumulative reaction probability from siegert eigenvalues: model studies*, J. Chem. Phys., 98 (1993).

[15] V.Ryaboy and N.Moiseyev, *Three dimensional study of predissociation resonances by the complex scaled discrete variable representation method: HCO/DCO*, J. Chem. Phys., 103 (1995).

[16] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, 1965. Reprinted in Oxford Science Publications, 1988.