Performance of Parallel Eigensolvers on Electronic Structure Calculations II

Robert C. Ward[1,2] and Yihua Bai[1,2]

Technical Report UT-CS-06-572[3]

University of Tennessee

September 2006

Abstract. Many models employed to solve problems in quantum mechanics, such as electronic structure calculations, result in both linear and nonlinear eigenproblems. Solutions to the nonlinear problems, such as the Self-Consistent Field method, typically involve iterative schemes requiring the solution of a large symmetric linear eigenproblem during each iteration. This paper evaluates the performance of various popular and new parallel symmetric linear eigensolvers applied to such eigenproblems in electronic structure calculations on the IBM distributed memory supercomputer at the Oak Ridge National Laboratory. Results using established routines from ScaLAPACK and vendor optimized packages are presented, as well as from three recently developed parallel eigensolvers, two implementations of the Multiple Relatively Robust Representations algorithm and the block divide-and-conquer algorithm. This paper updates an earlier version of this work reported in University of Tennessee Technical Report UT-CS-05-560 by Ward, Bai & Pratt in 2005 [33].

1. Introduction. The problem of describing the motion of N electrons in the field of M fixed nuclear point charges is a central problem in quantum chemistry. It translates into the task of finding and describing approximate solutions of the electronic Schrödinger equation

$$H\Phi = E_O\Phi . \qquad (1)$$

The solutions to this equation involving the electronic Hamiltonian operator H are the electronic wave functions Φ, which describe the motion of electrons, and the electronic energy operator E_O. The wave function for a single particle is called an orbital.

Using the Hartree-Fock approximation [29, 31] in the electronic Schrödinger equation leads to the nonlinear Hartree-Fock equation, a spatial integro-differential equation for the orthonormal Hartree-Fock orbitals and the corresponding orbital energies. The N orbitals with the lowest energies are called the occupied orbitals. In practice, the solutions of this equation are approximated by introducing a finite set of n basis functions, expanding the unknown molecular orbitals in terms of this basis, and converting the Hartree-Fock equation to a set of algebraic equations. The problem of computing the molecular orbitals then reduces to the problem of computing the matrix C of expansion coefficients, which can be formulated as the *Roothaan equations*

---

[1]Department of Computer Science, University of Tennessee, 203 Claxton Complex, 1122 Volunteer Blvd., Knoxville, TN 37996-3450.

[2]This work was partially supported by Oak Ridge National Laboratory's Computing and Computational Sciences Directorate and the University of Tennessee's Science Alliance Program.

[3]Available from: http://www.cs.utk.edu/~library/TechReports.html .

$$F(C)\,C = S\,C\,E \qquad (2)$$

with the Hermitian (usually real symmetric) n x n matrices S (*overlap matrix*) and F (*Fock matrix*). S is positive definite with unit entries along the diagonal, and its off-diagonal entries satisfy $|S_{ij}| < 1$ for $i \neq j$. E is a diagonal matrix that contains the orbital energies along the diagonal, and C contains the expansion coefficients columnwise.

1.1 The Self-Consistent Field Method. The Roothaan equations (2) establish a generalized eigenproblem with the unknowns E and C as the eigenvalues and eigenvectors, respectively. Since F depends on the coefficient matrix C, this is a nonlinear eigenproblem. Its solution is a very central and time-consuming task arising in many quantum chemistry applications. We have described the problem as it is normally derived in Hartree-Fock theory, however, very similar equations occur in density functional theory [26, 14] and in semi-empirical quantum chemistry [28, 10].

The standard approach in quantum chemistry for solving (2) is the *self-consistent field (SCF)* method, summarized in Algorithm 1.1. After reduction of the generalized nonlinear eigenvalue problem (2) to a standard nonlinear eigenproblem, for example, using a Cholesky factorization of the overlap matrix S, this nonlinear problem is transformed into a linear eigenproblem using an initial guess C_0 for the expansion coefficients. The linear eigenproblem is solved, the new expansion coefficients C_1 are computed from the eigenvectors of the linear eigenproblem, and a new linear eigenproblem is formulated. This procedure is iterated until *self-consistency*, i.e., convergence, has been achieved. As a convergence criterion, it is common to require that the difference between two successive values of the total energy $\left(\left| \sum_{i=1}^{n} (E_k)_{ii} - \sum_{i=1}^{n} (E_{k-1})_{ii} \right| \right)$ or the expansion coefficients $\left(\|C_k - C_{k-1}\| \right)$ be bounded by a tolerance δ. A value of $\delta = 10^{-6}$ is adequate for most purposes [31].

Input: *guess initial values* C_0
1. *factorize* $S = U\,U^T$
2. *transform (2) into a standard problem* $A(C)\,V = V\,E$
3. **repeat** $k = 1,2,\ldots$
   - *(i) compute* $A(C_{k-1})$
   - *(ii) solve* $A(C_{k-1})\,V = V\,E_k$
   - *(iii) compute* $C_k = U^{-T}\,V$

   until *converged*

Output: *electronic energy, orbitals*

Algorithm 1.1 Self-Consistent Field Method

The iterative Algorithm (1.1) does not always converge, for example, if the initial guess is poor. Various techniques have been suggested for ensuring or accelerating convergence [22].

1.2 Synopsis. The purpose of this paper is to report on the performance of various parallel standard symmetric eigensolvers as they solve linear eigenproblems in electronic structure calculations. Most of the test matrices in this study result from applying the SCF method to molecules from various types of materials, then solving the linear eigenproblem given by Step 3(ii) in the first iteration of Algorithm 1.1, thereby updating a previous similar study [33]. One family of test matrices, the Tetra-Cyanoquinodimethane (TCNQ) family, results from transforming the Schrödinger equation (1) directly into a linear eigenproblem. The computer used will be the IBM pSeries distributed memory supercomputer located in the National Center for Computational Sciences (NCCS) [1] at the Oak Ridge National Laboratory (ORNL).

The algorithms are briefly described in Section 2, the test matrices in Section 3, and the supercomputer and testing environment in Section 4. Section 5 presents the test results, and some conclusions are given in Section 6.

2. Parallel Eigensolvers. For our performance tests, we include the two parallel eigensolvers that are readily available to the users of the distributed memory supercomputers located in ORNL's NCCS. Both of these have been tuned for improved performance through the use of BLAS and PBLAS optimized by the computer vendor. In addition, we will include parallel implementations of two recent algorithms: the method of Multiple Relatively Robust Representations and the block divide-and-conquer algorithm. The salient features of each algorithm will be given below with algorithmic and parallel implementation details provided in the cited references.

2.1 PDSYEVD. The parallel eigensolver, PDSYEVD, used in our experiments is found in ScaLAPACK [8] and is based upon the divide-and-conquer algorithm [9]. It consists of the classical three steps: reduction to symmetric tridiagonal form, eigen-decomposition of the tridiagonal matrix, and back-transformation of the eigenvectors. The eigen-decomposition of the tridiagonal matrix is computed by the Tisseur-Dongarra parallel implementation [32] of the divide-and-conquer algorithm using the Gu-Eisenstat method [19] for stably computing the eigenvectors. The reduction to tridiagonal form does not incorporate some of the more recent methods [7, 15] designed to increase the number of BLAS3 operations. Algorithms implementing all three steps use a vendor-optimized version of BLAS. Memory requirements for PDSYEVD involve $4n^2$ (A, V, and $2n^2$ workspace) + $O(n)$.

2.2 PDSYEVX. The parallel eigensolver, PDSYEVX, used in our experiments is found in PESSL [3], which is a library of parallel solvers for scientific problems supplied by IBM. PESSL incorporates optimizations for the intended computational platform. PDSYEVX also consists of the classical three steps described above but uses a different algorithm for the eigen-decomposition of the symmetric tridiagonal matrix. The bisection method [23] is used for computing its eigenvalues and inverse iteration (with re-orthogonalization for tightly clustered eigenvalues) [23] is used for computing its

eigenvectors. The same algorithms as in PDSYEVD are used for the reduction and back-transformation steps. Memory requirements for PDSYEVX also involve $4n^2$ (A, V, and $2n^2$ workspace) + $O(n)$, although exact requirements are unknown in advance due to the unknown number of eigenvectors that must be re-orthogonalized.

2.3 PDSYEVR and PMR3. Significant progress has been made in computing the eigensystem of symmetric tridiagonal matrices. The method of Multiple Relatively Robust Representations (MRRR) [25, 11, 24, 12, 13] has greatly improved the efficiency of these solvers while maintaining accuracy and obtaining eigenvector orthogonality. The MRRR provides an $O(n^2)$ algorithm for computing the n eigenpairs to full accuracy and the eigenvectors to numerical orthogonality. Since the solution to the full eigenvector problem generally requires the computation of n^2 values, this algorithm is generally considered optimal.

We test two parallel implementations of the MRRR algorithm: PDSYEVR and PMR3. PDSYEVR, as described in [2], is a near-final version of the implementation expected to be in the next release of ScaLAPACK. PMR3, as described in [6], is the first parallel implementation of an MRRR algorithm and was obtained from the PLAPACK researchers at the University of Texas. The same basic algorithms as used in PDSYEVD and PDSYEVX, but with different implementations, are used in these two algorithms to reduce the full matrix to tridiagonal form and back-transform the eigenvectors of the tridiagonal matrix to those of the original matrix [18]. Memory requirements for PDSYEVR currently involve $4n^2$ (A, V, and $2n^2$ workspace) + $O(n)$, but future ScaLAPACK implementations should be able to reduce this to its theoretical requirement of $2n^2$ (A and V) + $O(n)$ as the PMR3 implementation requires.

2.4 PDSBTDC. The recently developed block divide-and-conquer algorithm [17, 16] has proven very attractive for computing eigensystems of symmetric, block tridiagonal matrices with reduced accuracy requirements, that is, accuracy less than machine precision, or with most of its largest elements near the diagonal, for example, representing a physical system with strong locality properties. This algorithm has been used with good results in electronic structure calculations[16]. A parallel implementation of this algorithm [4] has recently been developed. Parallel algorithms for reducing dense matrices to block tridiagonal form [5] have also been developed so that PDSBTDC can now be tested on the same matrices as the other algorithms described above; however, we will use reduced accuracy of 10^{-6} for this algorithm since that is its typical application and is especially relevant in electronic structure calculations. Note that all matrices are trivially in block tridiagonal form with two blocks, but in this paper we will only consider matrices with the number of diagonal blocks greatly exceeding two. As expected, PDSBTDC requires similar storage to PDSYEVD, that is $4n^2$ (A, V, and $2n^2$ workspace) + $O(n)$.

3. Test Matrices. The test matrices used in this eigensolver performance study come from various molecule families and derivation schemes. A brief description of each is given in the subsections below along with a picture of the normalized Fock matrix or Hamiltonian matrix used in our tests from each family. Except for the impure

hydrogen and TCNQ families where the zero-nonzero structure is shown, the picture uses a color to indicate the magnitude of the exponent of the element in that matrix position. The color-coded scale bar to the right of the picture shows the value of the exponent. Pictures of different matrices in the same family are very similar. The major difference in test matrices in most of the families comes from the number of molecules in the model and result in matrices of different orders. We report in this paper results from testing only the largest matrix in each family - except for the TCNQ family, which is our source for very large matrices. Note that the orders of the matrices range from 3,014, very small for parallel computers, to 63,504, very large.

3.1 Alkane Family. The alkane family consists of acyclic hydrocarbons in which the molecule has the maximum number of hydrogen atoms – hence has no double bonds. The general formula for alkanes is C_nH_{2n+2}. The nonlinear eigenproblem for this family has been derived from the semi-empirical method CNDO, and as such, does not have an overlap matrix. Thus, there is no normalization (steps 1 and 2 of Algorithm 1.1) of the Fock matrix.

The Fock matrix from the $C_{502}H_{1006}$ alkane with $n = 3014$ was included in our performance study. Figure 3.1 provides a picture of the element magnitudes.

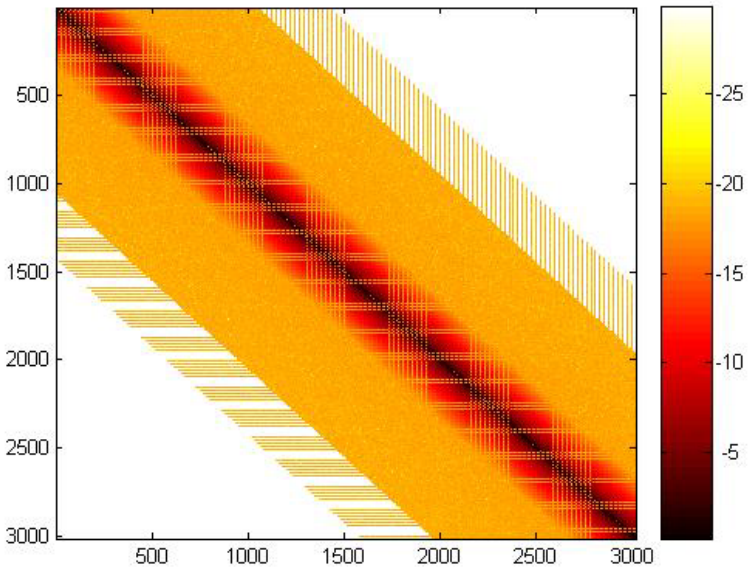


Figure 3.1 Element Magnitudes of the Initial Fock matrix for the $C_{502}H_{1006}$ Molecule in the Alkane Family

3.2 Polyalanine Family. The matrices in this family were formed from modeling polypeptides made solely from alanine. Linear polyalanine chains of differing lengths were constructed, and then a classical force field was used to randomize the chain conformations so that the molecules were no longer linear. The MINDO method was then used to construct the Fock matrices, all of which had a bandwidth of 79.

Our performance study includes the Fock matrix from a polyalanine chain of length 200, resulting in a matrix size of 5027. Figure 3.2 shows a picture of this matrix.

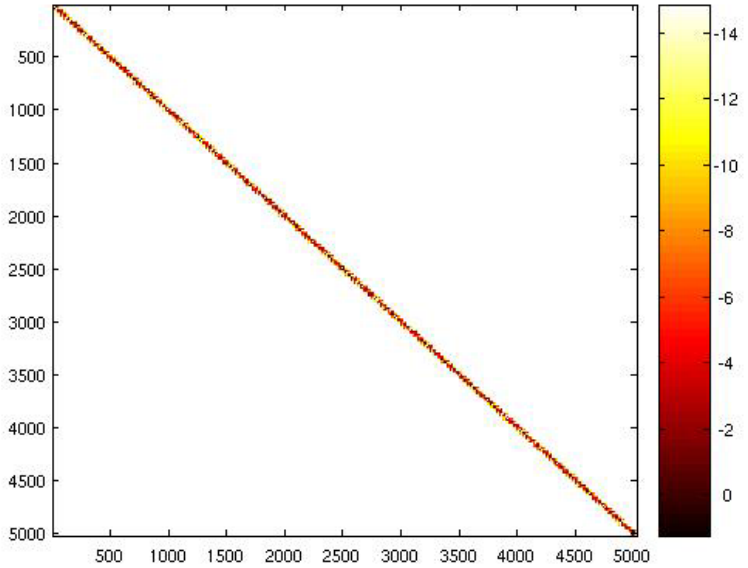


Figure 3.2 Element Magnitudes of the Fock matrix for the Polyalanine Chain of Length 200

3.3 Silicon Crystal Family. This family of test matrices has been generated using the PBE form [27] of Density Functional Theory on Silicon crystals with differing numbers of atoms. We denote the different problems in the family by the number of unit cells in each of the x, y, and z directions. Thus, the 111 problem has one unit cell in each of the directions, and the 432 problem has 4 unit cells in the x direction, 3 in the y direction, and 2 in the z direction. Each unit cell has 8 atoms using 104 basis functions from the Double Zeta basis set. Thus, the 111 problem models 8 atoms with 104 basis function resulting in an eigenproblem of order 104, and the 443 problem models 384 atoms with 4992 basis functions resulting in an eigenproblem of size 4992.

We include the Fock matrix derived by 544 Silicon crystals in our tests, which yields a matrix of order 8320. Figure 3.3 shows a picture of this test matrix.

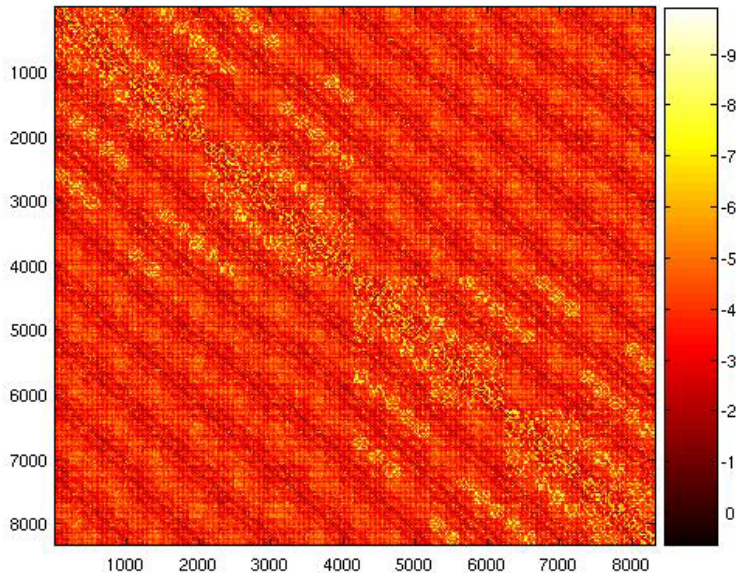


Figure 3.3 Element Magnitudes of the Initial Normalized Fock matrix for the 544 Lattice in the Silicon Crystals Family

3.4 Hydrogen Molecules with Impurities. These test matrices are derived from a finite-basis representation of the vibrational Hamiltonian of a randomly close, packed solid sample of hydrogen molecules H_2 in which 0.1% of the molecules have been replaced by impurities [20]. The solid is represented by a collection of 21 x 24 x 26 molecules on a lattice with periodic boundary conditions imposed in all three dimensions. The off-diagonal elements represent couplings between nearby H_2 molecules and decrease in magnitude the further they appear from the diagonal. Different matrices in this family are produced by including more distant neighbors in the model. All the matrices are of order 13,104.

The matrix used in our tests from this family involves multiple layers of molecular couplings. The matrix contains 716,273 nonzero elements (that is, 0.42% sparse), and all the nonzero elements are between -10 and -0.015. Its zero-nonzero structure is given in Figure 3.4. We test our dense parallel solvers on this matrix in spite of its obvious sparsity. Other matrices in this family are more dense but, as expected, produce very similar timing results.

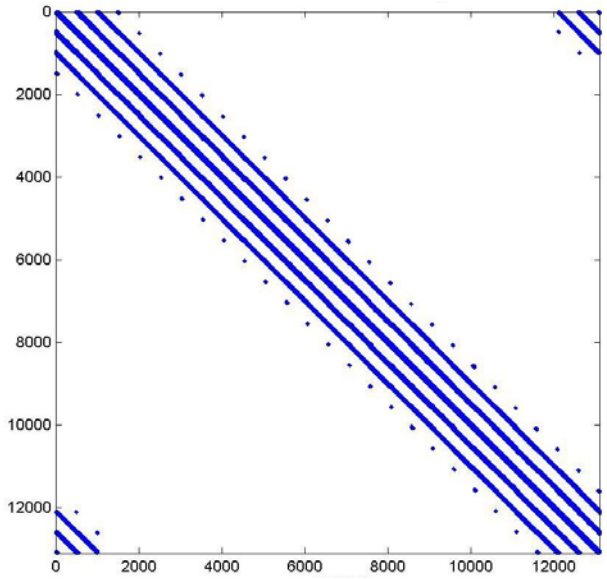


Figure 3.4 Zero-Nonzero Structure of Hamiltonian from Impure Hydrogen Lattice with 0.42% sparsity (Blue for nonzero, White for zero)

3.5 trans-Polyacetylene Family. Polyacetylene (PA) has been the subject of intensive investigation for many years. It shows a very fast response time upon laser excitation, making it a candidate for fast-optical switching and optical computation. Our test matrix comes from the trans-form of PA with a C-H chain sufficiently long that end effects may be ignored. The SSH Hamiltonian[30], which is a tight-binding approximation and includes only nearest neighbor atoms, is used within the Hartree-Fock approximation framework. As a tight-binding approximation, each cell unit has only one basis function; therefore, the size of the matrix is the number of atoms being modeled.

Our test matrix from this family models 16,000 atoms. Figure 3.5 presents a picture of the magnitude of the elements in the initial Fock matrix produced from this problem.

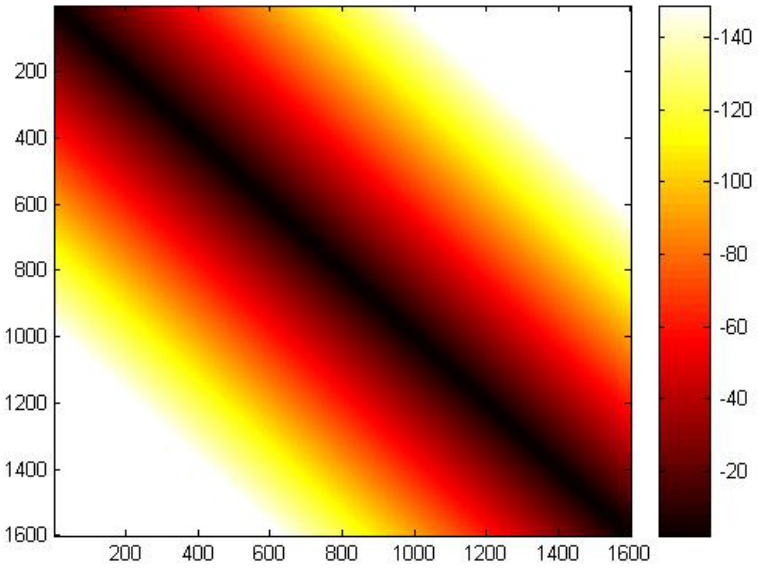


Figure 3.5 Element Magnitudes of the Fock matrix for a trans-Polyacetylene with 16,000 atoms

3.6 Tetra-Cyanoquinodimethane (TCNQ) Family. TCNQ is an electron-accepting molecule used in organic conductors. The test matrices are generated from applying the Hubbard model [21] to a one-dimensional TCNQ chain where the electron correlation is considered important. The Hubbard model is based on a tight-binding approximation with the nearest neighboring electron interaction included. For our generated test matrices, the TCNQ chain length L is 10, which means there are 10 atoms, and the electron filling is one-half, indicating that we have 10 electrons in the system. The chain length L determines the size of the Hamiltonian matrix, which is $\{L!/[(L/2)!(L/2)!]\}^2$. For example, with $L = 10$, the corresponding matrix size is 63,504.

To reduce the matrix size, Hamiltonian matrices may be truncated via a density-matrix renormalization group method [34] where only the most important eigenstates are reserved and used to generate a density matrix. The truncation is controlled by the number of states kept in the density matrix. With less truncation (more states), more electron correlation effects are included, which leads to a larger matrix.

We use two test matrices from this family: the full Hamiltonian without truncation (n = 63,504) and the truncated Hamiltonian keeping 120 states in the density matrix (n = 29,296). Figures 3.6 and 3.7 show the zero-nonzero structure for these matrices. As with the impure hydrogen matrices, we apply our dense eigensolvers to these matrices even though sparse or band solvers may be more appropriate.

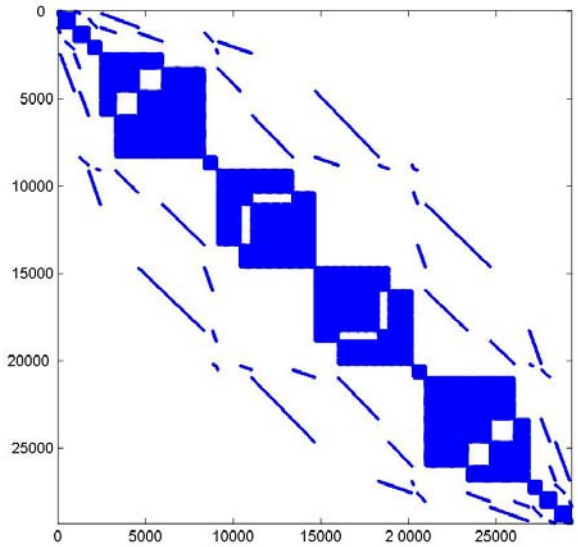


Figure 3.6 Zero-Nonzero Structure of 120-State Truncated Hamiltonian from TCNQ (Blue for nonzero, White for zero)

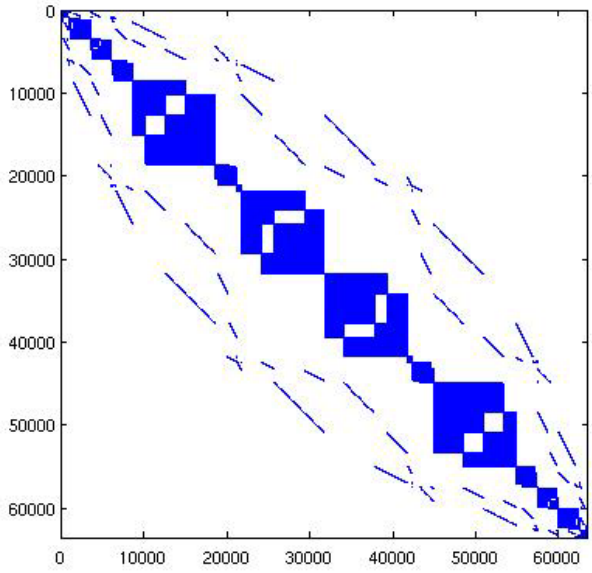


Figure 3.7 Zero-Nonzero Structure of the full Hamiltonian from TCNQ (Blue for nonzero, White for zero)

4. Supercomputer and Testing Environment. The test cases were run on the IBM pSeries distributed-memory supercomputer in the National Center for Computational Sciences at the Oak Ridge National Laboratory. The system, installed in 2002 and nicknamed Cheetah, consists of twenty-seven p690 nodes. Each node contains sixteen chips with two 1.3 GHz IBM Power4 processors per chip for a total of 864 processors. The nodes are connected via IBM's Federation interconnect. With each processor having a peak rating of 5.2 GFlops, the peak computational power of the system is 4.5 TFlops. The system has a peak LINPACK rating of 2.3 TFlops.

Twenty of the nodes have 32 GBytes of SMP memory each, five nodes have 64 GBytes, and two have 128 GBytes; thus, the total available main memory on the system is 1.2 TBytes. Access to data residing off-node is via the interconnect at a slower speed, resulting in a non-uniform memory access (NUMA) system. In addition, 14 nodes have 160 GB of local temporary disk space.

The Power4 memory hierarchy consists of 3 levels of cache with prefetching. The first and second levels are on-chip. The split L1 instruction and data caches are 128 KBytes and 64 KBytes respectively or 64 KBytes and 32 KBytes respectively per processor. The unified L2 cache is 1.5 MBytes and is shared between the 2 processors. The L3 cache also shared by the 2 processors is 32 MBytes and is located on a separate chip.

Our tests on Cheetah invoked the Fortran versions of PDSYEVD, PDSYEVX, PDSYEVR and PDSBTDC through version 8.1 of IBM's `xlf` compiler and C versions of PMR3 and PLAPACK through version 6 of IBM's `xlc` compiler. Both compilers were set to the default 32-bit compile mode and linked to the 32-bit PESSL library. The compiler options for PDSYEVD, PDSYEVX, PDSYEVR and PDSBTDC were:

```
-O4 –qstrict -qarch=auto -qcache=auto –qtune=auto
          -bmaxdata:0x70000000,
```

and for PMR3 and PLAPACK:

```
-O4 –qstrict -qtune=pwr4 -qarch=pwr4
          -bmaxdata:0x70000000.
```

5. Test Results. The wall-clock times (in seconds) for the different algorithms using consecutive powers of 2 as the number of processors are given in tabular form and graphically for each test matrix. For the smaller matrices, we start with 4 processors and go up to 512 and start with more processors for the larger matrices.

PMR3 ran into some difficulty on some of the test cases and did not complete the computations, ending with an error. A reasonable amount of effort was spent trying to solve the problems without success. The errors did not consistently appear in any one of the three steps (reduction, tridiagonal solver or back-transformation) and are likely a portability problem since some of these tests ran successfully on different clusters. These tests are shown with a DNF (did not finish) tag for the wall clock time. We anticipate these problems will disappear if PLAPACK is formally ported to the Cheetah architecture and results for those cases would be consistent with its performance on the other tests.

5.1.Timing Results. Table and Figure 5.1 present the timing results for $C_{502}H_{1006}$. The size of this matrix ($n = 3014$) is rather small for parallel computation, and assigning more than 16 processors to the computation yields very little change in the required time. All the algorithms reach the point in our tests where adding processors actually increases the time due to the overhead of parallel computation.

The three ScaLAPACK algorithms performed roughly the same with both PMR3 and PDSBTDC requiring longer computational time. Even though some benefiting magnitude structure exists in the matrix as illustrated in Figure 3.1, the block-tridiagonalization algorithm transforming the matrix into block tridiagonal form in PDSBTDC resulted in off-diagonal blocks with rather large ranks. Deflation could not offset this high computational complexity. The poor performance of PMR3 is almost solely a result of the time required to transform the full matrix into tridiagonal form, as shown in Table 5.8 in Section 5.2.

# procs	PDSYEVX	PDSYEVD	PDSYEVR	PMR3	PDSBTDC
4	21.8	19.1	21.6	37.4	63.0
8	11.5	11.2	12.7	25.2	42.3
16	7.2	7.9	8.5	DNF	26.0
32	5.0	5.5	6.5	DNF	15.3
64	4.2	4.5	5.5	DNF	12.5
128	4.0	4.2	5.0	11.9	10.2
256	3.9	4.5	5.6	14.5	8.9
512	5.0	5.4	6.6	17.1	9.0

Table 5.1 Wall Clock Time in Seconds for $C_{502}H_{1006}$

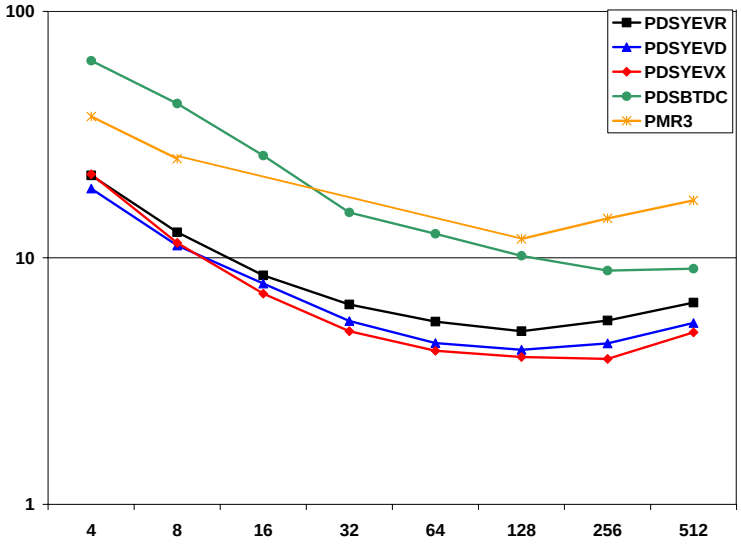


Figure 5.1 Plot of Wall Clock Time for $C_{502}H_{1006}$

Table and Figure 5.2 present the timings for a polyalanine chain of length 200 yielding a matrix of size 5027. Again, this matrix is rather small for parallel computation, and significant reductions in time do not result when using more than 32 processors.

The ScaLAPACK algorithms again perform roughly equivalently. The magnitude structure of the matrix is sufficiently "heavy" around the diagonal that PDSBTDC performs very well on this matrix. Again, PMR3 suffers from poor scalability in the reduction-to-tridiagonal step.

# procs	PDSYEVX	PDSYEVD	PDSYEVR	PMR3	PDSBTDC
4	105.2	75.6	88.4	85.2	49.2
8	61.4	47.1	48.3	51.3	23.0
16	33.0	31.1	32.7	DNF	16.2
32	18.8	17.6	19.7	DNF	8.8
64	12.3	13.4	15.5	DNF	6.3
128	9.6	10.0	12.1	29.3	5.1
256	8.4	10.0	12.0	32.5	3.5
512	9.4	10.4	12.6	38.2	4.8

Table 5.2 Wall Clock Time in Seconds for Polyalanine Chain of Length 200

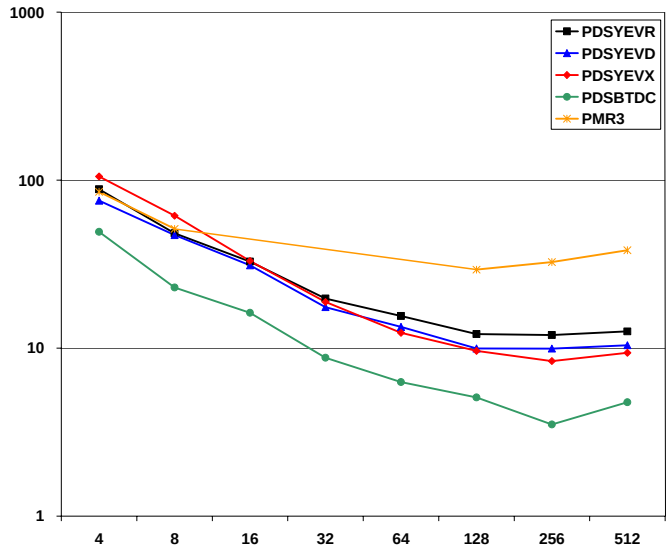


Figure 5.2 Plot of Wall Clock Time for Polyalanine Chain of Length 200

The Silicon crystal using a 544 lattice results in a matrix of size 8320, which begins to stress the available memory on standard serial computers required by eigensystem algorithms. Results for this matrix are shown in Table and Figure 5.3 and are similar to the results from the alkane family matrix except PMR3 is a competitive algorithm for small number of processors. Similar algorithmic results are also obtained from the hydrogen molecule with impurities as shown in Table and Figure 5.4. This matrix is of order 13,104, has 716,273 nonzeros and could be called sparse, although it is treated as a dense matrix in this study. On both matrices PMR3 poor performance again results from the poor scalability of the reduction step, and PDSBTDC's performance suffers from the large ranks of the off-diagonal blocks. Note that PDSBTDC appears to scale slightly better than the three ScaLAPACK algorithms. This is likely due to the larger computational complexity of PDSBTDC.

# procs	PDSYEVX	PDSYEVD	PDSYEVR	PMR3	PDSBTDC
4	595.7	408.7	363.2	351.6	959.6
8	277.3	208.6	222.8	204.1	581.0
16	128.2	124.9	127.1	DNF	385.6
32	68.2	68.6	70.8	87.2	170.3
64	37.5	44.9	48.9	DNF	97.8
128	25.3	27.9	33.9	73.4	63.4
256	19.3	22.9	29.1	80.5	43.5
512	19.1	21.2	27.4	87.8	34.8

Table 5.3 Wall Clock Time in Seconds for 544 Silicon Crystal Lattice

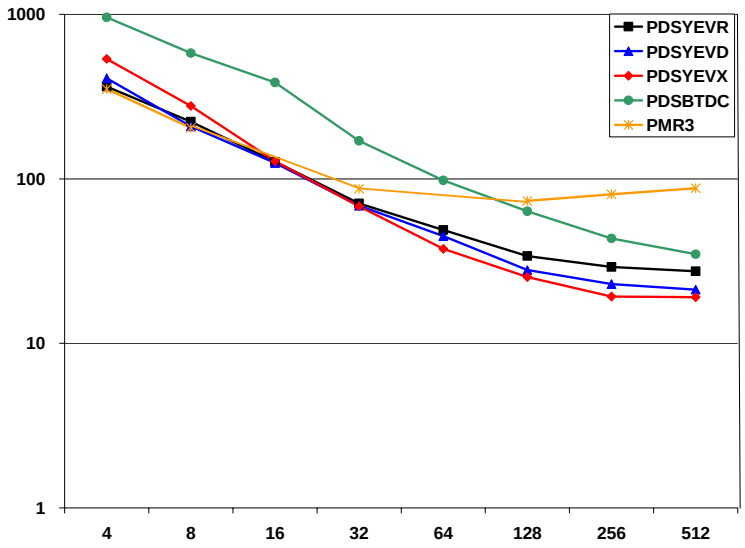


Figure 5.3 Plot of Wall Clock Time for 544 Silicon Crystal Lattice

# procs	PDSYEVX	PDSYEVD	PDSYEVR	PMR3	PDSBTDC
8	1271.1	969.2	809.2	890.5	3763.0
16	685.7	466.0	492.5	487.1	2071.8
32	282.4	269.7	258.6	284.4	639.1
64	127.6	151.2	155.2	DNF	334.9
128	74.5	78.0	86.0	169.6	175.8
256	48.3	59.7	68.7	179.0	110.2
512	41.6	45.0	58.8	196.7	76.3

Table 5.4 Wall Clock Time in Seconds for Impure Hydrogen

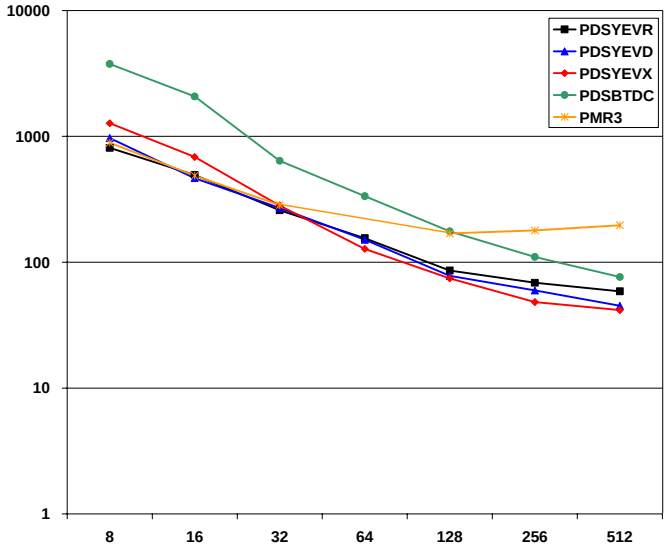


Figure 5.4 Plot of Wall Clock Time for Impure Hydrogen

The trans-polyacetylene family produces a very interesting case. For these matrices, there is considerable magnitude structure with all the larger elements appearing close to the diagonal. PDSBTDC performs extremely well on these matrices since the ranks of the off-diagonal blocks are typically small and considerable deflation occurs during matrix multiplications stage of the algorithm. Table and Figure 5.5 present the results. Note that PDSBTDC provides an order of magnitude improvement for this test case.

# procs	PDSYEVX	PDSYEVD	PDSYEVR	PMR3	PDSBTDC
16	1027.8	802.2	892.4	787.5	53.5
32	589.5	482.8	459.6	488.8	32.4
64	256.0	301.5	259.1	DNF	22.9
128	135.3	136.8	124.4	259.6	15.3
256	83.4	89.5	91.5	262.1	10.7
512	64.3	63.7	80.3	288.2	9.1

Table 5.5 Wall Clock Time in Seconds for16K-Atom trans-PA

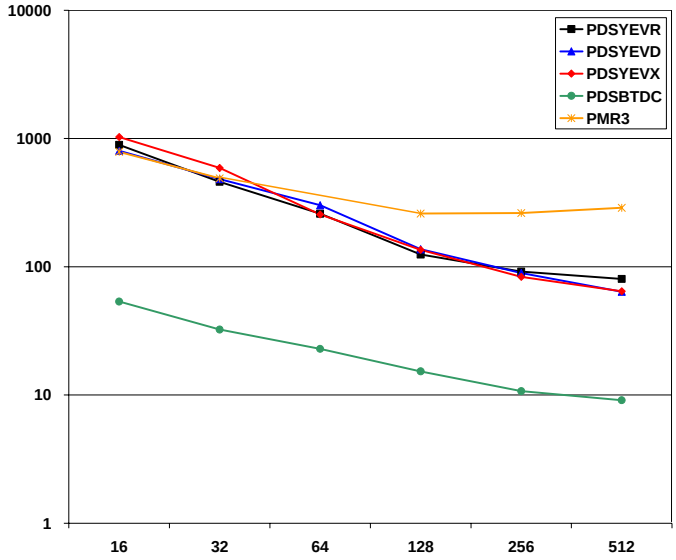


Figure 5.5 Plot of Wall Clock Time for 16K-Atom trans-PA

The results of tests on our two largest matrices, the matrices from the TCNQ family, are given below. Scalability of PDSYEVR becomes a concern on these large matrices due to the time required by the tridiagonal solver. Assigning clusters to processors becomes a more difficult task for large matrices with a large number of clusters. Clearly, PDSBTDC was not competitive – again due to the large ranks of the off-diagonal blocks.

# procs	PDSYEVX	PDSYEVD	PDSYEVR	PMR3	PDSBTDC
64	1738.7	1746.48	1813.5	1528.9	4994.8
128	888.5	797.3	856.0	1119.6	2613.6
256	417.4	447.8	549.0	882.8	1452.8
512	276.4	242.1	374.4	875.7	774.3

Table 5.6 Wall Clock Time in Seconds for the 120-State Truncated Hamiltonian from TCNQ

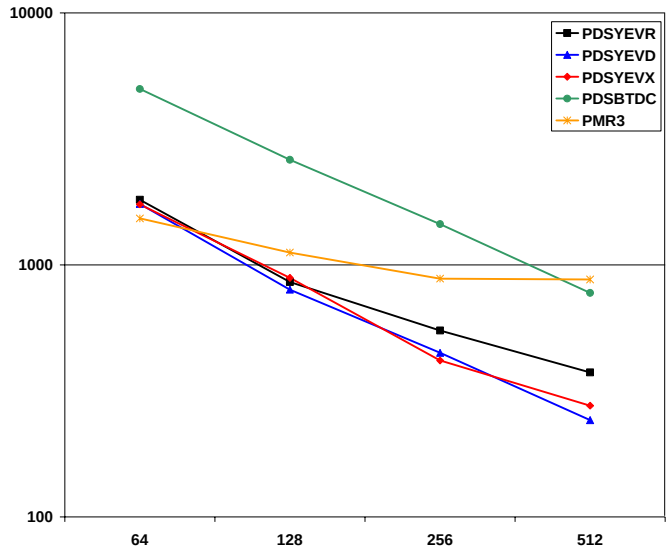


Figure 5.6 Plot of Wall Clock Time in Seconds for the 120-State Truncated Density Matrix from TCNQ

Since PDSBTDC performed poorly for the trancated TCNQ matrix given above, we did not test that algorithm on the larger TCNQ matrix. We expect similarly poor or worse performance due to the larger matrix size.

# procs	PDSYEVX	PDSYEVD	PDSYEVR	PMR3
256	5456.3	4564.7	5214.2	5689.3
512	2699.5	2294.0	2964.5	4564.8

Table 5.7 Wall Clock Time in Seconds for full Hamiltonian from TCNQ

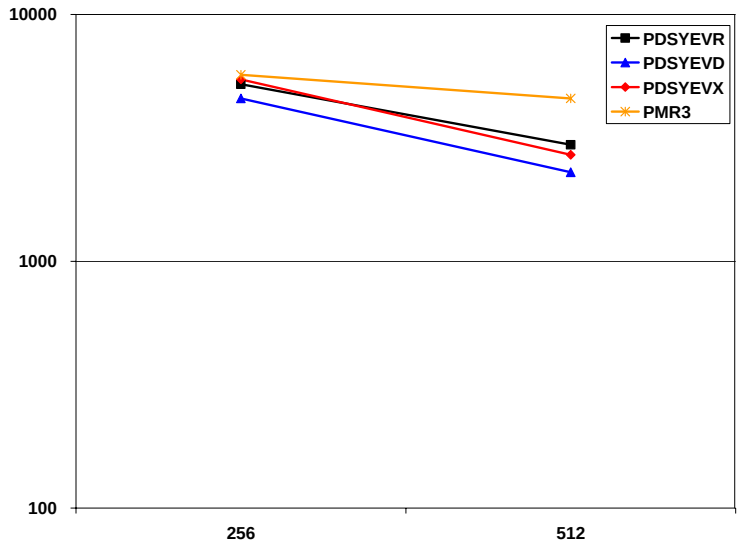


Figure 5.7 Plot of Wall Clock Time in Seconds for full Hamiltonian from TCNQ

5.2 Timing for the Classical 3-Steps in Eigensolvers. As mentioned earlier, many eigensolvers, including PDSYEVR, PDSYEVD, PDSYEVX and PMR3, involve the three major steps: (1) reduction to symmetric tridiagonal form, (2) eigen-decomposition of the tridiagonal matrix, and (3) back-transformation of the eigenvectors. During our tests, the time required for each step was recorded for PDSYEVR, PDSYEVD and PMR3. We were unable to include PDSYEVX in this test since the source code was not available for the insertion of proper timing statements. These timing results are given in Tables 5.8-5.14.

	PDSYEVD			PDSYEVR			PMR3		
# procs	**Reduc**	**Tridiag**	**Back**	**Reduc**	**Tridiag**	**Back**	**Reduc**	**Tridiag**	**Back**
4	9.9	3.3	5.9	9.9	5.5	6.2	26.0	3.7	7.7
8	5.7	2.3	3.2	5.7	3.8	3.2	20.5	1.8	2.9
16	4.4	1.4	2.1	4.0	2.6	1.9	DNF	DNF	DNF
32	3.2	1.1	1.2	3.2	2.1	1.2	DNF	DNF	DNF
64	3.0	0.8	0.7	2.9	1.8	0.7	DNF	DNF	DNF
128	3.2	0.7	0.4	3.0	1.6	0.4	10.9	0.2	0.8
256	3.5	0.6	0.3	3.6	1.7	0.3	13.2	0.1	1.1
512	4.5	0.6	0.3	4.6	1.8	0.2	15.6	0.1	1.5

Table 5.8 Wall Clock Time in Seconds for the 3 Steps Solving $C_{502}H_{1006}$

# procs	PDSYEVD			PDSYEVR			PMR3		
	Reduc	**Tridiag**	**Back**	**Reduc**	**Tridiag**	**Back**	**Reduc**	**Tridiag**	**Back**
4	37.5	11.7	26.4	42.2	17.2	28.9	50.7	10.3	24.2
8	21.7	9.0	16.4	22.3	12.2	13.8	34.0	5.1	12.2
16	17.6	4.8	8.7	19.1	4.3	9.3	DNF	DNF	DNF
32	10.0	3.0	4.6	10.1	5.0	4.6	DNF	DNF	DNF
64	8.7	2.1	2.7	8.6	4.2	2.8	DNF	DNF	DNF
128	6.9	1.6	1.4	6.9	3.8	1.4	26.7	0.4	2.2
256	7.6	1.3	1.0	7.4	3.6	1.0	29.9	0.2	2.4
512	8.4	1.3	0.6	8.3	3.6	0.7	35.4	0.1	2.7

Table 5.9 Wall Clock Time in Seconds for the 3 Steps Solving Polyalanine Chain

# procs	PDSYEVD			PDSYEVR			PMR3		
	Reduc	**Tridiag**	**Back**	**Reduc**	**Tridiag**	**Back**	**Reduc**	**Tridiag**	**Back**
4	203.3	66.8	138.5	166.9	66.0	130.3	196.6	21.3	133.7
8	101.4	33.8	73.4	109.5	39.3	74.0	127.7	11.6	64.8
16	68.3	20.7	35.9	64.3	25.2	37.7	DNF	DNF	DNF
32	36.8	11.5	20.2	37.0	14.8	18.9	66.1	3.1	18.0
64	27.7	7.2	10.0	27.9	11.0	10.0	DNF	DNF	DNF
128	17.5	5.1	5.2	19.2	9.4	5.3	66.0	0.9	6.6
256	15.9	3.7	3.2	17.4	8.5	3.3	74.2	0.6	5.8
512	16.1	3.2	1.9	17.4	8.1	1.9	81.4	0.3	6.2

Table 5.10 Wall Clock Time in Seconds for the 3 Steps Solving 544 Silicon Crystal

# procs	PDSYEVD			PDSYEVR			PMR3		
	Reduc	**Tridiag**	**Back**	**Reduc**	**Tridiag**	**Back**	**Reduc**	**Tridiag**	**Back**
8	574.3	129.6	265.3	508.1	65.0	236.1	592.3	38.0	260.1
16	250.8	67.2	148.0	293.7	44.7	154.2	329.8	18.7	138.7
32	159.7	38.7	71.3	158.6	29.5	70.6	210.2	10.2	63.9
64	92.5	22.5	36.3	94.9	22.8	37.5	DNF	DNF	DNF
128	46.6	14.1	17.3	49.6	18.9	17.5	147.8	2.6	19.2
256	39.4	10.0	10.3	41.2	17.3	10.2	163.5	1.4	14.1
512	32.2	7.2	5.6	35.7	16.6	6.4	184.5	0.8	11.4

Table 5.11 Wall Clock Time in Seconds for the 3 Steps Solving Impure Hydrogen

# procs	PDSYEVD			PDSYEVR			PMR3		
	Reduc	Tridiag	Back	Reduc	Tridiag	Back	Reduc	Tridiag	Back
16	467.7	92.5	242.0	576.7	59.5	256.1	527.8	30.6	229.2
32	311.1	51.1	120.7	311.9	27.6	120.2	356.5	15.1	117.2
64	207.4	29.3	64.8	180.2	14.4	64.5	DNF	DNF	DNF
128	87.0	18.3	31.4	83.6	9.4	31.4	222.0	3.5	34.0
256	60.3	11.8	17.4	63.9	9.8	17.8	237.8	2.1	22.2
512	45.3	9.2	9.2	46.4	24.7	9.3	270.1	1.0	17.2

Table 5.12 Wall Clock Time in Seconds for the 3 Steps Solving 16K-Atom trans-PA

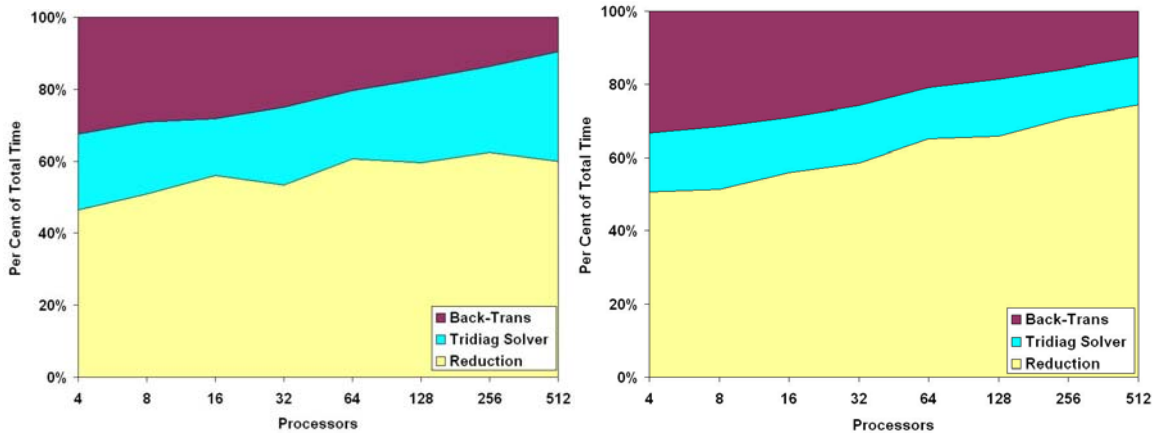
# procs	PDSYEVD			PDSYEVR			PMR3		
	Reduc	Tridiag	Back	Reduc	Tridiag	Back	Reduc	Tridiag	Back
64	1200.7	177.0	368.8	1224.4	209.4	379.8	1163.4	20.3	345.1
128	512.7	99.2	185.3	503.6	165.1	187.3	906.2	10.5	202.9
256	292.5	57.6	97.8	293.2	158.9	97.0	766.8	6.4	109.6
512	155.5	38.1	48.5	157.3	168.2	49.0	805.3	3.1	67.3

Table 5.13 Wall Clock Time in Seconds for the 3 Steps Solving Truncated TCNQ Hamiltonian

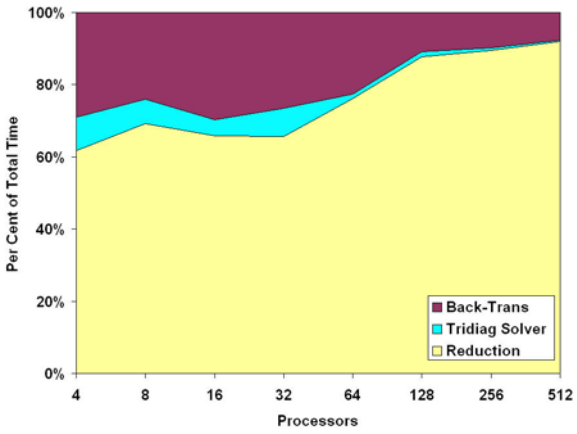
# procs	PDSYEVD			PDSYEVR			PMR3		
	Reduc	Tridiag	Back	Reduc	Tridiag	Back	Reduc	Tridiag	Back
256	3429.7	256.6	878.4	3595.5	719.3	899.4	4673.5	29.3	986.5
512	1692.0	155.7	446.2	1760.8	742.4	461.2	4005.6	15.2	543.9

Table 5.14 Wall Clock Time in Seconds for the 3 Steps Solving full TCNQ Hamiltonian

The below area charts present the average for each algorithm over all test matrices of the percentage of time spent in each step as a function of the number of processors. As is obvious, the reduction step continues to dominate eigensolvers. The below charts also illustrate the fact that PMR3's reduction step did not perform as well as PDSYEVD's or PDSYEVR's on most test cases, which is one reason PMR3 did not usually perform as well as those two algorithms. The tridiagonal solver in PMR3 was extremely efficient and was almost always the fastest tridiagonal solver of the three. Certainly part of its speed in comparison to PDSYEVR is due to the latter incorporating some recent theoretical results in an effort to compute more accurate eigenvectors.



(a) PDSYEVR *(b) PDSYEVD*



(c) PMR3

Figure 5.8 Mean Percentage of Solver Time Spent in the Three Steps as a Function of the Number of Processors

PDSBTDC has three similar steps when the matrix is not already in block tridiagonal form, although the first and third steps may employ different algorithms for different matrices. Most of the work is done in the second step, the block tridiagonal eigensolver, and we include a figure similar to the above for completeness. Figure 5.9 presents the percentage breakdown for the test cases that required all three steps, that is, the 544 Silicon Crystal, Impure Hydrogen and Truncated TCNQ Hamiltonian.

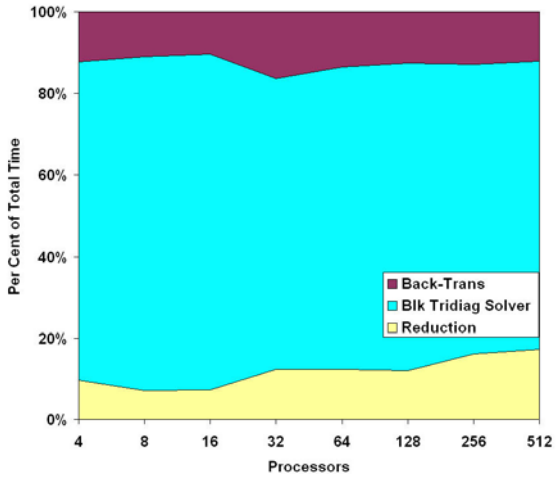


Figure 5.9 Mean Percentage of Solver Time Spent in the Three Steps of PDSBTDC as a Function of the Number of Processors

5.3 Accuracy of Computed Eigensystems. For an estimate of the accuracy of the computed eigensystem, we will use the following two scaled values:

$$\mathcal{R} = \max_{i,j=1,2,\cdots,n} \left[\left| A - VEV^T \right| \right]_{ij} \Big/ \max_i |E_{ii}|$$

$$\mathcal{O} = \max_{i,j=1,2,\cdots,n} \left[\left| V^T V - I \right| \right]_{ij} \Big/ n \ ,$$

where A is the Fock matrix or Hamiltonian matrix, V is the computed eigenvector matrix, and E is the computed diagonal matrix of eigenvalues. We use $\mathcal{R}$ and $\mathcal{O}$ as accuracy indicators rather than including the eigenvalue gaps [23] since these measures are frequently used in applications and are deemed sufficient in most cases. (Note that PDSBTDC was given a $\mathcal{R}$ accuracy request of 10^{-6} for all test matrices.)

$\mathcal{R}$ and $\mathcal{O}$ for all the test matrices, number of processors and algorithms are plotted below in Figures 5.10 and 5.11. Plots go from smallest number of processors to the largest from left to right for each matrix application.

Except for two test cases (the TCNQ matrix of order 63,504 using 256 and 512 processors), PDSYEVX returned the error warning: "Eigenvectors corresponding to one or more clusters of eigenvalues could not be reorthogonalized because of insufficient workspace." Only in one of the test cases receiving this warning (Impure Hydrogen using 512 processors) was the residual and orthogonality errors deem sufficiently unacceptable to rerun with additional workspace. The results for this test case shown in Figures 5.10 and 5.11 reflect this second run. For all the test cases except this one re-run, PDSYEVX was given the amount of workspace recommended by the code developers. It should also be noted that if sufficient workspace was given to reorthogonalize all the eigenvectors requiring this extra computation, then the execution time would have increased over the times given in Section 5.1.

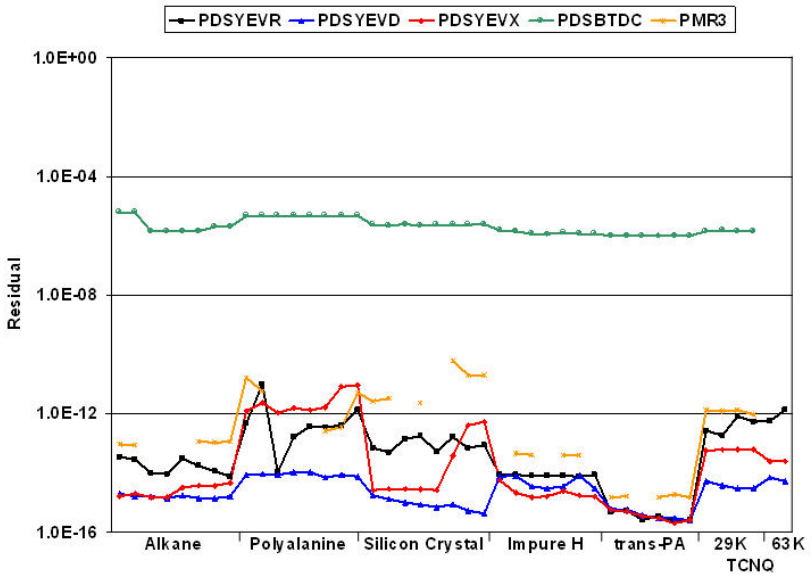


Figure 5.10 Residuals for Tested Matrices, Algorithms and Number of Processors

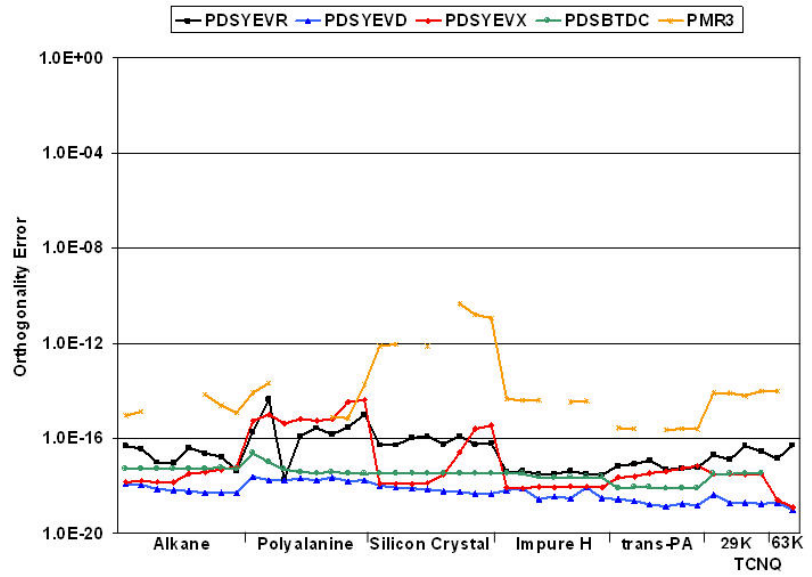


Figure 5.11 Orthogonality Errors for Tested Matrices, Algorithms and Number of Processors

6. Conclusions. Some obvious conclusions can be drawn from the test results. The three ScaLAPACK algorithms perform roughly the same with PDSYEVR, the latest code implementing the new MRRR algorithm, scaling slightly worse. Memory requirements for these three algorithms are currently the same, but it is expected that the MRRR version, PDSYEVR, will eventually be implemented requiring only $2n^2 + O(n)$ memory locations. When this occurs, PDSYEVR will have a significant advantage over the other two. PDSYEVX had the significant drawback of the amount of workspace required to

ensure re-orthogonalized eigenvectors being unknown in advance and of the possibility of unacceptable results returned due this problem.

PMR3 was competitive on the smaller number of processors, that is, when n divided by the number of processors was large. The PLAPACK algorithm reducing a matrix to tridiagonal form did not scale well. Since this step is typically the most time consuming, PMR3 did not perform well on the larger number of processors even though its tridiagonal solver was very efficient. PMR3 had the nice property of requiring only $2n^2 + O(n)$ memory locations; thus, larger problems could be solved on a smaller number of processors.

PDSBTDC was very efficient on matrices with most of the larger elements near the diagonal, which produced low ranks in some of the off-diagonal blocks. These matrices are called "heavy" diagonal matrices and occur in applications with significant locality properties. On the matrices with off-diagonal blocks near full rank, the algorithm did not perform as well as the others tested. The deflation encountered for these test cases could not offset its higher computational complexity.

The divide-and-conquer algorithms, PDSYEVD and PDSBTDC, typically computed eigenvectors with better orthogonality properties – a positive reflection upon the Gu-Eisenstat method [19] for computing eigenvectors. The residuals for PDSBTDC could not be directly compared to the other algorithms, but PDSYEVD typically produced results with smaller residuals as well. PMR3 typically produced larger residuals and orthogonality errors, although some of these results may be influenced by the same problems that prevented PMR3 from producing a computed eigensystem for some test cases.

Acknowledgments. The authors thank R. P. Muller of Sandia Laboratories for providing the alkane, polyalanine and silicon crystals families of test matrices, R. J. Hinde of the University of Tennessee for providing the hydrogen with impurities family, and Guoping Zhang of Indiana State University for providing the polyacetylene and tetra-cyanoquinodimethane families. We thank Robert van der Geijn, Peter Nagel and Paolo Bientinesi of the University of Texas for providing the PMR3 code and their assistance with the installation and porting of PLAPACK and PMR3 and Christof Vömel of University of California at Berkeley for providing the PDSYEVR code. We also thank A. S. Bland and Mark R. Fahey and Thomas H. Dunigan of the Oak Ridge National Laboratory for their invaluable assistance and detailed knowledge of the computing environment and systems at the ORNL National Center for Computational Sciences.

References.

[1] http://info.nccs.gov/resources/other_resources/cheetah.

[2] D. Antonelli and C. Vömel, *LAPACK Working Note 168: ScaLAPACK's Parallel (MRRR) Algorithm for the Symmetric Eigenvalue Problem*, Technical Report UCB//CSD-05-1399, Computer Science Division, University of California at Berkeley, Berkeley, CA 2005.

[3] *Parallel Engineering and Scientific Subroutine Library for AIX, Version 3 Release 1, and Linux on pSeries, Version 3 Release 1, Guide and Reference*, Publication Number SA22-7906-01, IBM Corporation, Poughkeepsie, NY, 2003.

[4] Y. Bai and R. C. Ward, *A Parallel Symmetric Block-Tridiagonal Divide-and-Conquer Algorithm*, Technical Report UT-CS-05-571, Department of Computer Science, University of Tennessee, Knoxville, TN, 2005.

[5] Y. Bai and R. C. Ward, *Parallel Block Tridiagonalization of Real Symmetric Matrices*, Technical Report UT-CS-06-578, Department of Computer Science, University of Tennessee, Knoxville, TN, 2006.

[6] P. Bientinesi, I. S. Dhillon and R. A. v. d. Geijn, *A Parallel Eigensolver for Dense Symmetric Matrices based on Multiple Relatively Robust Representations*, SIAM J Sci Comput., 27 (2005), pp 43-66.

[7] C. H. Bischof, B. Lang and X. Sun, *A Framework for Symmetric Band Reduction*, Technical Report ANL/MCS-P586-0496, Argonne National Laboratory, Argonne, IL, 1996.

[8] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. W. Demmel, I. Dhillon, J. J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker and R. C. Whaley, *ScaLAPACK Users' Guide*, SIAM Press, Philadelphia, PA, 1997.

[9] J. J. M. Cuppen, *A Divide and Conquer Method for the Symmetric Tridiagonal Eigenproblem*, Numer. Math., 36 (1981), pp. 177-195.

[10] M. J. S. Dewar and W. Thiel, *MNDO*, Journal of the American Chemical Society, 99 (1977), pp. 4899.

[11] I. S. Dhillon and B. N. Parlett, *Multiple representations to compute orthogonal eigenvectors of symmetric tridiagonal matrices*, Linear Algebra and Appl., 387 (2004), pp. 1-28.

[12] I. S. Dhillon, B. N. Parlett and C. Vömel, *LAPACK Working Note 162: The Design and Implementation of the MRRR Algorithm*, Technical Report UCB//CSD-04-1346, Computer Science Division, University of California at Berkeley, Berkeley, CA, 2004.

[13] I. S. Dhillon, B. N. Parlett and C. Vömel, *Glued Matrices and the MRRR Algorithm,* SIAM J Sci. Comput., 27 (2005), pp. 496-510.

[14] R. M. Dreizler and E. K. U. Gross, *Density Functional Theory*, Springer-Verlag, Berlin/Heidelberg/New York/Tokyo, 1993.

[15] W. N. Gansterer, D. F. Kvasnicka and C. W. Ueberhuber, *Multi-Sweep Algorithms for the Symmetric Eigenproblem*, in VECPAR'98 -- Third International Conference for Vector and Parallel Processing, J. M. L. M. Palma, J. J. Dongarra and V. Hernandez, eds., Springer-Verlag, Berlin/Heidelberg/New York/Tokyo,1998, pp. 20-28.

[16] W. N. Gansterer, R. C. Ward, Y. Bai and R. M. Day, *A Framework for Approximating Eigenpairs in Electronic Structure Computations*, IEEE Computing in Science & Engineering, 6 (2004), pp. 50--59.

[17] W. N. Gansterer, R. C. Ward, R. P. Muller and W. A. Goddard III, *Computing Approximate Eigenpairs of Symmetric Block Tridiagonal Matrices*, SIAM J Sci. Comput., 25 (2003), pp. 65-85.

[18] R. A. v. d. Geijn, *Using PLAPACK: Parallel Linear Algebra Package*, The MIT Press, 1997.

[19] M. Gu and S. C. Eisenstat, *A Divide-and-Conquer Algorithm for the Symmetric Tridiagonal Eigenproblem*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 172-191.
[20] R. J. Hinde, *Infrared-active vibron bands associated with substitutional impurities in solid parahydrogen*, J. Chem Phys, 119 (2003), pp. 6-9.
[21] J. Hubbard, *Electron Correlations in Narrow Energy Bands*, Proc. Roy. Soc. London, A276 (1963), pp. 238-257.
[22] R. P. Muller, J.-M. Langlois, M. N. Ringnalda, R. A. Friesner and W. A. Goddard III, *A Generalized Direct Inversion in the Iterative Subspace Approach for Generalized Valence Bond Wave Functions*, J. Chem. Phys., 100 (1994), pp. 1226.
[23] B. N. Parlett, *The Symmetric Eigenvalue Problem*, SIAM Press (reprinted version of Prentice Hall), Philadelphia, PA, 1997.
[24] B. N. Parlett and I. S. Dhillon, *Orthogonal eigenvectors and relative gaps*, SIAM J Matrix Anal. Appl., 25 (2004), pp. 858-899.
[25] B. N. Parlett and I. S. Dhillon, *Relatively robust representations of symmetric tridiagonals*, Linear Algebra and Appl., 309 (2000), pp. 121-151.
[26] R. G. Parr and W. Yang, *Density Functional Theory of Atoms and Molecules*, Oxford University Press, New York, 1989.
[27] J. P. Perdew, K. Burke and M. Ernzerhof, *Generalized Gradient Approximation Made Simple*, Phys. Rev. Lett., 77 (1996), pp. 3865-3868.
[28] J. A. Pople and D. L. Beveridge, *Approximate Molecular Orbital Theory*, McGraw-Hill, New York,1st, 1970.
[29] C. C. J. Roothaan, *New Developments in Molecular Orbital Theory*, Reviews of Modern Physics, 23 (1951), pp. 69.
[30] W. P. Su, J. R. Schrieffer and A. J. Heeger, *Soliton Excitations in Polyacetylene*, Phys Rev B, 22 (1980), pp. 2099-2111.
[31] A. Szabo and N. S. Ostlund, *Modern Quantum Chemistry*, McGraw-Hill, Berkeley, CA, 1989.
[32] F. Tisseur and J. Dongarra, *A Parallel Divide and Conquer Algorithm for the Symmetric Eigenvalue Problem on Distributed Memory Architectures*, SIAM J. Sci. Comput., 20 (1999), pp. 2223-2236.
[33] Robert C. Ward, Yihua Bai, and Justin Pratt, *Performance of Parallel Eigensolvers on Electronic Structure Calculations,* Technical Report UT-CS-05-560, Department of Computer Science, University of Tennessee, Knoxville, TN 2005.
[34] Steven R. White, *Density Matrix Formulation for Quantum Renormalization Groups*, Phys. Rev. Lett., 69 (1992), pp. 2863-2866.