Parallel solution of eigenvalue problems in electronic structure calculations *

A. Stathopoulos[†], Y. Saad[†], J. Chelikowsky[‡]

December 19, 1996

Abstract

Predicting the electronic structure of complex systems is an outstanding problem of condensed matter physics, and a source of computationally challenging eigenvalue problems. These eigenvalue problems are very large and sparse, and hundreds, or even thousands of eigenvalues are required. Their solution depends increasingly on complex data structures that reduce memory and time requirements, and on parallel computing. We present a parallel implementation of the electronic structure code on the Cray T3D that uses a combination of PVM and MPI communication libraries to achieve optimal performance. The good resulting scalability allows for new breakthrough calculations.

1 Introduction

One of the fundamental problems encountered today in chemistry and physics is to understand the dynamics of microscopic particles, in order to predict certain material properties from *ab-initio* principles. Frequently, the dynamics of atoms/electrons are followed until a 'steady state' corresponding to minimum total energy is reached. The central computation in this minimization is the repeated solution of a large, symmetric eigenvalue problem. An approach based on high-order finite difference schemes has been advocated recently. For localized systems, it has proved as accurate and more efficient than traditional plane-wave techniques[3, 4, 8, 7]. The resulting matrices are large and the number of eigenvalues and eigenvectors required is proportional to the number of atoms in the system. Complex systems involving hundreds or thousands of atoms call for more accuracy and therefore larger matrices and more eigenvalues.

The size of the eigenvalue problem, its repeated solution for each molecular dynamics step, and the large number of required eigenpairs, pose huge computational demands on high performance computers and eigenvalue solvers. To meet some of these demands, we solve the problem only in non-trivial subregions of the domain, and we use a preconditioned eigenvalue solver which can tackle hundreds of eigenpairs [11].

Parallel processing plays also a crucial role in meeting the challenges posed by electronic structure calculations. After briefly introducing the computational problem and the methods employed to solve it, we present a parallel implementation of the application on the Cray T3D. A combination of PVM and MPI communication libraries is necessary for

*Work supported by NSF grants DMR–9217287 and ASC 95-04038, and by the Minnesota Supercomputer Institute

[†]Department of Computer Science, University of Minnesota

[‡]Department of Chemical Engineering, University of Minnesota

an efficient implementation of the master-slave paradigm. Our benchmarking experiments show good scalability, despite the unstructured nature of the problem.

2 The Problem

The electronic structure of a condensed matter system, e.g., cluster, liquid or solid is described by a wave function Ψ which can be obtained by solving the Schrödinger equation: $\mathcal{H}\Psi = \mathcal{E}\Psi$, where $\mathcal{H}$ is the Hamiltonian operator for the system and $\mathcal{E}$ the total energy. Although, in its original form the operator $\mathcal{H}$ is very complex, it can be simplified through the Born-Oppenheimer and the one-electron approximations, yielding:

$$\left[\frac{-\hbar^2\nabla^2}{2m} + V_{tot}[\rho(r), r]\right]\psi(r) = E\psi(r), \tag{1}$$

where $\hbar$ is Planck's constant, m is the electron mass, and V_{tot} is the total potential at some point r in the system. The potential depends on a specific charge density $\rho(r)$, which in turn depends on the wavefunctions of the above equation via, $\rho(r) = \sum_{\text{occupied states}} |\psi_i(r)|^2$. The above problem can be viewed as a nonlinear eigenvalue problem because of the nonlinear dependence of the operator on the left-hand side on the eigenfunctions.

With the *local density approximation* theory [9], and the use of Pseudo-potentials, the potential V_{tot} is simplified to a summation of three distinct terms, specifically, $V_{tot} = V_{ion} + V_H + V_{xc}$, where V_{ion} is the ionic potential, V_H is the Hartree potential, and V_{xc} is the exchange-correlation potential. Once the charge density $\rho(r)$ is known, the Hartree potential is obtained by solving the Poisson equation:

$$\nabla^2 V_H = -4\pi\rho(r). \tag{2}$$

This can be solved by the Conjugate Gradient method, or by a fast Poisson solver. Both potentials V_H and V_{xc} have a local character and are represented by diagonal matrices in the discrete form of the problem. The ionic potential is more complex, consisting of both a local and a non-local term. It can be shown [4] that in the discrete form, the non-local term becomes a sum over all atoms a and quantum numbers l, m of rank-one updates:

$$V_{ion} = \sum_a V_{loc} + \sum_{a,\ l,m} ct_{a,l,m} U_{a,l,m} U_{a,l,m}^T, \tag{3}$$

where $U_{a,l,m}$ are sparse vectors, and $ct_{a,l,m}$ normalization coefficients.

Because the total potential and the wavefunctions are interdependent through the charge density, the above constitute a set of non-linear equations, which are solved by the Self Consistent Field (SCF) iteration. This can be viewed as an inexact Newton method.

Algorithm 2.1. **Self Consistent Iteration**

0. *Obtain initial $\rho(r)$ by superposing atomic charge densities.*
   Obtain V_{xc}, V_{ion} and initial V_H.
1. *Solve for eigenvalues/eigenvectors of the Hamiltonian.*
2. *Set $\rho(r) = \sum \psi_i^2(r)$.*
3. *Compute new V_H from (2), and V_{tot}.*
4. *'Mix' potentials with a Broyden-type quasi-Newton approach.*
5. *If self-consistent stop, else repeat from 1.*

The Hamiltonian matrix is the sum of a Laplacian matrix, three diagonal matrices (local potentials), and a matrix of summations of rank-one updates (nonlocal contributions). Typically, a small number of steps is required for the SCF iteration to converge.

3 Solving the Eigenvalue Problem

Problem Setup

In the SCF iteration the solution of the eigenvalue problem is the dominant computation, comprising more than 95% of the total time. To achieve the accuracy of the traditional plane-wave methods, our approach uses a twelfth-order finite difference scheme [1, 6, 2]. Moreover, the new approach overcomes many of the problems in plane-wave methods.

The basic grid is a uniform three dimensional cube. However, many points in the cube are often far from any atoms in the system and have a negligible charge. Their charge can be replaced by zero and computations with the associated points can be avoided. Special data structures may be used to optionally select points only from the nonzero charge regions. The size of the Hamiltonian is usually decreased by a factor of two to three. This is important considering the large number of eigenvectors that need to be saved. Further, since the Laplacian can be represented by a simple stencil, and since all local potentials sum up to a simple diagonal, the Hamiltonian need not be stored. The non-local potential is computed implicitly as the sum of several rank one matrices for each atom.

The Davidson algorithm

There are several difficulties with the eigenproblems produced in this application, besides their size and stencil representation of the matrix. First, the number of required eigenvectors is proportional to the atoms in the system, and can grow up to thousands. Maintaining the orthogonality of these vectors is an additional problem. Second, the separation of the eigenvalues, and thus the rate of convergence of iterative solvers becomes increasingly poor as the matrix size increases. Preconditioning techniques attempt to alleviate this problem. Finally, at each iteration, very good initial eigenvector estimates are available from the previous SCF iteration. An iterative method should be able to use this information.

The Davidson method is a popular preconditioning variant of the Lanczos iteration [5]. In this application, a code based on the generalized Davidson [10] has been developed. This code addresses the problems mentioned above by using implicit deflation (locking) and special targeting and reorthogonalization schemes. For more information see [11].

Preconditioning

One preconditioner used in our approach is based on a filtering idea and the fact that the Laplacian is an elliptic operator [13]. The eigenvectors corresponding to the few lowest eigenvalues of ∇^2 are smooth functions and so are the corresponding wavefunctions. When an approximate eigenvector is known at the points of the grid, a smoother eigenvector can be obtained by averaging the value at every point with the values of its neighboring points. Assuming a (x, y, z) coordinate system, the low frequency filter acting on the value at the point (i, j, k), which corresponds to one element of the eigenvector, is described by:

$$V_{i,j,k} := \frac{V_{i-1,j,k} + V_{i,j-1,k} + V_{i,j,k-1} + V_{i+1,j,k} + V_{i,j+1,k} + V_{i,j,k+1}}{12} + \frac{V_{i,j,k}}{2} \tag{4}$$

This preconditioning has provided significant improvements on the iteration count.

4 Parallel Implementations

Parallel paradigm

The master-slave paradigm is followed for the SCF procedure. The master performs most of the preprocessing, computing of scalar values, and receiving the new potential at each SCF iteration. The master is also responsible for applying the mixing scheme on the potentials.

The slaves solve for the eigenvalues and eigenvectors, update the charge density, and solve the Poisson equation for the Hartree potential.

There are several reasons dictating this choice. First, there are some inherently sequential parts in the code which require large memory but very short execution time. It is also common that one of the machines in a parallel environment is equipped with larger memory than the other nodes. Second, the code calls several library routines which have been written by various research groups over a long period of time. Despite their importance, these routines take only a few seconds to execute. Parallelizing them would require an immense effort in understanding the underlying physics principles, and the gains, if any, would be insignificant. Third, this paradigm allows for incremental parallelization of the code by implementing first the most time consuming procedures. In this way, the correctness of the code is assured. Finally, even for small cases most of the time is spent in solving the eigenvalue problem.

Problem Mapping

The problem is mapped onto the processors in a data parallel way because of the fine granularity parallelism present in the matrix-vector multiplication and orthogonalization operations. The rows of the Hamiltonian (and therefore the rows of the eigenvectors and potential vectors) are assigned on different processors, according to a partitioning of the physical domain. The subdomains can be chosen naturally as sub-cubes or slabs of the domain. The program only requires a function $P(i,j,k)$ that returns the processor number where point (i,j,k) resides. This facilitates the use of any available partitioning tools. Since the matrix is not actually stored, an explicit reordering can be considered so that the nodes of a processor are numbered consecutively. Under this conceptually easier scheme, only a list of pointers is needed that denote where the rows of each processor start. The non-local part of the matrix, which is a sum of rank-one updates, is mapped in a similar way. For each atom, and for each pair of quantum numbers, the vector of the rank-one update is partitioned according to the rows it contributes to. With this mapping, the storage requirements of the program are distributed and large problems can be solved.

Parallel Davidson

In the Davidson algorithm, the basis vectors and long work arrays, follow the same distribution as the eigenvectors. Thus, all vector updates (saxpy operations) can be performed in parallel, and all reduction operations (e.g., sdot operations) require a global reduction (e.g., global sum) of the partial results on each processor.

The matrix-vector multiply is performed in three steps. First, the contributions of the diagonals (potentials and the Laplacian diagonal) is computed in parallel on all processors. Second, the contribution of the Laplacian is considered on the rows of each processor. As in the sequential code, this is performed by using the stencil information. However, some of the neighbors of the local subdomain may reside on different processors, and communication is necessary. For this reason, a special preprocessing phase has been implemented. During the setup of the non-local information by the master, the slaves locate which of their rows are needed in the stencils of other processors. Data structures are built for all the information that needs to be communicated. In the second step of matrix-vector multiplication, this information is exchanged among the processors and the stencil multiplication can proceed in parallel. Since, only nearest neighbor communication is required, this technique is considerably more efficient than globally collecting the whole vector on all processors. In the third step, each of the rank-one updates of the non-local components is computed as a distributed dot product. All local dot products are first computed before a global summation of their values takes place. Thus, matrix-vector multiplication requires two communication-synchronization points. The solution of (2) for the Hartree potential with Conjugate Gradient and the preconditioning operation, also require the stencil and therefore they have the same communication pattern as the second step of the matrix-vector multiplication.

Orthogonalization is a very expensive phase and as the number of required eigenvectors increases, it is bound to be the dominant one. Although reorthogonalization recovers the numerical accuracy of the Gram-Schmidt procedure, its nature is sequential and induces several synchronization points. In the current application, global summations of the dot-products are delayed so that only one synchronization is needed. Also, easily obtained estimations of the vector norms are used to perform the reorthogonalization test. As a result the procedure has only two synchronization points and it is quite efficient.

The T3D and Communication libraries

We have developed two implementations of this parallel code; one using the PVM message passing library and one using the MPI standard communication interface. The codes have been tested on a variety of machines. On the SGI Power Challenge cluster the relatively slow communication between cluster nodes limits the scalability of the code. The Power Switch of the IBM SP2 improves communication performance and thus scalability, but only up to a 70% efficiency on eight processors[12]. The nodes on the T3D are several times slower and with smaller memory than the SP2 ones, but its network provides an order of magnitude improvement in performance over the Power Switch. Clearly, the T3D accommodates better the needs of data parallel applications. Moreover, the aggregate distributed memory from 512 processors on the T3D facilitates the solution of very large problems.

On the above machines, the native MPI implementation is more efficient than the corresponding PVM one. Although, in some cases performance might be comparable, MPI has more efficient implementations of the collective operations. On the T3D, the MPI EPCC implementation is twice as fast as the native PVM one. Thus, the need for efficient communication in our application prompts to the use of MPI.

The small memories of the T3D nodes necessitate the use of the front-end, Cray Y/MP as the master. Unfortunately, the MPI EPCC implementation does not allow message passing between the T3D and other machines. In addition, the latest MPICH implementation, although faster for point-to-point communications, it is still quite slower for collective operations. To meet these contradictory requirements of the application, we used PVM to communicate with the front-end and MPI EPCC for the communication among the slaves. Thus, we allow for flexibility of the code without compromising the efficiency of its data parallel part.

Several issues had to be addressed in this combination of libraries. Since MPI does not support dynamic process management, the slaves are spawned by the master through PVM. After the slave processes are initiated, they start MPI among themselves and create the necessary communication groups. All communication with the master is performed with PVM and the slaves use the MPI reduction and all-to-all primitives to solve the eigenvalue problem. Another problem is the fact that PVM cannot open more than 64 channels to the T3D simultaneously. Therefore, for large number of processors, broadcasts should be sent to only one slave, which afterwards broadcasts the data with MPI. We also faced memory allocation problems because of PVM buffering and the small memory of the T3D

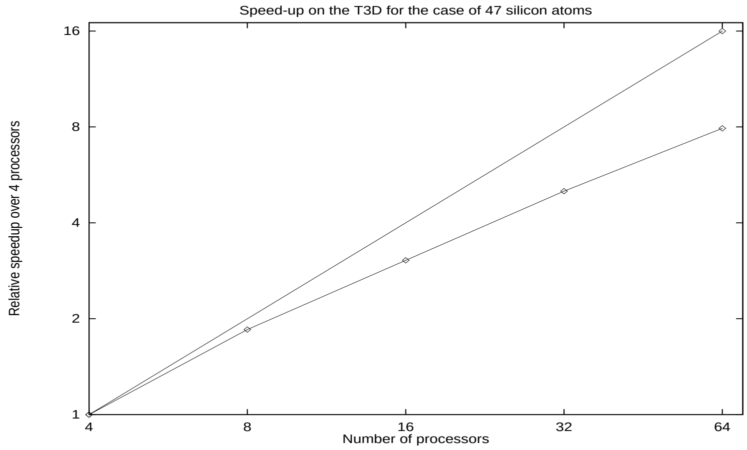

FIG. 1.

nodes. Large messages had to be split into manageable sizes and the receiving slave had to acknowledge reception to prevent buffer flooding by subsequent messages.

5 Results

To demonstrate the scalability of the code we ran several test cases. The first is a small one, involving 47 Silicon atoms, with matrix size 29281, and 165 eigenpairs are required. The second case is a larger one, involving 191 Silicon atoms, with matrix size of 83200, and 560 eigenvalues are required. The third is a case with 93 Ga and As atoms, with a larger matrix size of 145000, but only a moderate number of 163 required eigenvalues. A minimum of four processors were necessary to fit the first case on the T3D, and eight processors for the rest two cases. Speedups appear in Figures 1, 2, and 3 correspondingly. The figures show that the speedups are very good even for the small case, and that they improve with increasing problem sizes. In Figure 3, the small degradation in performance from 64 to 128 processors is due to nearest domain communication.

Besides the above scalability to the problem size, the code demonstrates another kind of scalability. As more eigenvalues converge, the orthogonalization becomes the dominant part of the computation and therefore the parallel efficiency of the code improves. The reason is that the two global reductions required by orthogonalization are much cheaper than the communication step in matrix-vector multiplication. Figure 4 depicts this behavior. As a result, difficult problems requiring large number of eigenvalues should exhibit very good efficiencies.

6 Conclusion

There are several challenges arising in the electronic structure problem. Besides improving algorithms for computing large numbers of eigenvectors, parallel computing can help meet these challenges. The codes developed for the T3D show good scalability despite the unstructured nature of the problem. This parallel code has been used sucessfully to solve systems with more than 500 atoms.

References

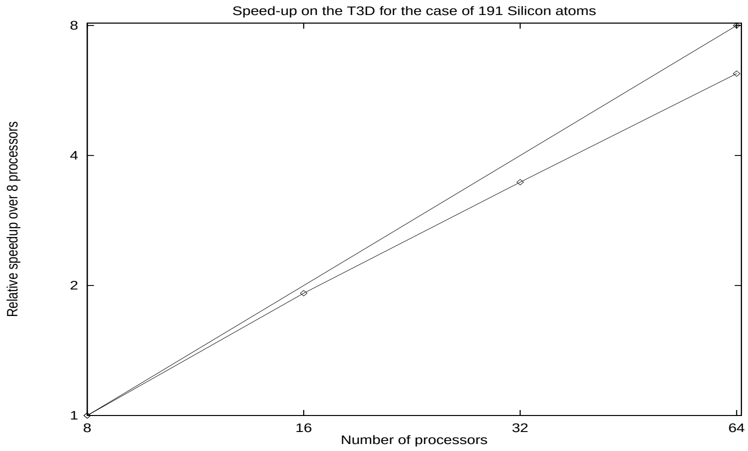

Fig. 2.

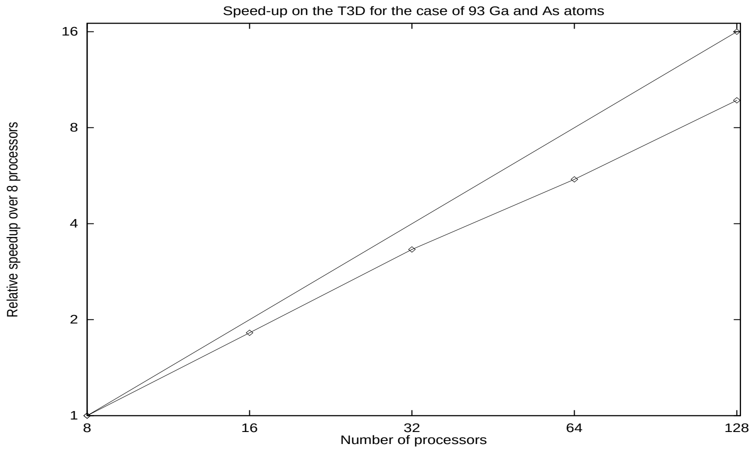

Fig. 3.

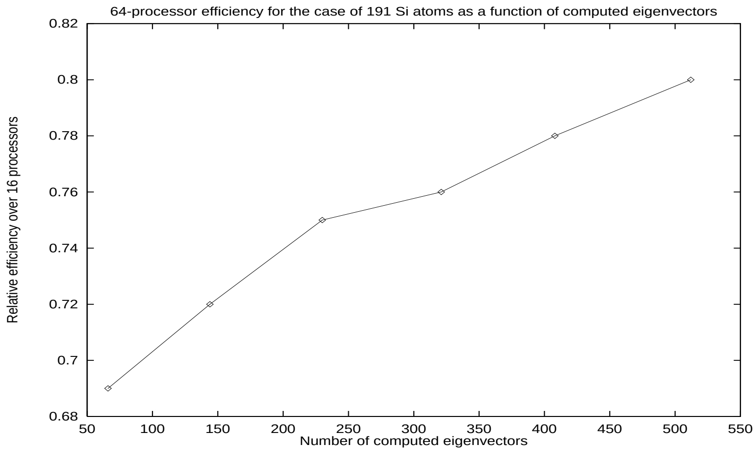

Fig. 4.

[1] J. R. Chelikowsky, N. Troullier, and Y. Saad. Finite-difference-pseudopotential method: electronic structure calculations without a basis, *Phys. Rev. Lett.*, 72,1240–3 (1994).

[2] D. T. Colbert and W. H. Miller. A novel discrete variable representation for quantum mechanical reactive scattering via the S-matrix Kohn method, *J. Comput. Phys.*, 96,1982–91 (1992).

[3] J.R. Chelikowsky, N. Troullier, K. Wu, and Y. Saad, Higher Order Finite Difference Pseudopotential Method: An Application to Diatomic Molecules, *Phys. Rev. B* 50,11355-64 (1994).

[4] J. R. Chelikowsky, N. R. Troullier, X. Jing, D. Dean, N. Binggeli, K. Wu, and Y. Saad. Algorithms for the structural properties of clusters. *Computer Physics Communications*, 85,325–335 (1995).

[5] E.R. Davidson, The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices, *J. Comput. Phys.*, 17, 87 (1975).

[6] B. Fornberg and D. M. Sloan. A review of pseudospectral methods for solving partial differential equations. *Acta Numerica*, 203–267 (1994).

[7] X. Jing, N. R. Troullier, J. R. Chelikowsky, K. Wu, and Y. Saad. Vibrational modes of silicon nanostructures. *Solid State Communication*, 96(4),231 (1995).

[8] X. Jing, N. R. Troullier, D. Dean, N. Binggeli, J. R. Chelikowsky, K. Wu, and Y. Saad. *Ab initio* molecular dynamics simulations of Si clusters using the higher-order finite-difference-pseudopotential method. *Phys. Rev. B*, 50,12234-7 (1994).

[9] W. Kohn and L. J. Sham, Phys. Rev., 140, A,1133 (1965).

[10] R.B. Morgan and D.S. Scott, Generalizations of Davidson's Method for Computing Eigenvalues of Sparse Symmetric Matrices, *SIAM J. Sci. Stat. Comput.* 7, 817 (1986).

[11] Y. Saad, A. Stathopoulos, J. R. Chelikowsky, K. Wu, and S. Öğüt. *BIT*, 36, 3, 563–578 (1996).

[12] A. Stathopoulos, Y. Saad, and J. R. Chelikowsky, Porting electronic structure calculations to the IBM-SP2, in Proceed. *International Conference on Parallel Computing,* 1996, Minneapolis, MN.

[13] C. H. Tong, T. F. Chan, and C. C. J. Kuo. Multilevel filtering preconditioners: extensions to more general elliptic problems. *SIAM J. Sci. Stat. Comput.*, 13,227–242 (1992).