Parallel methods and tools for predicting material properties

Version# 2

September 10, 1998

Parallel methods and tools for predicting material properties *

Andreas Stathopoulos † Serdar Öğüt ‡ Yousef Saad § James Chelikowsky ‡
Hanchul Kim ‡

Abstract

Predicting the electronic structure of complex systems is an outstanding problem in materials science. If the electronic structure of a given material is known, then many physical and chemical properties can be accurately determined without resorting to experiment. At present, the optical and structural properties of *simple* solid phases can be predicted solely from a knowledge of the atomic species present. Of course, while it is intriguing to speculate about designing complex materials on the computer, many technical and fundamental questions remain to be addressed. The chief obstacle faced by computational materials and computer scientists in computing the electronic structure of matter is the solution of a complex eigenvalue problem. Often, these eigenvalue problems are large and sparse, and 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 an electronic structure code implemented on the Cray T3D and T3E. This code uses a combination of PVM and MPI communication libraries to achieve optimal performance. The good resulting scalability has enabled us to perform some "breakthrough" calculations, *e.g.*, predicting the optical properties of $\sim$ 1,000 atom systems from first principles.

1 Introduction

A fundamental problem in materials science is the prediction of the electronic structure of condensed matter. Since all matter is composed of nuclei and electrons, once the interaction energies of these constituents are known, it is possible to sum them up to evaluate the total electronic energy. Changes in this total energy from one atomic configuration to another can be used to determine a variety of important properties. For example, suppose we consider two hypothetical crystal structures for a specified chemical composition. By evaluating the total electronic energy of the two structures, we can predict which one is more stable and likely to exist in nature. We can change the volume of the crystal and predict which phase will be stable as function of pressure. Or, we could calculate the changes of the total energy of the crystal in an applied electric field and predict the dielectric properties of the crystal. These are only simple examples. In principle, *any* physical or chemical property can be determined by knowing the electronic structure of matter as function of the structure, applied fields and chemical makeup. It is clear that a number of important scientific questions can be addressed by knowledge of the electronic structure properties, *e.g.*,

*Work supported by NSF grants DMR 95-25885 and ASC 95-04038, and the Minnesota Supercomputing Institute
†Department of Computer Science, College of William and Mary
‡Department of Chemical Engineering and Materials Science, University of Minnesota
§Department of Computer Science, University of Minnesota

predicting potential superconducting solids or superhard solids [1]. Also, a number of technologically relevant problems such as the design of lasers, optical detectors, and sensors, could be aided by an understanding of electronic properties.

While these problems can be addressed in principle, a determination of the electronic structure of matter is one of the major challenges of materials science. The complexity of the problem arises because of the number of particles involved, *e.g.*, a macroscopic amount of matter may contain 10^{23} electrons and nuclei, and the small energy differences involved between competing phases of matter, *e.g.*, the energy differences between two phases may be less than 10^{-6} times the total energy of the phase. Moreover, the problem involves the solution of the many-body Schrödinger equation:

$$\mathcal{H}(R_1, R_2, ...; r_1, r_2, ...)\Psi(R_1, R_2, ...; r_1, r_2, ...) = \mathcal{E}\Psi(R_1, R_2, ...; r_1, r_2, ...)$$

where the $\{R_i\}$'s give the nuclear coordinates and the $\{r_n\}$'s correspond to the electronic coordinates. The eigenvalue $\mathcal{E}$ corresponds to the total energy of the system. $\mathcal{H}$ is the Hamiltonian operator given by

$$\mathcal{H}(R_1, R_2, ...; r_1, r_2, ...) = -\sum_i \frac{\hbar^2\nabla_i^2}{2M_i} - \sum_n \frac{\hbar^2\nabla_n^2}{2m} + \frac{1}{2}\sum_{ij} \frac{Z_j Z_i e^2}{|R_i - R_j|} - \sum_{in} \frac{Z_i e^2}{|R_i - r_n|} + \frac{1}{2}\sum_{nm} \frac{e^2}{|r_n - r_m|} + ...$$

where Z_i is the atomic number of the ith species, e is the electronic charge of the electron, $\hbar$ is Planck's constant, m is the mass of the electron and M_i is the atomic mass of the ith species. The sums are over all particles and combinations thereof. The first two terms correspond to the kinetic energy of the nuclei and electrons, respectively. The next three terms correspond to nuclear-nuclear, nuclear-electron and electron-electron interactions. This Hamiltonian neglects some spin-related terms, but in general these are less important for the materials of interest to us, *e.g.*, semiconductors like silicon.

At first glance, it would appear that this is an intractable problem. Historically, the advent of electronic computing brought some optimism that at least small problems, *e.g.*, atomic systems with only a few electrons, might be handled. However, this optimism was not justified, and to date, few numerical solutions of the full many-body Schrödinger equation exist save for small atomic or molecular systems. Only recently have accurate solutions for the electronic energies been available and these solutions involve making some significant approximations (see sidebar 1).

Under these approximations, evaluating the total energy of the system involves the computation of the electronic energies of all occupied states. This can be a complicated task. Consider a relatively small cluster of matter, *e.g.*, a cluster of silicon atoms 20 Å in diameter. The cluster may contain several thousands of electrons; one must compute several thousand eigenvalues for such a system.

Traditional approaches for solving the eigenvalue problem utilize a plane-wave basis to expand the wavefunctions (eigenvectors) [5]. Using the fast Fourier transforms (FFT) to iteratively solve the resulting algebraic eigenvalue problem, plane waves have often proved an effective approach (see sidebar 2). However, our group has recently been advocating the use of *real-space* methods in which the problem is discretized in real space with the help of high-order finite difference schemes. For localized systems, the real space method has proved to be as accurate and often more efficient than techniques based on plane waves [5]. In contrast to plane wave methods, the resulting matrices are sparse, *i.e.*, they only have a small number of nonzero elements per row, but they are usually larger. With complex systems involving hundreds or thousands of atoms, the need for accuracy increases the size of the matrices even further.

The shear size of the matrix eigenvalue problem, as well as the large number of eigenpairs required, pose huge computational demands on high performance computers and eigenvalue solvers. In order to meet these demands, it is mandatory to combine two ingredients: (1) high-performance computing platforms such as a massively parallel computers and (2) efficient and robust matrix algorithms. In recent years substantial effort has been devoted to parallelizing electronic structures codes. Our main motivation for focusing on the real space approach has been the ease with which it can be implemented on parallel machines and its versatility. Though the physical region is a rectangular box, we solve the problem only in those subregions of the domain, where the electronic properties of the system are nonzero. This gives rise to unstructured sparse matrices. On the linear algebra side, a preconditioned Krylov technique is used to extract up to several hundreds of eigenpairs [14].

A question often raised with regard to the parallel implementation of eigenvalue methods is the potential inefficiency due to inner products and the orthogonalization process required by these methods. Though this issue will be discussed at length in this paper, it is worth mentioning at the outset that the large number of eigenvalues computed makes the problem actually far less critical. In fact the orthogonalization process is easy to parallelize and we attribute much of the efficiency of the algorithm to this part of the computation. This part becomes more efficient as the number of eigenvalues to be computed increases since this means that we will have larger dense computations which are easier to parallelize.

Existing software libraries for parallelizing scientific applications place several restrictions on the generality of the matrix storage format. Highly efficient codes which exploit special characteristics of our problem, its unstructured sparsity, the implicit use of the matrix, and its data parallel nature, are critical to the success of a parallel implementation.

After a brief introduction to the computational problem in Section 2 and the methods employed for its solution in Section 3, we present a parallel implementation of the application on the Cray T3D and Cray T3E in Section 4. Versions based on MPI and PVM communication libraries have been developed that can also be used on other parallel machines and clusters of workstations. However, for an efficient implementation of the master-worker paradigm on the T3D/T3E, a combination of PVM and MPI libraries is necessary. In Section 5, a few results on average and small size problems demonstrate the good scalability of the implementation. Finally, in Section 6, we show an application of this code to a computationally challenging problem on large Silicon quantum dots which has produced new theoretical insights.

2 The Computational Problem

Within the local density approximation (LDA) [2], the many-body Schrödinger equation is reduced to the one-electron equation

$$\mathcal{H}\ \psi_i(r) = [-\frac{\hbar^2\nabla^2}{2m} + V^p_{ion}(r) + V_H(r) + V_{xc}(r)]\ \psi_i(r) = \mathcal{E}_i\ \psi_i(r), \tag{1}$$

where $\mathcal{E}_i$ corresponds to an energy eigenvalue and ψ_i is the corresponding eigenstate. The pseudopotential V^p_{ion} replicates the potential of the core electrons plus the nuclear potential. The Hartree potential V_H is calculated from the valence charge density function $\rho(r)$ by solving the Poisson equation:

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

In turn, the valence charge density is calculated from

$$\rho(r) = -2e\ \sum_i |\psi_i(r)|^2 \tag{3}$$

Approximations for solving the Schrödinger equation

The Schrödinger equation in its original form is a function of all particles (nuclei and electrons) in the system and it involves interactions among all possible combinations thereof. Typically, three approximations are made to make the solution of this system tractable. The first approximation is that the motion of the nuclei which are much heavier than the electrons can be neglected. This is the Born-Oppenheimer approximation. Within this approximation, the nuclear positions are fixed and only the electronic degrees of freedom are considered. The second approximation made is to replace the many-body problem with the so-called "one electron" problem. Within this assumption, the electrons are treated independently. Each electron sees only the average potential from all the other electrons and nuclei in the system of interest. For large or extended systems, the one electron approximation is almost always implemented within a density functional theory [2]. Within this approach the spatial distribution of electrons (which in quantum mechanics is known only in a probablistic or statistical sense) is treated as an electronic charge density. If the charge density is known, then the electronic energy of the system is known. The third approximation is the pseudopotential approximation [3]. The "real potential" is replaced by a pseudopotential which considers only the outer electrons of the atom. The outer electrons, or valence electrons, determine the chemical properties of matter. An obvious reflection of this observation is the organization of the Periodic Table. Elements with similar chemical properties, *e.g.*, C, Si, Ge and Sn, all have an outer electron configuration of s^2p^2. The pseudopotential approximation allows one to focus only on the electrons of interest. It greatly reduces the size of the problem, *e.g.*, for Ge there are 4 valence electrons and 28 core electrons (inner electrons).

Sidebar 1

Plane wave and real space methods

The plane wave approach originated in solid state physics and it is akin to spectral techniques used in solving certain types of partial differential equations. Since crystalline matter is periodic, it is natural to expand the wave functions in plane waves and utilize fast Fourier transforms (FFT) to evaluate matrix elements. The traditional plane wave discretization, uses a basis of the form: $\left\{e^{i\ k \cdot r}\right\}_k$, where k is the wave number which is a three-component vector, and r is the 3-dimensional variable in space. The term $k.r$ is the scalar product between the 3-dimensional vectors k and r. In this basis, the Laplacian term of the Schrödinger operator is represented by a diagonal matrix. In contrast, the potential term gives rise to a dense matrix. In practice, these matrices are never formed explicitly since with appropriate use of FFT we can easily operate with this matrix by going back and forth between real-space and Fourier space. Indeed, in real space it is trivial to operate with the potential term which is represented by a diagonal matrix and in Fourier space it is trivial to operate with the Laplacean term which is also represented by a diagonal matrix. When iterative methods are used to solve the eigenvalue problem, the plane wave bases also gives rise to a natural preconditionning matrix obtained by employing a smaller plane wave basis, neglecting the effect of high frequency terms on the potential.

For periodic systems, these methods are quite effective. However, for non-periodic systems such as clusters, liquids or glasses, the plane-wave basis must be combined with a *supercell method* [3], which artificially repeats the localized configuration to impose periodicity to the system. As with spectral techniques, plane waves loose their superiority when more general, non periodic geometries are required. In addition to these difficulties the two FFT performed at each iteration can be costly. This is exacerbated in high performance environments where FFTs require an excessive amount of communication and are difficult to implement efficiently.

In the *real-space* approach the problem is discretized in real space with the help of finite difference or finite element schemes. This approach overcomes the above problems of non-periodicity and can be used to describe non-crystalline matter. In our methods we use a high-order finite difference scheme on a uniform grid, but other real space methods have also appeared [19, 20, 21]. Although the resulting matrices can be larger than with plane-waves, they are sparse and the methods are easier to parallelize. Even on sequential machines, our experiments have shown that the method can be an order of magnitude faster than the traditional approach.

Sidebar 2

where the sum is over all occupied eigenstates. The factor of two arises as each eigenstate can accommodate two electrons. Partially occupied states are often also allowed. The exchange-correlation potential V_{xc} is defined as a functional of the charge density, $V_{xc}(r) = V_{xc}([\rho(r)])$ [2].

A complicating issue is that the ion core term is described by a non-local pseudopotential [3]. The interactions between valence electrons and ion cores may be separated into a local pseudopotential and a Kleinman and Bylander [4] form of a non-local pseudopotential:

$$V_{ion}(r)\psi_n(r) = \sum_a V_{loc}(|r_a|)\psi_n(r) + \sum_{a,\ lm} K^a_{n,lm} u_{lm}(r_a)\Delta V_l(r_a) \tag{4}$$

$$K^a_{n,lm} = \frac{1}{< \Delta V^a_{lm} >} \int u_{lm}(r_a)\Delta V_l(r_a)\psi_n(r) d^3r, \tag{5}$$

and $< \Delta V^a_{lm} >$ is the normalization factor,

$$< \Delta V^a_{lm} >= \int u_{lm}(r_a)\Delta V_l(r_a)u_{lm}(r_a)d^3r, \tag{6}$$

where $r_a = r - R_a$, and the u_{lm} are the atomic pseudopotential wave functions of angular momentum and azimuthal quantum numbers, (lm), from which the l dependent ionic pseudopotential $V_l(r)$ are generated. $\Delta V_l(r) = V_l(r) - V_{loc}(r)$ is the difference between the l component of the ionic pseudopotential and the local ionic potential. The quantities needed to define the ionic pseudopotential, $(u_{lm},\ \Delta V_l)$, can be determined from atomic structure programs which involve a one-dimensional wave function [3]. This is a trivial exercise when compared to the three dimensional problems of a solid or liquid.

Within this formalism, the total electronic energy of the system of interest can be calculated:

$$E_{total} = \sum_i \mathcal{E}_i - \frac{1}{2}\int V_H(r)\rho(r)\ d^3r + \int [E_{xc}(\rho(r)) - V_{xc}(\rho(r))]\rho(r)\ d^3r + E_{ion-ion} \tag{7}$$

where the sum is over occupied eigenvalues, and E_{xc} is the energy density functional within the local density approximation: $V_{xc} = \frac{\delta E_{xc}}{\delta \rho}$.

To obtain the Hartree and exchange-correlation terms, we need to know the charge density. Since the charge density depends on the eigenvalues, and is initially unknown, a self-consistent field (SCF) approach is used. This is a Newton-type method that solves the set of non-linear equations (1), (2), (3), (4). To start the problem, atomic charge densities are used to construct the Hartree and exchange-correlation potentials. The atomic wave functions are only approximate and the initial potentials are usually not very accurate. However, once the eigenvalues are known from a solution of the eigenvalue problem, a new Hartree and exchange-correlation potentials are calculated. The process is repeated until the potentials and charge densities are "self-consistent."

ALGORITHM **2.1 Self Consistent Field (SCF) 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 $\rho(r) = \sum \psi_i^2(r)$, for all occupied states.*
3. *Compute new V_H from (2), and then update V_{tot}.*
4. *'Mix' potentials with a Broyden-type quasi-Newton approach.*
5. *If self-consistent stop, else repeat from 1.*

From a computational point of view, the Hartree potential is obtained by using the Conjugate Gradient method or a fast Poisson solver. For example, with a cluster geometry, the Hartree potential can be described by a multi-pole expansion outside of the domain of interest. The multipoles can be determined from the charge density using standard electrostatics. Since the Hartree potential is known outside of the cluster domain, it can be used to set the boundary condition. Both potentials V_H and V_{xc} have a local character and in the finite difference scheme of discretization, they are represented by diagonal matrices.

Handling the ionic potential is complex, since it consists of both a local and a non-local term. In the discrete form, the non-local term becomes a sum over all atoms a and quantum numbers l, m of rank-one updates:

$$V_{ion} = \sum_a V_{loc} + \sum_{a,\ l,m} ct_{a,l,m} U_{a,l,m} U_{a,l,m}^T, \tag{8}$$

where $U_{a,l,m}$ are sparse vectors which are non-zero only in few localized regions of the domain, and $ct_{a,l,m}$ are normalization coefficients.

From the above, the Hamiltonian matrix is the sum of a Laplacian matrix, three diagonal matrices (local potentials), and a matrix consisting of a sum of rank-one updates (nonlocal contributions). The number of steps required for the SCF iteration to converge is system dependent, but often only a few iterations are required.

3 Solving the Eigenvalue Problem

The most computationally intensive part of an electronic structures calculation is in the self consistency loop. Within each SCF iterations, the dominant computation lies in the solution of the eigenvalue problem. Therefore, it is important to develop discretizations and approximation methods that can take advantage of efficient eigenvalue solvers.

Problem Setup

One advantage of the finite difference approach over traditional plane-waves is that the resulting matrices, though larger, are sparse. This results in solution methods that are easier to parallelize. To achieve the accuracy of the plane-wave methods we use a twelfth-order finite difference scheme [5]. This increases the accuracy while keeping the matrices at manageable sizes. Depending on the kind of atoms involved in the simulation, lower order schemes may be sufficient.

The basic grid we use is a uniform three dimensional cube, with each grid point corresponding to a row in the matrix. However, many points in the cube are often far from any atoms in the system and their negligible charge may then be replaced by zero. Special data structures may be used to discard these points and keep only those having a nonzero charge (see figure 1). The size of the Hamiltonian matrix is usually reduced by a factor of two to three with this strategy, which is quite important considering the large number of eigenvectors which must be saved. Further, since the Laplacian can be represented by a simple stencil, and since all local potentials sum up to a simple diagonal matrix, 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 method

There are several difficulties with the eigenproblems generated in this application in addition to the size of the matrices. First, the number of required eigenvectors is proportional to the atoms in the system, and can grow up to thousands. Besides storage, maintaining the orthogonality of these vectors can be a formidable task. Second, the relative separation of the eigenvalues becomes increasingly poor as the matrix size increases and this has an adverse effect on the rate of convergence of the eigenvalue solvers. Preconditioning techniques attempt to alleviate this problem.

On the positive side, the matrix need not be stored as was mentioned earlier and this reduces storage requirement. In addition, good initial eigenvector estimates are available at each iteration from the previous SCF loop. An iterative method should exploit this information.

The Davidson method is a popular extension of the Lanczos iteration that uses preconditioning. In this work, we developed a code based on the generalized Davidson [11] method, in which the preconditioner is not restricted to a diagonal matrix as in the Davidson method. The code addresses the problems mentioned above by using implicit deflation (locking), a windowing approach to gradually compute all the required eigenpairs, and special targeting and reorthogonalization schemes. A more detailed description can be found in [14].

Preconditioning

A preconditioning technique we used in our approach is based on a filtering idea and the fact that the Laplacian is an elliptic operator [15]. 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 an (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:

$$V_{i,j,k} := \frac{V_{i-1,j,k} + V_{i,j-1,k} + V_{i,j,k-1} + V_{i+1,j,k} + V_{i,j+1,k} + V_{i,j,k+1}}{12} + \frac{V_{i,j,k}}{2} \tag{9}$$

It is worth mentioning that other preconditioners that have been tried have resulted in mixed success. Shift-and-invert, a traditional way to precondition the eigenvalue problem [12], involves solving linear systems with $A - \sigma I$, where A is the original matrix and the shift σ is close to the desired eigenvalue. This method would be prohibitively expensive in our situation, given the size of the matrix and the number of times that $A - \sigma I$ must be factored. Alternatives based on an approximate factorization such as ILUT [13] are ineffective beyond the first few eigenvalues. Methods based on approximate inverse techniques have been somewhat more successful, performing better than filtering at additional preprocessing and storage cost. Preconditioning 'interior' eigenvalues, i.e., eigenvalues located well inside the interval containing the spectrum, is still a very hard problem. Current solutions only attempt to dampen the effect of eigenvalues which are far away from the ones being computed. This is in effect what is achieved by filtering and sparse approximate inverse preconditioning. These techniques do not reduce the number of steps required for convergence in the same way that shift-and-invert techniques do. However, filtering techniques are inexpensive to apply and result in fairly substantial savings in iterations.

4 Parallel Implementation

When highly optimized, an initial implementation of the code on the Cray C90 delivered a sustained 450 MFLOPs performance, despite the indirect addressing induced by its sparse, unstructured nature. As the cluster sizes become larger, so do the memory requirements of the code, and the relatively large memory resources of the C90 quickly become inadequate. In addition, execution times become excessively long. Parallel processing provides the only means of meeting these computational demands.

Recently, software libraries such as AZTEC [7], PETSC [9], and PSPARSLIB [8] have appeared which provide a framework for solving partial differential equations on distributed memory computers. Ease of use, portability, relatively good efficiency, and a selection of iterative solvers and preconditioners make them an attractive choice for developing parallel applications. Yet, for an application like ours these libraries cannot take advantage of the special form of the operator, since they require the matrix to be stored in some sparse format. Any efficiency gains from special data structures and highly optimized communication routines and eigensolvers will clearly pay off during the life-cycle of a production code. In the following, we describe an implementation of the code on the Cray T3D and T3E based on a specially designed extension of the PSPARSLIB library, and we outline our efforts to finely tune the code for efficiency.

4.1 Parallel Paradigm

For distributed memory parallel computers, the SPMD (Single Program Multiple Data) model has emerged as the most popular programming paradigm. In our implementation of the SCF procedure we have followed a hybrid of the SPMD and the master-worker paradigm. The master performs most of the preprocessing, computing of scalar values, and processing of the new potential at each SCF iteration. The master is also responsible for applying the mixing scheme on the potentials. The workers solve the eigenvalues and eigenvectors, update the charge density, and solve the Poisson equation for the Hartree potential in an SPMD fashion.

There are several reasons dictating the master-worker choice. First, there are some inherently sequential parts in the code which require large memory but short execution time. It is also common that one of the nodes in a parallel environment is equipped with larger memory than the others. 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 all would require an inordinate amount of effort with doubtful results as to the achievable gains. Third, this paradigm allows incremental parallelization of the code, implementing first the most time consuming procedures, such as the eigensolver, then gradually adding parallelism to other parts. Correctness of the code is also easier to maintain with this strategy. Finally, the resulting code is portable to other parallel platforms without requiring large amounts of memory for all the worker processors.

4.2 Problem Mapping and Partitioning

The primary sources of parallelism intrinsically available in the application are: (1) the multitude of required eigenvectors, and (2) parallelism from spatial decomposition. Assigning each processor the task of calculating all the eigenpairs in a segment of the spectrum would provide excellent coarse grain parallelism and parallel efficiency. For each eigenpair, one could use inverse iteration with some iterative method. However, the linear systems to be solved are highly indefinite and iterative methods for the inverse iteration converge extremely slowly. As was mentioned earlier shift-and-invert is impractical for the large matrices at hand. An alternative is to use a polynomial preconditioning approach. A polynomial p can be found such that the dominant eigenvalues of $p(A)$ are the transforms by p of the eigenvalues in the desired subinterval. Then these dominant eigenvalues and associated eigenvectors can be computed and the corresponding eigenvalues of the original matrix can then be evaluated. A major advantage with this approach is that global orthogonality does not need to be maintained since the eigenvectors of a Hermitian matrix are orthogonal if they are computed accurately enough; only eigenvectors associated with eigenvalues in a given subinterval must be orthogonalized during the computation. This is a workable approach but the book-keeping required in order to ensure that no eigenvalues are missed and that they are all represented only once may be quite cumbersome. In addition high degree polynomials may be needed that reduce the gains from parallelism.

Instead of this 'spectral decoupling' idea, we have adopted a domain decomposition approach based on partitioning the physical space. 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 to processors according to a partitioning of the physical domain. The subdomains can be chosen naturally as sub-cubes or slabs of the cube, but since the zero-charge areas can be arbitrarily distributed in the domain, a general partitioning is more appropriate. We have designed the mapping routines to be independent of the partitioner, requiring only a function $P(i,j,k)$ which returns the number of the processor where point (i,j,k) resides. This facilitates the use of many publicly available partitioning tools. In our experiments we have tested two ways of partitioning. The first is a greedy approach that optimizes load balancing by ordering the points and assigning the same number of points to each processor, but it ignores the amount of communication which is induced. The second approach uses the popular partitioning package METIS [10] which seeks to optimize both load balancing and the communication volume between processors.

Since the matrix is not actually stored, an explicit reordering can be considered so that the rows on 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, a sparse vector $U_{a,l,m}$ in eqn. (8) is partitioned according to the rows it contributes to. Even though the number of non-zero elements of the U-vectors is small, their partitioning is fairly well balanced if the matrix partitioning is well balanced. With this mapping, the large storage requirements of the program are distributed.

The tools that we developed for mapping, setting up the data structures and performing the communication are independent of the nature of our problem and can be embedded in other applications for unstructured stencil computations, that use any of the general data structures described in the following sections.

4.3 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 vector reduction operations (e.g., sdot operations) require a global number reduction (e.g., global sum) of the partial results on each processor.

The matrix-vector multiplication 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. In the parallel implementation communication is necessary, since some of the neighboring points of the local subdomain may reside on different processors. For this reason, each processor maintains the following data structure, which maps the local grid points to the local rows, and appends the needed interface points from other processors at the end of the local row list:

$$index(i,j,k) = \begin{cases} \text{row number in local ordering,} & \text{if } (i,j,k) \text{ is on local processor} \\ \text{index below local ordering,} & \text{if } (i,j,k) \text{ is a needed interface row} \\ \text{special index,} & \text{if } (i,j,k) \text{ is not considered (zero charge).} \end{cases} \quad (10)$$

The workers build this and other supporting data structures during the setup phase, by locating which of their rows are needed in the stencils of other processors. In the second step of the matrix-vector multiplication, this interface information is exchanged among nearest neighbors and the stencil multiplication can proceed in parallel. In the third step, each of the rank-one updates of the non-local components is computed as a sparse, distributed dot product. All local dot products are first computed before a global sum of their values takes place. The solution of eqn. (2) for the Hartree potential with the Conjugate Gradient method 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 an expensive phase and as the number of required eigenvectors increases, it is bound to dominate the cost. Reorthogonalization is performed every time a vector norm reduces significantly after orthogonalization. Although reorthogonalization recovers the numerical accuracy lost in the Gram-Schmidt procedure, it induces additional synchronization points. Yet, the amount of communication is negligible and therefore parallelization of this phase scales well as the size of the matrix grows. In the current application, global sums of the dot-products are delayed so that only one synchronization is needed. In addition, by performing the reorthogonalization test through easily obtained estimates of the vector norms, we introduce only two synchronization points in the procedure.

4.4 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. In an effort to identify the target production architecture, we have tested both implementations on a variety of parallel machines available at the University of Minnesota. On an SGI Power Challenge cluster the relatively slow communication between cluster nodes limits the scalability of the code. On an IBM SP2, the higher bandwidth of the Power Switch improves scalability, but only marginally. The main reason can be traced to the fast speed of the RS6000 processors on the SP2 and to the small problem size solvable on the less than 20 available nodes. In contrast, the less powerful nodes on the T3D and its very fast network provided better support for the fine granularity of our application. Moreover, the aggregate distributed memory from 512 T3D processors facilitates the solution of problems at least five times larger than on the SP2 or the Cray C90.

At the time we tested our code on the above machines, the native implementations of the MPI library were more efficient than the corresponding PVM ones. In some cases, certain point-to-point communication primitives were comparable, but performance of PVM collective operations lagged behind. Specifically on the T3D, the MPI EPCC implementation was still twice as fast as the native PVM one, and faster than the latest MPICH implementation for collective operations. The need for efficient communication prompted us to turn to MPI.

The small memories of the T3D nodes (64 MB) necessitated the use of the front-end Cray Y/MP as the master. Unfortunately, the MPI EPCC implementation did not allow message passing between the T3D and other machines, and using either MPICH or PVM would have increased the communication costs. To meet these contradictory requirements of the application, we used PVM to communicate between the master and the T3D and MPI EPCC for the communication among the workers. In this way, flexibility of the code was achieved 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 workers are spawned by the master through PVM. After the worker processes are initiated, they start MPI among themselves and create the necessary communication groups. This sequence of events was actually the only way to implement this combination scheme. Subsequently, all communication with the master was performed with PVM and the workers used the MPI reduction and all-to-all primitives to solve the eigenvalue problem. Another problem was the fact that PVM could not open more than 64 channels to the T3D simultaneously. Therefore, for large number of processors, messages to be broadcasted were sent to only one worker, which then broadcasted them using MPI. We also faced memory allocation problems because of PVM buffering and the small memory of the T3D nodes. For large cases, the memory available in the runtime stack was not sufficient to hold an incoming message, causing the process to crash. Therefore, large messages were to be split into manageable sizes and the receiving processor acknowledged reception to prevent buffer flooding by subsequent messages. This message chattering occurred only in the setup phase, and its cost was negligible.

4.5 Porting to the T3E

In July 1996, the T3D at the Minnesota Supercomputer Center was replaced by a 272-node Cray T3E-900 multiprocessor. Processor performance improved four to five times over the T3D, while the network gained a factor of 1.5 in speed, and the memory on each processor was doubled.

Porting the T3D code on the T3E was straightforward because of their similar architecture and operating system. The same PVM-MPI combination scheme was followed, with a Cray C90 as a master. Optimizing the code for the architecture of the T3E processors was more involved due to the presence of a second level cache, and six concurrent stream buffers between the cache and the main memory. We obtained significant performance improvements by unrolling the main matrix-vector multiplication loop to the appropriate depth, and by using the compiler option for stream buffer optimization. We also made a more drastic change to the data structure in (10). Besides the memory requirements on the processors, the neighboring points in the stencil (e.g., $index(i, j+1, k)$, $index(i, j, k-1)$) display no spatial memory locality causing severe cache misses. To face these problems, we dispensed with the array *index* and introduced on each processor an array JA which holds the stencil neighbors for each of the considered local grid points:

$$JA(i,j) = \begin{cases} \text{row number of the } i_{th} \text{ stencil-neighbor of local row } j \\ \text{special index if the } i_{th} \text{ neighbor is not considered (zero charge).} \end{cases} \tag{11}$$

The array JA can be viewed as a special case of the ELLPACK sparse storage format. The second (long) dimension involves only the local rows on each processor and therefore the memory requirements of JA are distributed.

5 Scalability Results

To demonstrate the scalability of the code, we run a few typical cases from our application problems, but of smaller size and with our greedy partitioning. The rationale is to present small cases whose performance will always be superseded by actual production runs. In addition, because of memory and time constraints we cannot test large problems on a variety of processor configurations. For each of the following cases we report speedups and timings for solving one eigenvalue problem from the SCF procedure.

The first test case is a small one, involving 47 Silicon atoms ($Si_{47}H_{60}$), with matrix size 29281, and 165 eigenpairs are required. The second case involves 191 Silicon atoms ($Si_{191}H_{148}$), with a matrix size of 83200, and 560 eigenvalues are required. Although the matrix is of modest size, the number of required eigenpairs is closer to the ones in typical production runs. The third was a system with 53 Ga and As atoms ($Ga_{27}As_{26}H_{41}$), with a somewhat larger matrix size of 145,000, 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. The curves for application speedup versus the ideal speedup appear in figures 3, 4 and 5 respectively.

The figures show that speedups are good, even for the small case of 47 atoms, and that they improve with the problem size, as expected. In figure 5 we notice only a small degradation in performance from 64 to 128 processors. In this case, the effects of the nearest domain communication start to show up, in spite of the twelfth-order finite difference scheme. These effects become more obvious in larger problems. Finally, we should point out that in the 53 GaAs atom case, more than 60% efficiency was obtained on 128 nodes, for a Davidson run that required only 3.7 minutes and 6618 matrix-vector multiplications to converge.

Besides the above scalability with respect to the matrix size, the code also demonstrates a scalability with respect to the number of eigenpairs. 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 cheaper than the communication step in matrix-vector multiplication. Figure 6 depicts this behavior. As a result, difficult problems requiring a large number of eigenvalues should exhibit good efficiencies.

While the node performance of the original code was impaired by inefficient cache usage on the T3E, the optimizations we outlined yield significantly better cache utilization. The resulting code on one T3E processor runs more than four times faster than its counterpart on the T3D.

Because the processor speed on the T3E is enhanced substantially more than the network bandwidth, relative to the T3D, communication costs are more visible in our timings. Figure 7 shows the speedup on the T3E for the 191 Si atom case. Although the curve lies slightly below the one in figure 4, it is still satisfactory, yielding 78% efficiency on 32 nodes and in less than one third of the time.

6 Physical Applications: The Quantum Dot

Recently, optical properties of semiconductor quantum structures with reduced dimensions have been of great experimental and theoretical interest. While silicon is the material of choice for electronic devices, it is not an efficient material for opto-electronic applications as its energy gap does not reside in the visible part of the spectra. However, if one considers small clusters of silicon, *e.g.*, clusters of 20-100 Å, the optical gap is strongly modified from the crystalline gap. Localizing the optical excitation to a small spatial regime (such as a cluster) is called *quantum confinement.* There is abundant experimental evidence for this effect. In particular the discovery of visible luminescence from porous Si [16] has focused attention on understanding quantum confinement. In porous silicon, there are large voids connected and interlaced by fine "wires" and localized regions of silicon. It is widely thought that localizing the optical excitation in these regimes strongly modifies the optical absorption properties of silicon. Likewise, small spherical clusters of silicon have been prepared. Because of the geometry, these clusters are known as quantum dots [17]. The dots are easier to examine from a theoretical perspective as the geometry is straightforward. The interior of the dots are thought to consist of silicon atoms in a regular crystalline environment. The surface of the dots are often electrically passivated with hydrogen atoms or polymer coatings.

Since quantum dots involve systems with hundreds to thousands of atoms, there has been very little theoretical work on these systems. Most of the theoretical approaches to these systems have focused on empirical or semi-empirical approaches. These approaches are subject to several assumptions, *e.g.*, the transferability of crystalline interactions to confined regime. *Ab initio* methods do not suffer from such assumptions as the electronic states are allowed to respond the geometrical configuration via a self-consistent relaxation. Moreover, we can test these methods for smaller clusters where the empirical methods are known to fail. However, due to large computational demands, *ab initio* calculations have so far been limited to small systems (≤ 15 Å in diameter) which usually do not correspond to the sizes of quantum dots for which experimental data is available (≥ 20 Å).

The parallel code has enabled us to perform *ab initio* calculations for two large Si quantum dots of compositions $Si_{293}H_{172}$ and $Si_{525}H_{276}$, corresponding to diameters of 22.4 Å, and 27.2 Å, respectively. The atomic structure of the larger quantum dot ($Si_{525}H_{276}$) is shown in figure 8. We used a finite difference grid spacing $h = 0.9$ atomic units and required the wavefunctions to vanish outside a spherical domain which was at least 7 atomic units away from the last shell of Si atoms. These parameters can be estimated via calculations for a single atom and verified by decreasing the grid spacing and increasing the spherical domain size. With these parameters, the calculation for the $Si_{293}H_{172}$ cluster required the self-consistent solution of a Hamiltonian of size $\approx$ 125,000 using 800 eigenpairs, while the corresponding Hamiltonian sizes and the number of eigenpairs for $Si_{525}H_{276}$ were 200,000 and 1350, respectively. The calculations for the two clusters were performed on 48 processors on the T3E; the minimum configuration that could accommodate the large case. Self-consistency was reached in approximately 13 diagonalizations. The full self consistent calculations for $Si_{293}H_{172}$ and $Si_{525}H_{276}$ took 4.5 and 20 hours, respectively.

The optical gap of the quantum dot can be calculated from a knowledge of the energy to add an electron (the electron affinity), E_a, and to remove an electron (the ionization energy), E_i [18]. From Eq. 8, we can write these energies as $E_a = E_{total}(N+1) - E_{total}(N)$ and $E_i = E_{total}(N-1) - E_{total}(N)$ where N is the number of electrons in the dot. The difference in these energies is the band gap, or quasiparticle gap: $E_{qp} = E_i - E_a$. Physically, this energy corresponds to the energy required to form a non-interacting electron-hole pair created when a photon is absorbed [1, 18]. To determine the optical gap, we must include the interactions between the hole and the electron. Details of the techniques used for this can be found elsewhere [18].

As a result of these calculations, we have been able to show that fundamental quasiparticle gaps are enhanced substantially (by 1-2 eV) with respect to bulk values and decrease upon increasing the cluster size as a result of quantum confinement. The power law we find is inconsistent with simple models of quantum confinement. Also, the quasiparticle gap values obtained from first principles were also found to be much larger than those obtained from semi-empirical calculations due to the approximations used in the latter. The calculated optical gaps were found to be in excellent agreement with absorption data from Si quantum dots as shown in figure 9 [18].

Our results demonstrate that it is entirely feasible to apply electronic structure codes to systems of $\sim$1,000 atoms or more. Moreover, our work illustrates the possibility of investigating the evolution of the optical gap from atoms to clusters to crystalline matter. We are now in a position to determine quantitatively how electronic interactions evolve as a function of size.

7 Conclusions

There are several challenges arising in the electronic structure problem. These challenges can be met by a combination of algorithmic advances and parallel processing. Despite the unstructured sparsity of the problem, our implementations on the T3D and T3E demonstrate good scalability. To achieve this, we have developed tools as an extension to the PSPARSLIB library which can easily be incorporated in the library for use in matrix-free calculations of similar applications. Moreover, a combination of the PVM and MPI communication libraries is necessary for optimal parallel performance. Finally, even on this rather simple heterogeneous parallel environment, several issues should be addressed to obtain high performance.

The parallel code has been used successfully to determine properties of clusters of more than 800 atoms. In the future, it will enable us to explore new exciting problems that involve even larger systems.

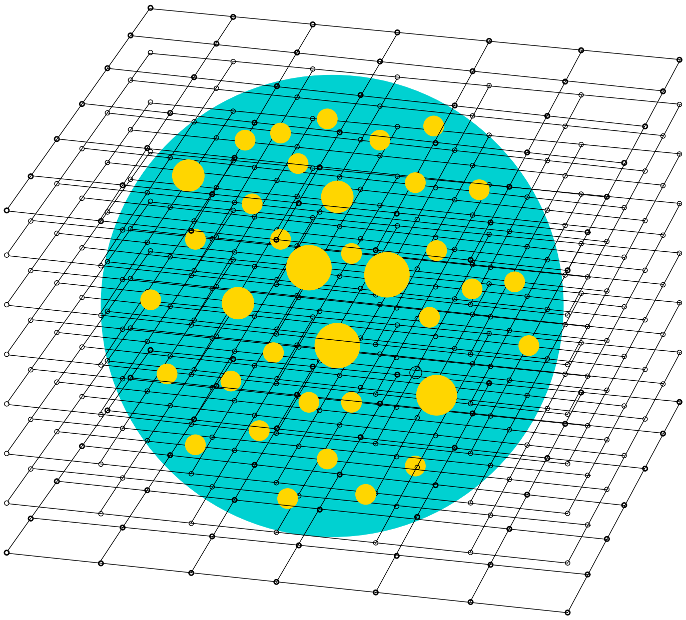

Figure 1: Uniform 3-D grid where only the points with non-zero charge density (blue sphere) are considered for constructing the matrix. Away from the atoms in the system (yellow spheres), the charge density decreases rapidly to zero. Other geometries can be chosen as the blue region to inscribe more complex atomic structures.

Figure 2: An example partitioning of the domain on the procesors. Although shown as sub-cubes, these domains can be chosen with arbitrary, unstructured geometries. Each processor considers only the non-zero charge points (blue area) around those atoms (yellow spheres) which are local in its assigned subdomain. Special data structures are used for storing these points and for communicating the stencil information with neighboring processors.

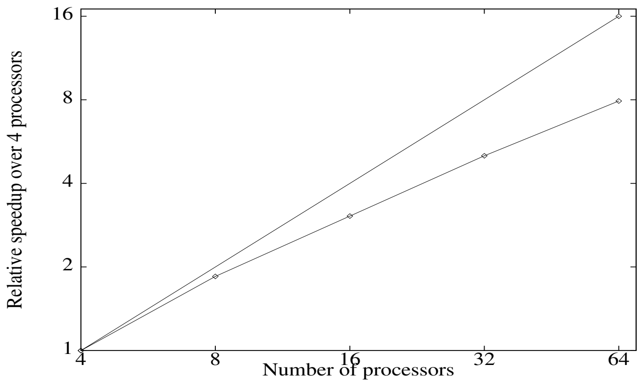

Figure 3: T3D speedup for the 47 Si atom case. Time for one iteration on 64 nodes: 4.7 min.

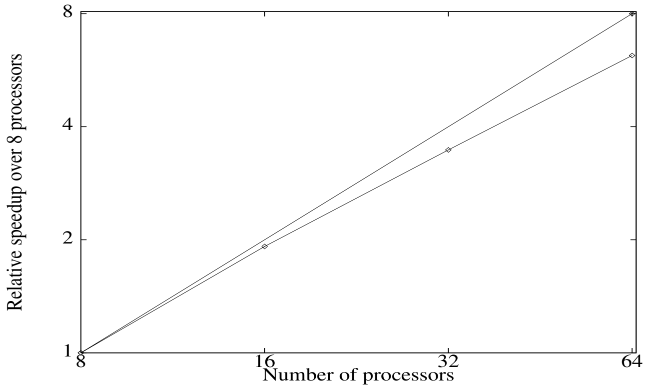

Figure 4: T3D speedup for the 191 Si atom case. Time for one iteration on 64 nodes: 31.6 min.

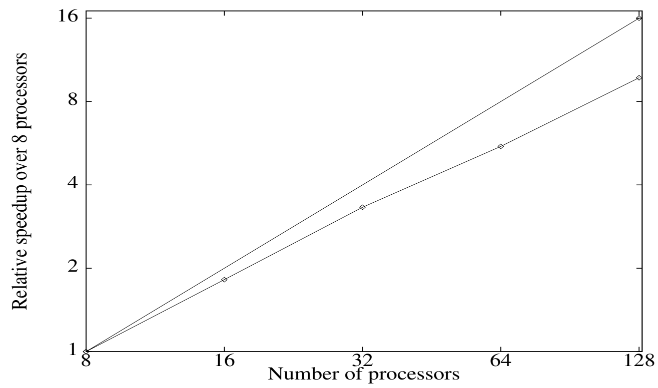

Figure 5: T3D speedup for the 53 GaAs atom case. Time for one iteration on 128 nodes: 3.7 min.

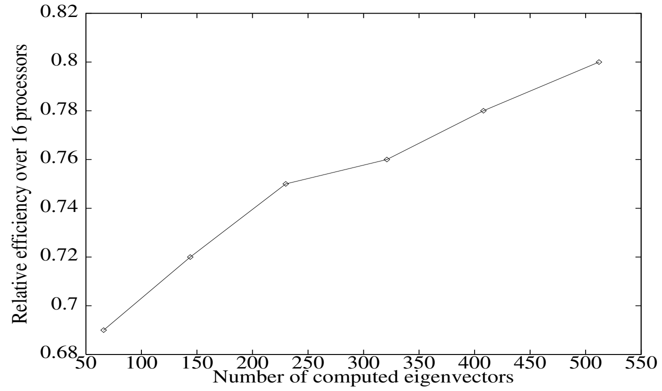

Figure 6: Efficiency on 64 T3D processors for the 191 Si atom case, as a function of converged eigenvectors.

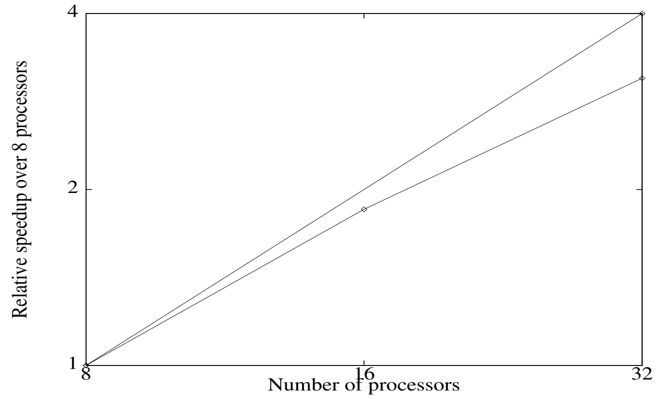

Figure 7: T3E speedup for the 191 Si atom case. Time for one iteration on 32 nodes: 19 min.

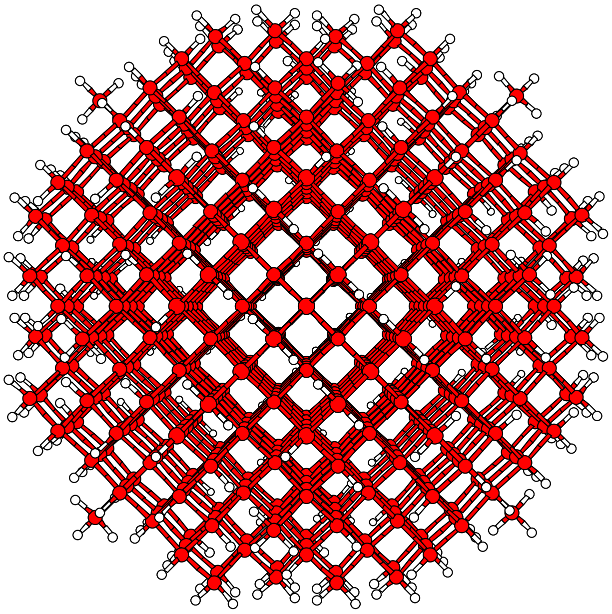

Figure 8: Atomic structure of $Si_{525}H_{276}$

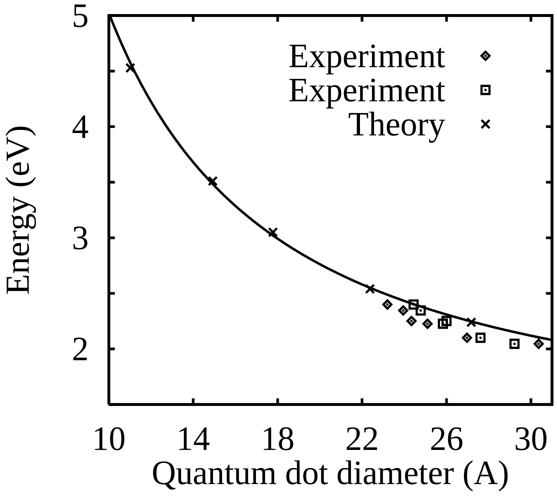

Figure 9: Calculated optical gaps (shown by × fitted to the solid line), and experimental absorption data from Si nanocrystallites (□ and ◇ from Ref. [6]) as a function of the quantum dot diameter (in Å). The two sets of experimental data (◇ and □) differ by the method to estimate the nanocrystalline size.

References

[1] J.R. Chelikowsky, and S.G. Louie, editors: *Quantum Theory of Real Materials*, (Kluwer Press, 1996).

[2] W. Kohn and L. J. Sham, Phys. Rev., 140, A, 1133 (1965); P. Hohenberg and W. Kohn, Phys. Rev. **136**, B864 (1964); S. Lundqvist and N.H. March, *Theory of the Inhomogeneous Electron Gas*, (Plenum, NY), 1983 and refences therein.

[3] J.R. Chelikowsky and M. L. Cohen, "Ab initio Pseudopotentials for Semiconductors," *Handbook on Semiconductors*, edited by P. T. Landsberg (Elsevier, Amsterdam, 1992), Vol. 1, p. 59.

[4] L. Kleinman and D. M. Bylander, "Efficacious form for model pseudopotentials" Phys. Rev. Lett. **48**, 1425 (1982).

[5] 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).

[6] S. Furukawa and T. Miyasato, Quantum size effects on the optical band gaps of microcrystalline Si:H, *Phys. Rev. B* 38, 5726-5729 (1988).

[7] S. A. Hutchinson, J. N. Shadid and R. S. Tuminaro, Aztec User's Guide, Massively parallel computing research laboratory, Sandia National Laboratories, 1995.

[8] A. V. Malevsky and Y. Saad, Data structures, computational, and communication kernels for distributed memory sparse iterative solvers, Parallel Computing Technologies, Third International Conference, PaCT-95, 252-7, (1995).

[9] S. Balay, W. Gropp, L. C. McInnes, B. Smith, mboxhttp://www.mcs.anl.gov/petsc/petsc.html.

[10] G. Karypis and V. Kumar, Parallel multilevel graph partitioning, Proceedings of the 10th International Parallel Processing Symposium, 314-19 (1996).

[11] 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).

[12] B. N. Parlett, The symmetric Eigenvalue problem, Prentice-Hall, (Englewood Cliffs, NJ), 1980.

[13] Y. Saad, Iterative methods for sparse linear systems, PWS Publishing Company, 1996.

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

[15] 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).

[16] L. T. Canham, Silicon quantum wire array fabrication by electrochemical and chemical dissolution of wafers, *Appl. Phys. Lett.*, 57, 1046–1049 (1997).

[17] R. T. Collins, P. M. Fauchet, and M. A. Tischler, Physics Today, "Porus silicon : From luminescence to LEDs" **50** (1), 24 (1997).

[18] S. Öğüt, J. R. Chelikowsky, and S. G. Louie, Quantum Confinement and Optical Gaps in Si Nanocrystals, *Phys. Rev. Lett.*, 79, 1770-1773 (1997).

[19] E. L. Briggs, D. J. Sullivan, and J. Bernholc, Large Scale Electronic Structure Calculations with Multigrid Acceleration, Phys. Rev. B **52**, 5471-5474 (1995).

[20] F. Gygi and G. Galli, Real Space Adaptive Coordinate Electronic Structure Calculations, Phys. Rev. B 52, 2229-2232 (1995).

[21] G. Zumbach, N. A. Modine, E. Kaxiras, Adaptive coordinate, real-space electronic-structure calculations on parallel computers, Solid State Commun. **99**, 57-61 (1996).