Solution of Large Eigenvalue Problems in Electronic Structure Calculations*

Y. Saad[†], A. Stathopoulos[†], J. Chelikowsky[‡], K. Wu[†], and S. Öğüt[‡]

December 1995

Abstract.

Predicting the structural and electronic properties of complex systems is one of the outstanding problems in condensed matter physics. Central to most methods used in molecular dynamics is the repeated solution of large eigenvalue problems. This paper reviews the source of these eigenvalue problems, describes some techniques for solving them, and addresses the difficulties and challenges which are faced. Parallel implementations are also discussed.

1 Introduction

One of the fundamental problems encountered today in chemistry and physics is to understand the dynamics of microscopic particles. It is possible to explain or predict certain material properties at a microscopic scale from knowledge of their initial state and given external perturbations. For example, it may be desirable in certain cases to follow the dynamics of atoms/electrons until a 'steady state' corresponding to minimum total energy is reached.

The numerical problems to be solved in these calculations are among the most challenging in computational sciences today. The central computation in this minimization is the repeated solution of a large, symmetric eigenvalue problem. Traditional approaches use a plane-wave basis to expand the required wave-functions (eigenvectors). This approach is comparable to spectral techniques used in solving other types of partial differential equations. A projection process on the finite plane-wave basis gives rise to a discrete matrix eigenvalue problem which is dense. The matrix is not formed explicitly, but matrix-vector product operations are performed with the help of FFT transforms. An alternative to this approach based on high-order finite difference schemes has recently been advocated. For localized systems, it has proved to be as accurate and more efficient than plane-wave techniques[6, 7, 16, 15]. The matrices resulting from both finite difference methods and plane-wave techniques are large, and the number of eigenvalues and eigenvectors required is proportional to the number

*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

of atoms in the system. Herein lies the challenge. For complex systems involving hundreds or thousands of atoms, more accuracy is required, which means larger matrices, but also more eigenvalues are needed, due to the physics.

The size of the eigenvalue problem, the large number of required eigenpairs, as well as its repeated solution for each molecular dynamics step, pose huge computational demands on high performance computers and eigenvalue solvers. In this paper, some of the efforts for meeting these demands are described. We present an eigenvalue solver based on the preconditioned Lanczos (generalized Davidson) method [11, 8, 23] which can tackle hundreds of eigenpairs. We also address several issues such as efficient preconditioning, matrix-vector multiplication, orthogonalization, as well as robustness.

Parallel processing can help make substantial improvements in solution times and thus play a crucial role in meeting some of the challenges posed by electronic structure calculations. A parallel implementation of the application on a cluster of workstations is outlined. In particular, we discuss an efficient design of the partitioning scheme and other preprocessing techniques needed to reduce the impact of frequent communication required during the iteration process.

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:

$$\mathcal{H}\Psi = \mathcal{E}\Psi, \tag{2.1}$$

where $\mathcal{H}$ is the Hamiltonian operator for the system and $\mathcal{E}$ the total energy. In quantum mechanics, $|\Psi(r)|^2$ represents the charge density at a location r in space. In its original form the operator $\mathcal{H}$ is very complex, involving sums over all electrons and nuclei, Laplacian related to each nucleus, etc. However, most theories of condensed matter systems make two fundamental approximations which make the problem more tractable. These are the Born-Oppenheimer approximation and the one-electron approximation [20].

With these approximations the following simplified form of the Schrödinger equation is obtained:

$$\left[\frac{-\hbar^2\nabla^2}{2m} + V_{tot}[\rho(r), r]\right]\psi(r) = E\psi(r), \tag{2.2}$$

where the Laplacian is the kinetic operator, $\hbar$ 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 $\rho(r)$. The charge density function in turn depends on the wavefunctions of the above equation via,

$$\rho(r) = \sum_{\text{occupied states}} |\psi_i(r)|^2. \tag{2.3}$$

The dependence of the potential V_{tot} on $\rho(r)$ is a complicated nonlinear function. 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 [20], and the use of Pseudopotentials, the potential V_{tot} is simplified to a summation of three distinct terms, specifically,

$$V_{tot} = V_{ion} + V_H + V_{xc} \tag{2.4}$$

where V_{ion} is the ionic potential, V_H is the Hartree potential, and V_{xc} is the exchange-correlation potential. The exchange-correlation potential V_{xc} is easily approximated by a potential from the local density expression. Once the charge density $\rho(r)$ is known, the Hartree potential is obtained by solving the Poisson equation:

$$\nabla^2 V_H = -4\pi\rho(r). \tag{2.5}$$

This can be solved either 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[7]:

$$V_{ion} = \sum_a V_{loc}(|r_a|)\psi + \sum_{a,\ lm} \frac{\Delta V_l(r_a)}{< \Delta V_{lm}^a >} u_{lm}(r_a) \int u_{lm}(r_a)\Delta V_l(r_a)\psi d^3r, \tag{2.6}$$

where $< \Delta V_{lm}^a >$ is the normalization factor,

$$< \Delta V_{lm}^a > = \int u_{lm}(r_a)\Delta V_l(r_a) u_{lm}(r_a) d^3r. \tag{2.7}$$

Here, the superscript a denotes an atom in position R_a, $r_a = r - R_a$, and the functions u_{lm} are the atomic pseudopotential wave functions of angular momentum associated with quantum number l, m, 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. What is of interest to matrix computations specialists is the nature of the non-local term in (2.6). If we set $U_{a,l,m}(r_a) = \Delta V_l(r_a)u_{lm}(r_a)$ and we consider the discretization of the equation, the non-local term becomes a sum of rank-one updates:

$$V_{ion} = \sum_a V_{loc}\psi + \sum_{a,\ l,m} ct_{a,l,m} U_{a,l,m} U_{a,l,m}^T \psi, \tag{2.8}$$

where $ct_{a,l,m}$ are normalization coefficients and all other variables are the discretized matrix and vector forms of the operators and functions.

From (2.3), ρ can be computed from the wavefunctions ψ_i for all occupied states. However, ψ_i are the solution of the eigenvalue problem in (2.2) whose coefficients depend on the potential. In turn, the V_H potential depends on ρ. The above constitute a set of non-linear equations, which are solved by the so-called Self Consistent Field (SCF) iteration. This algorithm can be viewed as an inexact Newton method for accelerating the fixed-point iteration $V_{tot}^{new} = \mathcal{M}(V_{tot})$

in which $\mathcal{M}(V_{tot})$ represents the new potential obtained from solving (2.2) and using (2.4) with the resulting wavefunctions and values. The term 'Mix' used in line 4, refers to any quasi-Newton approach for this acceleration.

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 (2.5), and V_{tot}.*
4. *'Mix' potentials with a Broyden-type quasi-Newton approach.*
5. *If self-consistent stop, else repeat from 1.*

In matrix form, the Hamiltonian is the sum of a Laplacian matrix, three diagonal matrices and a matrix representing the nonlocal contributions. Two of three diagonal matrices arise from discretizing V_{xc} and V_H. The third is due to the local part of the ionic potential. The nonlocal matrix is the sum of simple rank-one updates over all atoms and quantum numbers. Typically, a small number of steps is required for the SCF iteration to converge.

3 Solving the Eigenvalue Problem

3.1 Problem Setup

In the SCF iteration the solution of the eigenvalue problem is the dominant computation, comprising more than 80% of the total time. Traditionally, a plane-wave basis has been used to expand the wavefunctions and obtain a matrix eigenvalue problem. The problem is solved in Fourier space and the solution is transferred back to real space through fast Fourier transforms (FFT). This method is very similar to spectral methods used in solving certain types of Partial Differential Equations. The basis used is of the form: $\{e^{ik.r}\}_k$, where k is the wave number, a vector with three components (k_x, k_y, k_z) and r is the variable, a vector of three components (r_x, r_y, r_z) representing an arbitrary location in three-dimensional space. When representing the Hamiltonian $H = \Delta + V$, with respect to this basis, the Laplacian term Δ gives rise to a diagonal matrix while the potential term V gives rise to a dense matrix. These matrices are never formed explicitly however.

There are two main advantages of this approach. First, the pseudo-wavefunctions for isolated atoms are smooth and plane-waves are a reasonable basis, providing rapid convergence in Fourier space. Second, it is easy to precondition the resulting matrix in Fourier space by exploiting the fact that a smaller subset of planewaves gives a good approximation to the large matrix. One technique that has been used with success [32] is to approximate the desired eigenvalues by using a smaller planewave basis, and the larger ones by using the eigenvalues of the Laplacian ignoring the potential terms. An approximation to $(H - \sigma I)^{-1}$ can be easily built this way.

However, there are several disadvantages to this approach. In case of non-periodic systems such as clusters, liquids or glasses, the plane-wave basis can only be applied if a *supercell method* [4] is employed. This is very similar to the situation with spectral methods. Spectral methods are excellent at solving problems with rectangular geometries which have periodic boundary conditions. The supercell technique artificially repeats the localized configuration to impose periodicity on the system. Difficulties arise when this basis fails to replicate the localized system and vacuum regions at the same time, and when cells interact with each other. Besides these physical problems, the eigenvalue problem can be expensive to solve mainly because the matrix is dense. Although this dense matrix is not explicitly formed, two FFTs must be carried out at each step of the iteration procedure. This can be costly, especially in high performance environments [3]. Further, the size of the matrix increases with the number of atoms, and for large problems the cost of these methods is prohibitive.

In recent years, finite difference techniques have started to challenge planewave techniques. Surprisingly, these methods have only appeared sparingly in the area the last half century [17, 12]. Our approach combines finite differences with a high-order expansion of the derivatives [3, 13, 5]. Since the pseudo-wavefunctions for isolated atoms are smooth, the wave-functions can be expanded locally in Taylor series and a good approximation is expected as with plane-waves. In addition, the supercell problems are overcome, the eigenvalue calculations can be performed efficiently in real space using sparse matrices, and the codes are much easier to parallelize. There are two disadvantages with finite differences: larger matrices may be required to reach the same accuracy as with plane-waves, and these are more difficult to precondition. Nevertheless, the size of the matrices increases more slowly with the number of atoms in finite differences than in the plane-waves, and the method is fast. In fact, the new method can be an order of magnitude faster than a traditional one based on planewaves (Figure 3.1).

The basic grid is a uniform three dimensional cube. However, it is often the case that many points in the cube are 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 spheres surrounding the atoms, and cylinders connecting such spheres. In some cases, the size of the Hamiltonian is decreased by a factor of two to three. 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 (2.8).

3.2 The Preconditioned Lanczos Algorithm

Iterative methods which require the matrix only in the form of matrix-by-vector products are the most common means of solving large eigenvalue problems such as the ones considered here. In addition to their sheer size, the eigenvalue problems in this application present other difficulties. First, the number of required eigenvectors is proportional to the atoms in the system, and can grow

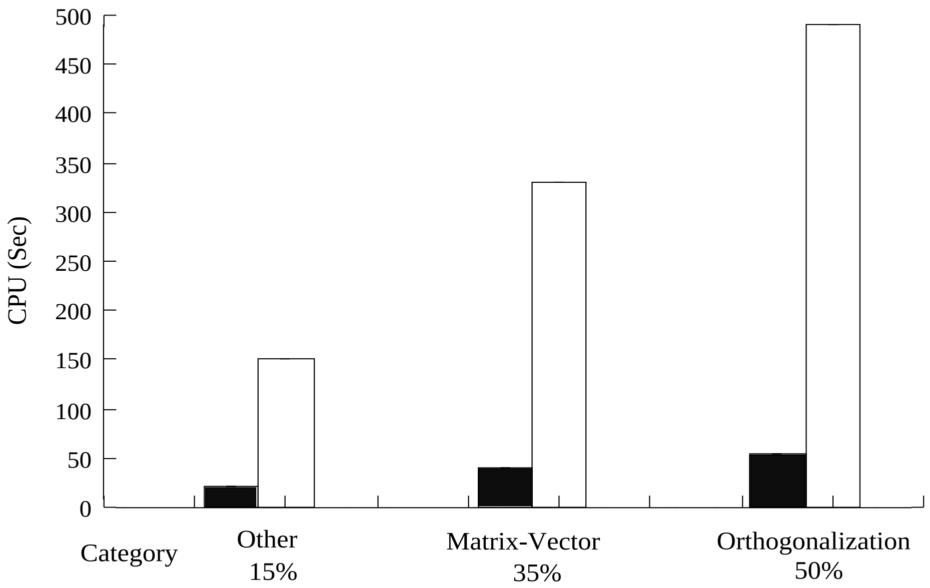

Figure 3.1: CPU time comparison of FD (filled bars) and plane-waves on one processor Cray C90.

up to several hundreds or even thousands. Maintaining the orthogonality of these vectors is an additional problem. Second, the separation of the eigenvalues which determines the rate of convergence of iterative solvers becomes increasingly poor as the matrix size increases. Preconditioning techniques attempt to alleviate this problem. Finally, for every eigenvalue problem to be solved, 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 [11, 25]. In its original form it used only diagonal (Jacobi) preconditioning, and some general purpose Davidson codes are available [28]. In this application, a code based on the preconditioned Lanczos (generalized Davidson [23]) has been developed. This code addresses the problems mentioned above, and it is has been favored over other Lanczos codes mainly for its preconditioning flexibility and the exploitation of initial guesses. Moreover, effective Lanczos codes are used without restarting. Besides the significant storage requirements in these cases, maintaining orthogonality, or detecting spurious eigenvalues may be very expensive [24, 9]. The algorithm is sketched below for the case where the smallest eigenvalue of a matrix A is computed. The basis size is limited to m vectors, and at each step a preconditioner $M_{(j)}$ is applied. MGS denotes the modified Gram-Schmidt procedure.

ALGORITHM 3.1. **Preconditioned Lanczos**

0. Choose an initial unit vector v_1.
1. Until convergence Do:
2.   *Set* $V_1 = [v_1]$.
3.   *For* $j = 1, \ldots, m$ *Do:*
4.   *Compute* $w := Av_j$.
5.   *Compute* $V_{(j)}^T w$, *the last column of* $H_{(j)} := V_{(j)}^T A V_{(j)}$.
6.   *Compute the smallest eigenpair* λ, y *of* $H_{(j)}$.
7.   *Compute* $u := V_{(j)} y$ *and* $r := Au - \lambda u$.
8.   *Test* $\|r\|$ *for convergence. If satisfied Return.*
9.   *If* $j < m$ *compute* $t := M_{(j)}^{-1} r$
10.   *and* $V_{(j+1)} := MGS([V_{(j)}, t])$.
11.   *Enddo*
12.   *Set* $v_1 := u$ *and restart.*
13. Enddo

To compute the residual r in line 7, without an additional matrix-vector product operation, the vectors $w_j = Av_j$ are saved in $W_{(j)}$ along with the basis vectors $V_{(j)}$. It is then possible to compute r as $r := W_{(j)} y - \lambda u$.

As the number of required eigenvectors increases, the number of basis vectors m should also increase. The targeted applications of the method involve about a thousand eigenpairs. For these applications, the storage demands of both V and W arrays are very large. Disk storage is a rather expensive alternative since the basis vectors are needed in several steps of the algorithm. Another problem with large basis sizes is the solution of the Rayleigh-Ritz problem in step 6. We need to solve dense matrix eigenvalue problems of size, say, 1000, at every step.

The solution provided in the current program is implicit deflation. Every time an eigenvector converges, it is locked out of the current basis and all basis vectors are orthogonalized against it. Therefore eigenvectors are not altered after they are locked out. Notice that the work space required for the basis and the K eigenvectors sought can be viewed as:

$$\overbrace{u_1, u_2, \ldots, u_k,}^{\text{Converged eigenvectors}} \quad \overbrace{v_1, v_2, \ldots, v_m,}^{\text{Basis vectors}} \quad \text{for } k < K, \text{ and a user defined } m.$$

The benefits are twofold. First, the Rayleigh-Ritz problem grows only up to the user defined size m. Second, since only v_i contribute to the small problem, only $w_i, i = 1, \ldots, m$ need to be stored. Therefore, the storage requirements of the method reduce from $O(2K)$ to $O(2m + K)$, and the small problem is much easier to solve.

There are some considerations with using this deflation scheme. The first involves the choice of a proper convergence test, and the second the choice of the targeting vector, i.e., which approximation to improve in the next step. Deciding on convergence of some eigenvector can be misleading, since a poorly converged locked eigenvector (e.g., u_1) may hinder convergence to the following eigenvectors (e.g., $u_2, \ldots$). On the other hand, extreme eigenvector accuracy is wasteful since low levels of accuracy are often adequate in the SCF iteration. In our approach a heuristic is used in which the required accuracy is decreased as more eigenvectors inside the spectrum converge. In case of a few tens of eigenvalues, the convergence criterion for the residual of each eigenpair $i = 1, \ldots, K$ is $\|res_i\|\ i^2 < \epsilon$, while in case of hundreds of eigenvalues, the criterion used is $\|res_i\|(K - i + 1) < \epsilon$. This usually eases convergence of inner eigenpairs and also saves considerable time.

The issue of targeting is handled by the use of a window of possible targets from the set of basis vectors. The window size is $p < m$, and depending on a user option, up to p eigenvectors can be targeted simultaneously (block Davidson [22]). The robustness of the code is enhanced by carefully selecting and locking the converged eigenpairs from the targeted ones. This scheme significantly reduces the possibility of "missing" an eigenpair.

Finally, orthogonalization is a very expensive part of the code since it involves all converged and basis vectors. Nonetheless, it is a crucial one. Solutions cannot be obtained if the basis vectors loose their mutual orthogonality. For that purpose, a reorthogonalization procedure has been employed similar to the one proposed in [10]. Whenever the norm of a vector is reduced below a threshold during the orthogonalization, the procedure is repeated. The procedure is cheaper than full reorthogonalization, yet provides adequate accuracy. Currently, we are looking at alternative ways to avoid or reduce orthogonalization constraints imposed by this problem.

3.3 Preconditioning

As the size of the Hamiltonian grows, the relative separation of the eigenvalues becomes smaller, causing slow convergence of the Lanczos process. Although the lowest eigenvalues remain roughly the same with finer discretization, the larger ones are less affected by the local and non-local potentials and are therefore governed by the largest eigenvalues of the Laplacian. One strategy is to focus on this part of the Hamiltonian.

In the earlier plane-wave version of the code, subspace iteration with Jacobi preconditioning was used [32]. This worked well because of the diagonal form of the Laplacian in Fourier space. In real space, the Laplacian consists of a number of diagonals and Jacobi preconditioning is not effective. An idea would be to solve only the Laplacian in Fourier space. This can be performed very efficiently with fast Poisson solvers for $(\nabla^2 - \sigma I)x = f$. However, it is not clear how efficient it will be for eigenvalues well inside the spectrum. Also, FFT transforms of two vectors are needed at each step adding on the expense of the preconditioned Lanczos.

One preconditioner used in our approach is based on a filtering idea and the fact that the Laplacian is an elliptic operator [31]. The eigenvectors corresponding to the few lowest eigenvalues of ∇^2 are smooth functions. Physically, the wavefunctions corresponding to the lowest occupied states are smoothly varying functions. When an approximate eigenvector is known at the points of the grid, a smoother eigenvector, and therefore a better approximation, 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:

$$V_{i,j,k} := \frac{V_{i-1,j,k} + V_{i,j-1,k} + V_{i,j,k-1} + V_{i+1,j,k} + V_{i,j+1,k} + V_{i,j,k+1}}{12} + \frac{V_{i,j,k}}{2} \tag{3.1}$$

This preconditioning has provided significant improvements on the iteration count of preconditioned Lanczos. However, this is only true for the lowest few (few tens) eigenvalues. Eigenvectors associated with higher eigenvalues may not benefit as much from the above filter, possibly due to the lower degree of smoothness of the associated eigenfunction. In case of hundreds or thousands of eigenpairs, new preconditioners should be devised that extend the above idea of filtering. A major weakness of current eigenvalue solvers is that they do not address the issue of preconditioning for eigenvalues that are well inside the spectrum. Direct solution techniques are the only known means of preconditioning which work well in this situation. However they are excessively costly. Despite the above, preconditioner (3.1) has been used successfully with very large calculations, where 800 eigenpairs are required. In these calculations, an average of 30 matrix-vector multiplications per eigenpair sought is observed.

4 Parallel Implementations

To study real systems with thousands of atoms it is vital to be able to reduce the execution time by several orders of magnitude. As is often the case in such calculations, a combination of efficient algorithms and powerful computing platforms can provide gains in time that can help solve problems that would otherwise be practically infeasible. Finite difference techniques provided one order of magnitude improvement over of plane-wave techniques. The choice of better preconditioners and eigenvalue codes might provide a significant additional factor. These improvements must be realized with parallel computing in mind. There are few limits in the gains that can be achieved through parallel processing, safe for the need to use appropriate algorithms.

4.1 Parallel environment and models

The parallel implementation of the current application uses the PVM communication library [14, 29]. The parallel environment used for the development is a cluster of four Silicon Graphics Power Challenge workstations, each with four processing elements, interconnected through HiPPI and Fiber channels. A cluster of eight IBM 590 workstations, interconnected through an A.T.M. An IBM SP-2 multiprocessor with 10 processors will soon be incorporated as well. The architecture of the SGI symmetric multiprocessors encourages the use of a hierarchical programming model with shared memory paradigm among intra-node processing elements and message passing among inter-node workstations.

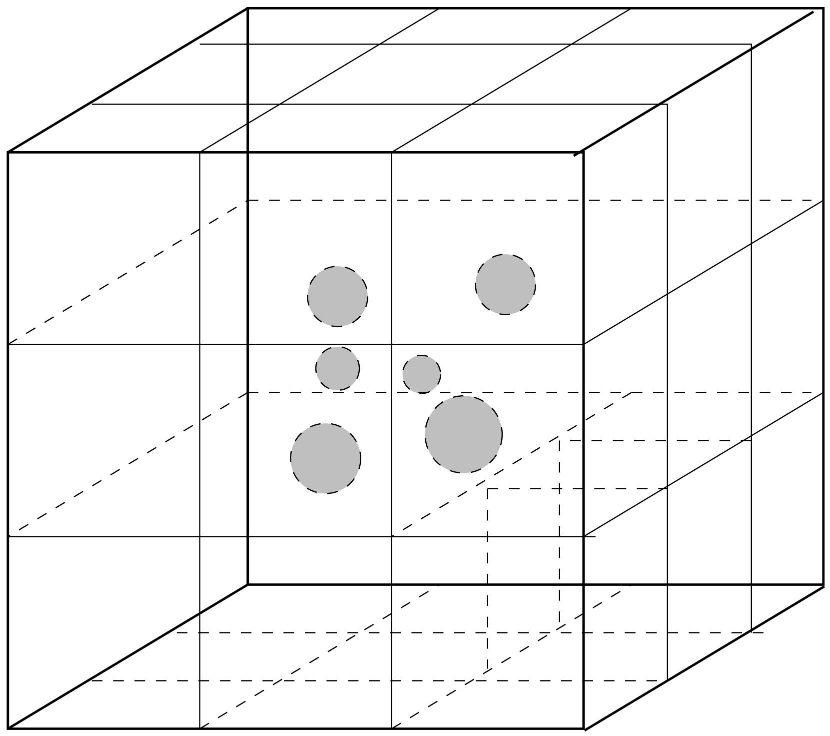

Figure 4.1: Partition of the domain into subcubes. The spheres represent the non-zero charge-density areas around each atom.

Although the program is based on PVM and the message passing paradigm, directives and compiler optimizations can easily be performed so that each workstation runs with one process but uses multithreading to parallelize each loop, as in shared memory machines.

The master-slave model 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 are assigned all the computationally intensive work: solving for the eigenvalues and eigenvectors, updating the charge density, and solving the Poisson equation for the Hartree potential.

4.2 Problem Mapping

The problem is mapped onto the processors in a data parallel way. 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 are naturally chosen as sub-cubes or slabs of the domain as illustrated in Figure 4.1. The program only requires a function $P(i, j, k)$ that returns the processor number where point (i, j, k) resides. Arbitrary partitionings are thus possible, although it is not clear whether other ones than the ones mentioned may be beneficial. 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 completely distributed and large problems can be solved on the aforementioned SGI cluster.

4.3 Parallel Preconditioned Lanczos

In the preconditioned Lanczos algorithm, the basis vectors V, and the auxiliary vectors W 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, which is a special case of the PSPARSLIB library [26]. 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 among the appropriate processors. In the second step of matrix-vector multiplication, this information is exchanged among the processors and the stencil multiplication can proceed completely 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. Notice that solving (2.5) for the Hartree potential with Conjugate Gradient, also requires a matrix-vector multiplication with the stencil. The procedure is equivalent to the second step above.

Orthogonalization is a very expensive phase of the preconditioned Lanczos, 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 have been delayed so that only one synchronization is needed for their computation. 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. We are currently working on an orthogonalization procedure which is based only on BLAS-3 routines, and it also involves only two synchronization points.

Finally, the preconditioning used is still the averaging scheme over the neighbors. However, the use of the stencil requires the same communication pattern as in matrix-vector multiplication. Experiments have shown that if the averaging is limited on the local rows on each processor, the number of Lanczos iterations does not increase considerably, but the total execution time may be reduced. Future, directions include a domain decomposition preconditioner based on the physical partitioning of the domain.

5 Beyond Eigenvalue Computations

One of the biggest challenges in electronic structure calculations is in molecular dynamics. What is sought is the ground state corresponding to the minimum energy level, among all possible topological structures of the cluster.

The typical approach in this context uses simulated annealing [18]. The goal is to find a 'steady-state' distribution, meaning, for p particles, find locations $R_1, \ldots, R_p$ to minimize the total energy $E(R_1, R_2, \ldots, R_p)$ of the system. This is done by solving an artificial time-dependent problem of the form [7],

$$m \frac{dv}{dt} = F + \gamma v + G(t, T)$$

In the above equation, v is the velocity of the given particle, F is the force acting on it, and γ is some damping (viscosity) coefficient. The additional term G, a variable of time t and temperature T, is a random force that is added to the system essentially to prevent converging to a local minimum. The forces F are easily obtained from the energies by differentiation, and explicit formulas are available for this. Thus, the system is started from some initial configuration, and then time-stepped using the forces obtained from previous step and a simple randomization scheme. This is repeated until the energy stabilizes. An illustration is shown in Figure 5.1.

It is important to realize that the number of steps required to reach this ground state can be very large, potentially in the order of tens of thousands. Each of these steps itself requires a full-blown self-consistent calculation, i.e., a few iterations each requiring the calculation of possibly hundreds or thousands of eigenvalues. This constitutes a real computational challenge. The core of the calculation resides in the computation of a large number of eigenvectors which is the main source of difficulty. Indeed, most known numerical methods for large eigenvalue problems are efficient in computing a small number of the smallest (or largest) eigenvalues. When this number is an important fraction of the size

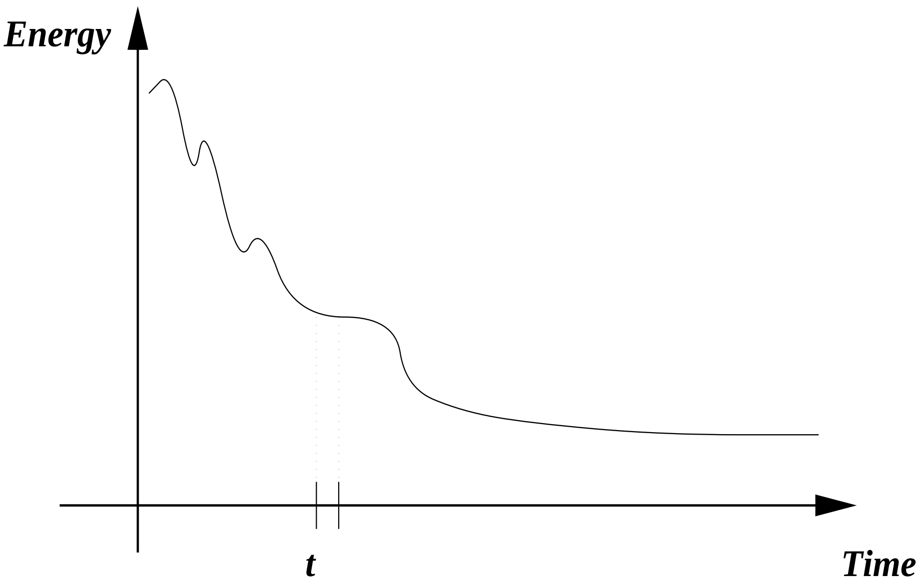

Figure 5.1: Time-stepping in a molecular dynamics simulation

n of the matrix, these methods become ineffective. It is important to develop new techniques which can deal more efficiently with interior eigenvalues.

However, to numerical analysts, the natural question seems to be whether eigenvalues can be avoided altogether. In other words: is it possible to reformulate the original problem in terms of a calculation which does not require the eigenvalues? Strictly speaking, eigenvectors and therefore eigenvalues are not really needed. The eigenvectors are only needed to compute the charge density via equation (2.3). Note that after discretization the function ψ becomes a vector of length N whose i-th component is the approximation of ψ at the mesh-point r_i. If we call V the matrix whose column vectors are the (normalized) eigenvectors ψ_i, $i = 1, \ldots, s$, for the s occupied states, then

$$P = VV^H \tag{5.1}$$

is a projector, and the charge density at a given point r_i in space is the i-th diagonal element of P. A number of techniques have been developed based on this observation [1, 21, 30]. In the following we will only sketch the main ideas for the purpose of illustrating possible new linear algebra problems of interest.

In the so-called Order-n methods, an approximation to the projector P in (5.1) is constructed without knowledge of eigenvectors. Denote by p_{ij} the coefficients of the matrix P. A number of properties are exploited for this purpose. First, are the following two relations:

$$\begin{aligned}\mathrm{tr}[P] &= \sum_i p_{ii} &&= \text{particle number}\\ \mathrm{tr}[PH] &= \sum_{i,j} p_{ij}H_{ji} &&= \text{system energy.}\end{aligned}$$

The first relation is a consequence of the fact that each p_{ii} represents the charge density at point r_i as was mentioned above. The second is a consequence of the fact that $PH = PHP$ is the restriction of H to the invariant subspace associated with eigenvalues corresponding to the occupied states. The trace of PH is therefore the sum of these eigenvalues, which is the total energy of the system.

Another important property that is exploited is the physical fact that P *decays away from the main diagonal*. Hence the idea to try to find a P whose trace is fixed and which minimizes the trace of PH. The trace constraint can be avoided by shifting H

$$\mathrm{tr}[P(H - \mu I)] = \mathrm{tr}[PH] - \mu N_e,$$

where N_e is the particle number.

The above optimization problem is not yet well posed since without constraints on P the minimum can be arbitrarily negative or small. The missing constraint is to force P to be a projector. As it turns out, this can be achieved by forcing its eigenvalues to be between zero and one [21]. The minimization will yield a matrix P which has eigenvalues equal to either one or zero, thus satisfying the desired idempotent constraint automatically.

One strategy that has been used in [21] for this purpose is to seek P in the form

$$P = 3S^2 - 2S^3$$

If the eigenvalues of S are in the range $[-0.5, 1.5]$ this transformation will map them into [0,1]. The procedure then is to seek a banded S that minimizes

$$\mathrm{tr}[(3S^2 - 2S^3)(H - \mu I)]$$

using descent type-algorithms. The gradient of the above function is computable.

The drawback of this approach is its lack of accuracy. The minimization is also not well understood. In addition, the band required for S may not be so small for certain systems. Nevertheless, it is interesting to note that explicit eigenvalue calculations are avoided. Some global approximation of the invariant subspace associated with these eigenvalues is extracted, meaning that a global approximation to the set of eigenfunctions is computed via the approximation to the projector P.

6 Conclusion

There are several challenges arising in the electronic structure problem. Among the current issues, are to find algorithms for computing large numbers of eigenvectors, specifically methods for preconditioning. The methods currently in use are expensive when the number of eigenvalues is very large. Beyond these issues, there is a potential to make orders of magnitude improvements in standard electronic calculations by combining three ingredients: (1) efficient matrix methods; (2) better solution to the underlying optimization problem; and (3) parallel processing.

REFERENCES

1. S. Baroni and P. Giannozzi, Towards very large-scale electronic-structure calculations, *Europhys. Lett.* 17, 547 (1992).
2. E. J. Bylaska, S. Kohn, S. Baden, A. Edelman, R. Kawai, M. E. Ong, and J. Weare, Scalable Parallel Numerical Methods and Software Tools for Material Design, *Proceedings of the Seventh SIAM Conference on Parallel Processing for Scientific Computing, San Francisco, CA,* 1995.
3. 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).
4. J.R. Chelikowsky and M. L. Cohen, Handbook on Semiconductors, edited by P. T. Landsberg, Vol. 1, p. 59, (Elsevier, Amsterdam, 1992).
5. 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).
6. 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).
7. 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).
8. M. Crouzeix, B. Philippe, and M. Sadkane, The Davidson method, *SIAM J. Sci. Comput.*, 15, 62–76 (1994).
9. J. Cullum and R. A. Willoughby, Lanczos algorithms for large symmetric eigenvalue computations 2: Programs, Progress in Scientific Computing, v. 4, (Birkhauser, Boston, 1985).
10. J. Daniel, W. B. Gragg, L. Kaufman, and G. W. Stewart, Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization, *Math. Comput.,* 30, 772 (1976).
11. 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).
12. D. E. Ellis and G. S. Painter, *Phys. Rev. B,* 2 2887 (1970).
13. B. Fornberg and D. M. Sloan. A review of pseudospectral methods for solving partial differential equations. *Acta Numerica*, 203–267 (1994).

14. A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek and V. Sunderam, PVM3 Users's Guide and Reference Manual, *ORNL,* TM-12187.
15. 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).
16. 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).
17. G. E. Kimball and G. H. Shortley, *Phys. Rev.,* 45, 815 (1934).
18. C. M. Kirkpatrick and D. S. Marynick. Localized molecular orbital studies of transition metal complexes. II. simple π-accepting ligands. *Journal of Computational Chemistry*, 6,142–147.
19. W. Kohn and S. Baden, The Parallelization of an Adaptive Multigrid Eigenvalue Solver with LPARX, *Proceedings of the Seventh SIAM Conference on Parallel Processing for Scientific Computing, San Francisco, CA,* 1995.
20. W. Kohn and L. J. Sham, Phys. Rev., 140, A,1133 (1965).
21. X.-P. Li, R. W. Nunes, and D. Vanderbilt, Density-matrix electronic-structure method with linear system-size scaling, *Phys. Rev. B* 47, 10891 (1993).
22. B. Liu, in: Numerical Algorithms in Chemistry: Algebraic Methods, eds. C. Moler and I. Shavitt, LBL-8158 Lawrence Berkeley Laboratory (1978).
23. 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).
24. B.N. Parlett and D.S. Scott, The Lanczos Algorithm with Selective Orthogonalization, *Math. Comp.*, 33, 217 (1979).
25. Y. Saad, Numerical Methods for Large Eigenvalue Problems (Manchester Univ. Press, Manchester, 1992).
26. Y. Saad and A. V. Malevsky, P-SPARSLIB: A Portable Library of Distributed Memory Sparse Iterative Solvers, *University of Minnesota Supercomputer Institute,* Research Report 95/180, September 1995.
27. Y. Saad and K. Wu. Design of an iterative solution module for a parallel matrix library (P_SPARSLIB). In W. Schonauer, editor, *Proceedings of IMACS conference, Georgia, 1994*, 1995. Was TR 94-59, Department of Computer Science, University of Minnesota.
28. A. Stathopoulos and C. F. Fischer, A Davidson program for finding a few selected extreme eigenpairs of a large, sparse, real, symmetric matrix, *Comput. Phys. Commun.*, 79, 268 (1994).
29. A. Stathopoulos and A. Ynnerman and C. F. Fischer, A PVM implementation of the MCHF atomic structure package, *The International Journal*

of Supercomputer Applications and High Performance Computing, Spring (1996).

30. A. P. Seitsonen, M. J. Puska, and R. M. Nieminen, Real-space electronic-structure calculations: Combination of the finite-difference and conjugate-gradient methods, *Phys. Rev. B* , 51, 14057 (1995).
31. 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).
32. N. R Troullier and J. L. Martins, Efficient pseudopotentials for plan-wave calculations. *Phys. Rev. B* 43,1993-1997 (1991).