Parallel Self-Consistent-Field Calculations via Chebyshev-Filtered Subspace Acceleration *

Yunkai Zhou † Yousef Saad † Murilo L. Tiago ‡ James R. Chelikowsky ‡

September 3, 2006

Abstract

Solving the Kohn-Sham eigenvalue problem constitutes the most computationally expensive part in self-consistent *density functional theory* (DFT) calculations. A nonlinear Chebyshev-filtered subspace iteration is developed which avoids computing explicit eigenvectors, except at the first SCF iteration. The method may be viewed as an approach to solve the original nonlinear Kohn-Sham equation by a nonlinear subspace iteration technique, without emphasizing the intermediate linearized Kohn-Sham eigenvalue problems. The method reaches self-consistency within a similar number of SCF iterations as eigensolver-based approaches. However, replacing the standard diagonalization at each SCF iteration by a Chebyshev subspace filtering step results in a significant speedup over methods based on standard dagonalization. Algorithmic details of a parallel implementation of this method are discussed. Numerical results are presented to show that the method enables to perform a class of highly challenging DFT calculations that were not feasible before.

Key words: Density functional theory, self-consistent-field, Chebyshev filtered subspace iteration, Chebyshev-Davidson, eigenproblem, real-space, pseudopotential.

1 Introduction

Electronic structure calculations based on first principles use a very successful combination of *density functional theory* (DFT) [15, 17] and *pseudopotential theory* [26, 27, 8, 23]. DFT reduces the original multi-electron Shrödinger equation into an effective one-electron Kohn-Sham equation, where all non-classical electronic interactions are replaced by a functional of the charge density. The pseudopotential theory further simplifies the problem by replacing the true atomic potential with an effective "pseudopotential" that is smoother but takes into account the effect of core electrons. Combining pseudopotential with DFT greatly reduces the number of one-electron wave-functions to be computed. However, even with these simplifications, solving the final Kohn-Sham equation can still be computationally challenging, especially when the systems being studied are complex or contain thousands of atoms.

The traditional approach for solving the Kohn-Sham equation utilizes planewave bases (e.g., [25, 18]) to expand the wave-functions for periodic structures. Real-space methods, which do not explicitly use a functional basis, have gained ground in recent years [9, 10, 33, 6] due in great part to their simplicity. Their first advantage, is that they are quite easy to implement in parallel. A second advantage relative

*Work supported by DOE under grants DE-FG02-03ER25585 and DE-FG02-03ER15491, by NSF grants ITR-0551195 and ITR-0428774, and by the Minnesota Supercomputing Institute.

†Department of Computer Science & Engineering, University of Minnesota, Minneapolis, MN 55455, USA. (zhou@msi.umn.edu, saad@cs.umn.edu).

‡Institute for Computational Engineering and Sciences, University Station, University of Texas at Austin, Austin, Texas 78712, USA. (mtiago@ices.utexas.edu, jrc@ices.utexas.edu).

to the planewave approach is that they do not require to use super-cells for non-periodic systems. Third, the application of potentials onto electron wave-functions is performed directly in real-space. Although the Hamiltonian matrices with a real-space approach is larger than with planewave, the Hamiltonians are highly sparse and never stored or computed explicitly. Only matrix-vector products that represent the application of the Hamiltonians on wave-functions need to be computed.

This paper focusses on effective techniques to handle the most computationally expensive part of DFT calculations, namely the self-consistent-field (SCF) iteration. We present details of a recently developed nonlinear Chebyshev-filtered subspace iteration (CheFSI) method, implemented in our own DFT package called PARSEC (Pseudopotential Algorithm for Real-Space Electronic Calculations).

The Standard SCF iteration framework is used in CheFSI, and a self-consistent solution is sought, which means that CheFSI has the same accuracy as other standard DFT approaches based on SCF iteration. One can view CheFSI as a technique to directly tackle the original nonlinear Kohn-Sham eigenvalue problems by a form of nonlinear subspace iterations, without emphasizing the intermediate linearized Kohn-Sham eigenvalue problems. In fact, within CheFSI, explicit eigenvectors are computed only at the first SCF iteration iteration, in order to provide a suitable initial subspace. After the first SCF step, the explicit computation of eigenvectors at each SCF iteration is replaced by a single subspace filtering step. The method reaches self-consistency within a number of SCF iterations that is close to that of eigenvector-based approaches. However, since eigenvectors are not explicitly computed after the first step, a significant gain in execution time results when compared with methods based on explicit diagonalization. Around tenfold or more speedup over well-known efficient eigenvalue packages such as ARPACK [22] and TRLan [40, 41] is normally observed. CheFSI enables us to perform a class of highly challenging DFT calculations, including clusters with over ten thousand atoms, which were not feasible to solve before. Numerical results are presented in Section 6.

The sequential version of CheFSI method is described in [45]. We also refer to [45] for a more complete literature survey. This paper begins with a summary of SCF for DFT calculations and the main features of the parallel paradigm of PARSEC, then turns to the description of the CheFSI method and its parallel implementation within the PARSEC code. Note that in [45] the first diagonalization is done by calling the thick-restart Lanczos (TRLan) method [40, 41], here we also discuss the block Chebyshev-Davidson [44, 43] method, which improves the efficiency for diagonalization at the first SCF iteration.

2 Eigenvalue problems in DFT SCF calculations

Within DFT, the multi-electron Schrödinger equation is simplified as the following Kohn-Sham equation:

$$\left[-\frac{\hbar^2}{2M}\nabla^2 + V_{total}(\rho(r), r)\right]\Psi_i(r) = E_i\Psi_i(r), \tag{1}$$

where $\Psi_i(r)$ is a wave function, E_i is a Kohn-Sham eigenvalue, $\hbar$ is the Planck constant, and M is the electron mass. In practice we use atomic units, thus $\hbar = M = 1$.

The *total potential* V_{total}, also referred to as the *effective potential*, includes three terms,

$$V_{total}(\rho(r), r) = V_{ion}(r) + V_H(\rho(r), r) + V_{XC}(\rho(r), r), \tag{2}$$

where V_{ion} is the ionic potential, V_H is the Hartree potential, and V_{XC} is the exchange-correlation potential.

The Hartree and exchange-correlation potentials depend on the *charge density* $\rho(r)$, which is defined as

$$\rho(r) = 2\sum_{i=1}^{n_{occ}} |\Psi_i(r)|^2. \tag{3}$$

Here n_{occ} is the number of occupied states, which is equal to half the number of valence electrons in the system. The factor of two comes from spin multiplicity. Equation (3) can be easily extended to situations where the highest occupied states have fractional occupancy or when there is an imbalance in the number of electrons for each spin component.

The most computationally expensive step of DFT is in solving the Kohn-Sham equation (1). Since V_{total} depends on the charge density $\rho(r)$, which in turn depends on the wavefunctions of (1), eqn. (1) can be viewed as a *nonlinear eigenvalue problem*. The SCF iteration is a general technique used to solve this nonlinear eigenvalue problem. It starts with an initial guess of the charge density, then obtains the initial V_{total} and solves (1) for $\Psi_i(r)$'s to update $\rho(r)$ and V_{total}. Then (1) is solved again for the new $\Psi_i(r)$'s and the process is iterated until V_{total} (also the wave functions) becomes stationary. The standard SCF process is described in Algorithm 2.1.

Algorithm 2.1 *Self-consistent-field iteration:*

---

1. *Provide initial guess for* $\rho(r)$*, get* $V_{total}(\rho(r), r)$.
2. *Solve for* $\Psi_i(r),\ i = 1, 2, ...,$ *from*
$$\left[-\frac{1}{2}\nabla^2 + V_{total}(\rho(r), r)\right]\Psi_i(r) = E_i\Psi_i(r). \tag{4}$$
3. *Compute the new charge density* $\rho(r) = 2\sum_{i=1}^{n_{occ}} |\Psi_i(r)|^2$.
4. *Solve for new Hartree potential* V_H *from* $\nabla^2 V_H(r) = -4\pi\rho(r)$.
5. *Update* V_{XC}*; get new* $\tilde{V}_{total}(\rho, r) = V_{ion}(r) + V_H(\rho, r) + V_{XC}(\rho, r)$ *with a potential-mixing step.*
6. *If* $\|\tilde{V}_{total} - V_{total}\| < tol$*, stop; Else,* $V_{total} \leftarrow \tilde{V}_{total}$*, goto step 2.*

The number of eigenvectors needed in *Step 2* of Algorithm 2.1 is just the number of occupied states. In practice a few more eigenvectors are usually computed. For complex systems, i.e., when the number of valence electrons is large, each of the linearized eigenvalue problems (4) can be computationally demanding. In addition to the large number of eigenvectors needed, the code must also cope with Hamiltonian matrices of very large size.

For this reason, it is the goal of any DFT code to lessen the burden of solving (4) in the SCF iteration. One possible avenue to achieve this is to use better diagonalization routines. However this approach is limited as most diagonalization software has now reached maturation. At the other extreme, one can attempt to avoid diagonalization altogether, and this leads to the body of work represented by linear-scaling or order-N methods (see e.g. [13]). This approach however has other limitations. In particular, the approximations involved rely heavily on some decay properties of the density matrix in certiain function bases. In particular, they will be difficult to implement in real-space discretizations. Our approach lies somewhere between these extremes. We take advantage of the fact that accurate eigenvectors are unnecessary at each SCF iteration, since Hamiltonians are only approximate in the intermediate SCF steps, and exploit the nonlinear nature of the problem. The main point of the new algorithm is that once we have a good starting point for the Hamiltonian, it suffices to filter each basis vector at each iteration. In the intermediate SCF steps, these vectors are no longer eigenvectors but together they represent a good basis of the desired inveriant subspace. This will be discussed in Section 4. The next section summarizes parallel implementation issues in PARSEC.

3 The parallel environment in PARSEC

PARSEC uses pseudopotential real-space implementation of DFT. The motivation and original ideas behind the method go back to the early 1990s, see [9, 10]. Within PARSEC, an uniform Cartesian grid in real-space is placed on the region of interest, and the Kohn-Sham equation is discretized by a high order finite-difference method [12] on this grid. Wavefunctions are expressed as functions of grid positions. Outside a specified sphere boundary that encloses the physical system, wavefunctions are set to zero for non-periodic systems. In addition to the advantages mentioned in the introduction, another advantage of the real-space approach is that periodic boundary conditions are also reasonably simple to implement [1].

The latest version of PARSEC is written in Fortran 90/95. PARSEC has now evolved into a mature, massively parallel package, which includes most of the functionality of comparable DFT codes [20]. The reader is referred to [31, 35] for details and the rationale of the parallel implementation. The following is a brief summary of the most important points.

The parallel mode of PARSEC uses the standard Message Passing Interface (MPI) library for communication. Parallelization is achieved by partitioning the physical domain which can have various shapes depending on boundary conditions and symmetry operations. Figure 1 illustrates four cube-shaped neighboring sub-domains. For a generic, confined system without symmetry, the physical domain is a sphere which contains all atoms plus some additional space (due to delocalization of electron charge). In recent years, PARSEC has been enhanced to take advantage of physical symmetry. If the system is invariant upon certain symmetry operations, the physical domain is replaced with an irreducible wedge constructed according to those operations. For example, if the system has mirror symmetry on the xy plane, the irreducible wedge covers only one hemisphere, either above or below the mirror plane. For periodic systems, the physical domain is the periodic cell, or an irreducible wedge of it if symmetry operations are present. In any circumstance, the physical domain is partitioned in compact regions, each assigned to one processor only. Good load balance is ensured by enforcing that the compact regions have approximately the same number of grid points.

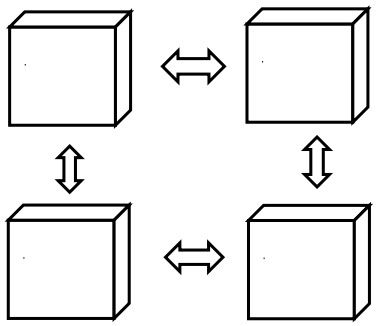

Figure 1: Sample decomposition of a physical domain in PARSEC.

Once the physical domain is partitioned, the physical problem is mapped onto the processors in a data-parallel way: each processor is in charge of a block of rows of the Hamiltonian corresponding to the block of grid points assigned to it. The eigenvector and potential vector arrays are row-wise distributed in the same fashion. The program only requires an index function $indx(i, j, k)$ which returns the number of the processor in which the grid point (i, j, k) resides.

Because the Hamiltonian matrix is never stored, we need an explicit reordering scheme which renumbers rows consecutively from one processor to the next one. For this purpose we use a list of pointers that gives for each processor, the row with which it starts.

Since finite difference discretizetion is used, when performing an operation such as a matrix-vector product, communication will be require between nearest neighbor processors. For communication we use two index arrays, one to count how many and which rows are needed from neighbors, the other to count the number of local rows needed by neighbors.

With this design of decomposition and mapping, the data required by the program can be completely distributed. Being able to distribute the memory requirement is quite important in solving large problems on standard supercomputers.

Parallelizing subspace methods for the linearized eigenvalue problems (represented as eqn. (4)) becomes quite straightforward with the above mentioned decomposition and mapping. Note that the subspace basis vectors contain approximations to eigenvectors, therefore the rows of the basis vectors are distributed in the same way as the rows of the Hamiltonian. All matrix-matrix products, matrix-vector products, and vector updates (e.g., linear combinations of vectors), can be executed in parallel.

Reduction operations, e.g., computing inner products and making the result available in each processor, are efficiently handled by the MPI reduction function `MPI_ALLREDUCE()`.

4 The nonlinear Chebyshev-filtered subspace iteration

As previously mentioned, the Hamiltonians of the intermediate SCF steps are approximate, therefore there is no need to compute eigenvectors of the intermediate Hamiltonians to high accuracy. Moreover, as observed in, e.g., [4, 36, 33, 13, 7, 45], the (discretized) charge density is the diagonal of the "functional" charge density matrix defined as $P = \Phi\Phi^T$, where the columns of the matrix Φ are discretized wavefunctions corresponding to occupied states. Notice that for any orthonormal matrix Q of a suitable dimension, $P = (\Phi Q)(\Phi Q)^T$. Therefore explicit eigenvectors are not needed to calculate the charge density. Any orthonormal basis of the eigensubspace corresponding to occupied states can give the desired intermediate charge density.

The proposed method combines the outer SCF iteration and the inner iteration required for diagonalization at each SCF step into one nonlinear subspace iteration. In this approach an initial subspace is progressively refined by a low degree Chebyshev polynomials filtering. This means that each basis vector u_i is processed as follows:

$$u_{i,new} := p_m(H)u_i$$

where p_m is some shifted and scaled Chebyshev poynomial whose goal is to enhance eigencomponents of u_i associated with the occupied states. Throughout the paper, the integer m denotes the degree of the polynomial p_m which is used for filtering.

If it were not for the nonlinear nature of the SCF loop, i.e., if H were a fixed operator, this approach would be equivalent to the well-known Chebyshev accelerated Subspace iteration proposed by Bauer [5], and later refined by Rutishauser [28, 29] (who later published an Algol routine in the "handbook for automatic computations: linear algebra", see [29]).

Chebyshev polynomial filtering has long been utilized in electronic structure calculations (see e.g. [32, 36, 14, 2, 3, 16]), focussing primarily on approximating the Fermi-Dirac operator where Chebyshev polynomials only over interval $[-1, 1]$ were considered. Therefore Chebyshev polynomials of rather high degree were necessary and additional techniques were required to suppress the Gibbs phenomena. In contrast, our approach exploits the well-known fast growth property outside the $[-1, 1]$ interval of the Chebyshev polynomial, so that only low degree Chebyshev polynomials are required to achieve sufficient filtering.

The main idea of CheFSI is to start with a good initial subspace V corresponding to occupied states of the initial Hamiltonian, this initial V is usually obtained by a diagonalization step. No diagonalizations are necessary after the first SCF step. Instead, the subspace from the previous iteration is filtered by a degree-m polynomial, $p_m(t)$, constructed for the current Hamiltonian H. The polynomial differs at each SCF step since H changes. Note that the goal of the filter is to make the subspace spanned by $p_m(H)V$

approximate the eigensubspace corresponding to the occupied states of the final H. At the intermediate SCF steps, the basis need not be an accurate eigenbasis since the intermediate Hamiltonians are not exact. The filtering is designed so that the resulting sequence of subspaces will progressively approximate the desired eigensubspace of the final Hamiltonian when self-consistency is reached. At each SCF step, only two parameters are required to construct an efficient Chebyshev filter, namely, a lower bound and an upper bound of the higher portion of the spectrum of the current Hamiltonian H in which we want $p_m(t)$ to be small. These bounds can be obtained with little additional cost, as will be seen in Section 4.2.

After self-consistency is reached, the Chebyshev filtered subspace includes the eigensubspace corresponding to occupied states. Explicit eigenvectors can be readily obtained by a *Rayleigh-Ritz refinement* [24] (also called *subspace rotation*) step.

4.1 Chebyshev-filtered subspace iteration

The main structure of CheFSI, which is given in Algorithm 4.1, is quite similar to that of the standard SCF iteration (Algorithm 2.1). One major difference is that the inner iteration for diagonalization at *Step 2* is now performed only at the first SCF step. Thereafter, diagonalization is replaced by a single Chebyshev subspace filtering step, performed by calling Algorithm 4.2.

Although the charge density (3) requires only the lowest n_{occ} states, the number of computed states, which is the integer s in Algorithm 4.1, is typically set to a value larger than n_{occ}, in order to avoid missing any occupied states. In practice we fix an integer n_{state} which is slightly larger than n_{occ}, and set $s = n_{state} + n_{add}$ with $n_{add} \leq 10$.

The parallel implementations of Algorithms 4.1 and 4.2 are quite straightforward with the parallel paradigm discussed in Section 3. We only mention that the matrix-vector products related to filtering, computing upper bounds, and Rayleigh-Ritz refinement, can easily execute in parallel. The re-orthogonalization at *Step 4* of Algorithm 4.2 uses a parallel version of the iterated Gram-Schmidt DGKS method [11], which scales better than the standard modified Gram-Schmidt algorithm.

Algorithm 4.1 *CheFSI for SCF calculation:*

---

1. *Start from an initial guess of* $\rho(r)$*, get* $V_{total}(\rho(r), r)$.
2. *Solve* $\left[-\frac{1}{2}\nabla^2 + V_{total}(\rho(r), r)\right] \Psi_i(r) = E_i \Psi_i(r)$ *for* $\Psi_i(r),\ i = 1, 2, ..., s$.
3. *Compute new charge density* $\rho(r) = 2\sum_{i=1}^{n_{occ}} |\Psi_i(r)|^2$.
4. *Solve for new Hartree potential* V_H *from* $\nabla^2 V_H(r) = -4\pi\rho(r)$.
5. *Update* V_{XC}*; get new* $\tilde{V}_{total}(\rho, r) = V_{ion}(r) + V_H(\rho, r) + V_{XC}(\rho, r)$ *with a potential-mixing step.*
6. *If* $\|\tilde{V}_{total} - V_{total}\| < tol$*, stop; Else,* $V_{total} \leftarrow \tilde{V}_{total}$ *(update* H *implicitly),*
   call the Chebyshev-filtered subspace method (Algorithm 4.2) to get s *approximate wavefunctions; goto step 3.*

The estimated complexity of the algorithm is similar to that of the sequential CheFSI method in [45]. For parallel computation it suffices to estimate the complexity on a single processor. Assume that p processors are used, i.e., each processor shares N/p rows of the full Hamiltonian. The estimated cost

Algorithm 4.2 *Chebyshev-filtered Subspace (CheFS) method:*

---

1. *Get the lower bound* b_{low} *and* a_0 *from previous Ritz values (use the largest one and the smallest one, respectively).*
2. *Compute the upper bound* b_{up} *of the spectrum of the current discretized Hamiltonian* H *(call Algorithm 4.4 in Section 4.2).*
3. *Perform Chebyshev filtering (call Algorithm 4.3 in Section 4.2) on the previous basis* Φ*, where* Φ *contains the discretized wavefunctions of* $\Psi_i(r),\ i = 1, ..., s$*:* $\Phi = \texttt{Chebyshev\_filter}(\Phi,\ m,\ b_{low},\ b_{up},\ a_0)$.
4. *Ortho-normalize the basis* Φ *by iterated Gram-Schmidt.*
5. *Perform the Rayleigh-Ritz step:*
   (a) *Compute* $\hat{H} = \Phi^T H \Phi$*;*
   (b) *Compute the eigendecomposition of* $\hat{H}$*:* $\hat{H}Q = QD$*, where* D *contains non-increasingly ordered eigenvalues of* $\hat{H}$*, and* Q *contains the corresponding eigenvectors;*
   (c) *'Rotate' the basis as* $\Phi := \Phi Q$*; return* Φ *and* D*.*

---

of Algorithm 4.2 on each processor with respect to the dimension of the Hamiltonian denoted by N, and the number of computed states s, is as follows:

- The Chebyshev filtering in *Step 3* costs $O(s * N/p)$ flops. The discretized Hamiltonian is sparse and each matrix-vector product on one processor costs $O(N/p)$ flops. *Step 3* requires $m * s$ matrix-vector products, at a total cost of $O(s * m * N/p)$ where the degree m of the polynomial is small (typically between 8 and 20).
- The ortho-normalization in *Step 4* costs $O(s^2 * N/p)$ flops. There are additional communication costs because of the global reductions.
- The eigen-decomposition at *Step 5* costs $O(s^3)$ flops.
- The final basis refinement step ($\Phi := \Phi Q$) costs $O(s^2 * N/p)$.

If a standard iterative diagonalization method is used to solve the linearized eigenproblem (4) at each SCF step, then it also requires (i) the orthonormalization of a (typically larger) basis; (ii) the eigen-decomposition of the projected Rayleigh-quotient matrix; and (iii) the basis refinement (rotation). These operations need to be performed several times within this single diagonalization. But Algorithm 4.2 preforms each of these operations only once per SCF step. Therefore, although Algorithm 4.2 scales in a similar way to standard diagonalization-based methods, the scaling constant is much smaller. For large problems, CheFS can achieve a tenfold or more speedup per SCF step, over using the well-know efficient eigenvalue packages such as ARPACK [22] and TRLan [40, 41]. The total speedup can be more significant since self-consistency requires several SCF iteration steps.

To summarize, a standard SCF method would have an outer SCF loop—the usual nonlinear SCF loop, and an inner diagonalization loop, which iterates until eigenvectors are within specified accuracy.

Algorithm 4.1 simplifies this by merging the inner-outer loops into a single outer loop, which can be considered as a *nonlinear subspace iteration algorithm*. The inner diagonalization loop is reduced into a single Chebyshev subspace filtering step.

4.2 Chebyshev filters and estimation of bounds

To construct Chebyshev polynomials to filter the subspace efficiently, it is necessary to find two parameters which allow to achieve a desired filtering. These correspond to upper and lower bounds for the undesired part of the spectrum and they can be obtained in a simple and effective way. We begin with a brief review of the well-known Chebyshev polynomials.

Chebyshev polynomials of the first kind ([24, p.371] [30, p.142]) are defined by

$$C_k(t) = \begin{cases} \cos(k\ \cos^{-1}(t)), & -1 \le t \le 1, \\ \cosh(k\ \cosh^{-1}(t)), & |t| > 1. \end{cases}$$

Note that $C_0(t) = 1, C_1(t) = t$. The following important 3-term recurrence is well-known

$$C_{k+1}(t) = 2t\ C_k(t) - C_{k-1}(t), \qquad t \in \mathbb{R}. \tag{5}$$

For the desired filtering, we want to exploit the rapid growth outside $[-1, 1]$ of the Chebyshev polynomial. This well-known property is discussed in [24], see also [44, 45]. Assume that the full spectrum of H (denoted as $\sigma(H)$) is contained in $[a_0, b]$. Then, in order to approximate the eigensubspace associated with the lower end of the spectrum, say $[a_0, a]$ with $a_0 < a < b$, it is necessary to map $[a, b]$ into $[-1, 1]$ before applying the Chebyshev polynomnial. This can be easily realized by an affine mapping defined as

$$\mathcal{L}(t) := \frac{t-c}{e}; \quad c = \frac{a+b}{2}, \quad e = \frac{b-a}{2}$$

where c denotes the center and e the half-width of the interval $[a, b]$.

The Chebyshev iteration utilizing the three-term recurrence (5) to dampen values on the interval $[a, b]$ is listed in Algorithm 4.3, see also [45]. The algorithm computes

$$Y = p_m(H)X \qquad \text{where} \qquad p_m(t) = C_m\left[\mathcal{L}(t)\right]. \tag{6}$$

This yields the iteration

$$X_{j+1} = \frac{2}{e}(H - cI)X_j - X_{j-1} \quad j = 1, 2, .., m-1.$$

with X_0 given and $X_1 = (H - cI)X_0$. As can be easily seen this is equivalent to a power iteration of the form

$$\begin{pmatrix} X_{j+1} \\ X_j \end{pmatrix} = \underbrace{\begin{pmatrix} \frac{2}{e}(H - cI) & -I \\ I & 0 \end{pmatrix}}_{\mathcal{B}} \begin{pmatrix} X_j \\ X_{j-1} \end{pmatrix}.$$

A little analysis would show that all the eigenvalues of the nonsymmetric matrix $\mathcal{B}$ are complex and of modulus one, except that those corresponding to eigenvalues of H that are less than a are mapped to real eigenvalues larger than one in magnitude. Therefore, just as for the standard power method, a scaling is required. The simplest strategy, discussed in [30] is to consider the scaled sequence

$$\tilde{X}_j = \frac{C_j[\frac{2}{e}(H - cI)]}{C_j[\frac{2}{e}(a_0 - cI)]} X_0$$

Thus, the scaling factor is $\rho_j = C_j[\frac{2}{e}(a_0 - cI)]$. Clearly this requires an estimate for a_0, but since this is used for scaling, only a rough value is needed. For the first SCF iteration, we can use the smallest

Ritz value of T from the same Lanczos run (Algorithm 4.4 below) as used to obtain the upper bound b for a_0. For the latter SCF steps, the smallest Ritz value from the previous SCF step can be used. Clearly, the vector sequence is not computed as shown above because ρ_j itself can be large and this would defeat the purpose of scaling. Instead, each $\tilde{X}_{j+1}$ is updated using the scaled vectors $\tilde{X}_j$ and $\tilde{X}_{j-1}$. The corresponding algorithm, discussed in [30] is shown in Algorithm 4.3 (the tildes and vector subscripts are omitted).

Algorithm 4.3 $[Y] =$ `Chebyshev_filter`(X, m, a, b, a_0).

Purpose: Filter column vectors of X by an m degree Chebyshev polynomial in H. Output in Y.

1. $e = (b-a)/2; \quad c = (b+a)/2;$
2. $\sigma = e/(a_0 - c); \qquad \sigma_1 = \sigma; \qquad \gamma = 2/\sigma_1.$
3. $Y = \frac{\sigma_1}{e}(HX - cX)$
4. *For* $i = 2 : m$
5. $\quad \sigma_2 = 1/(\gamma - \sigma);$
6. $\quad Y_{new} = \frac{2\sigma_2}{e}(HY - cY) - \sigma\sigma_2 X;$
7. $\quad X = Y;$
8. $\quad Y = Y_{new};$
9. $\quad \sigma = \sigma_2;$
10. *End For*

Algorithm 4.3 explicitly dampens the $[a, b]$ interval. The eigen-components associated with eigenvalues in $[a, b]$ will be transformed to small values while those to the left of $[a, b]$ will be around unity due to the properties of the Chebyshev polynomials. This is the desired filtering property when computing an approximation to the eigensubspace associated with the lower end of $\sigma(H)$. As seen in Algorithm 4.3, a desired filter can be easily controlled by adjusting two endpoints that bound the higher portion of $\sigma(H)$.

The wanted lower bound can be any value which is larger than the Fermi-level but smaller than the upper bound. It can also be a value slightly smaller than the Fermi-level; thanks to the monotonicity of the shifted and scaled Chebyshev polynomial on the spectrum of H, and the fact that we compute $s > n_{occ}$ number of Ritz values, the desired lowered end of the spectrum will still be magnified properly with this choice of lower bound.

Since the previous SCF iteration performs a Rayleigh-Ritz refinement step, it provides naturally an approximation for the lower bound a. Indeed, we can simply take the largest Rayleigh-quotient from the previous SCF iteration step as an approximation to the lower bound for the current Hamiltonian. In other words, a is taken to be the largest eigenvalue computed in step 5-(b) of Algorithm 4.2 from the previous SCF iteration, with no extra computation.

The upper bound for the spectrum (denoted by b) can be estimated by a k-step standard Lanczos method. As pointed out in [44], the higher endpoint b must be a bound for the full spectrum of H. This is because the Chebyshev polynomial also grows fast to the right of $[-1, 1]$. So if $[a, b]$ with $b < \sigma_{max}(H)$

is mapped into $[-1, 1]$, then the $[b, \sigma_{max}(H)]$ portion of the spectrum will also be magnified, which will cause the procedure to fail. Therefore, the bound b must be larger than $\sigma_{max}(H)$. On the other hand it should not be too large as this results in slow convergence. The simplest strategy which can be used for this is to use Gerschgorin's Circle Theorem. Bounds obtained this way can, however, overestimate $\sigma_{max}(H)$.

An inexpensive way to estimate an upper bound of $\sigma(H)$ by the standard Lanczos [21] method is described in Algorithm 4.4, to which a safeguard step is added. The largest eigenvalue λ of the tridiagonal matrix T is known to be below the largest eigenvalue λ of the Hamiltonian. If $\tilde{u}$ is the corresponding Ritz vector and $r = (H - \tilde{\lambda}I)\tilde{u}$ then there is an eigenvalue of H in the interval $[\tilde{\lambda} - \|r\|, \tilde{\lambda} + \|r\|]$ (see e.g. [24]). Algorithm 4.4 estimates λ_{max} by $max(\tilde{\lambda}) + \|f\|$, since it is known that $\|r\| \leq \|f\|$. This is not theoretically guaranteed to return an upper bound for λ_{max} - but it is generally observed to yield an effective upper bound. The algorithm for estimating b is presented in Algorithm 4.4 below. Note that the algorithm is easily parallelizable as it relies mostly on matrix-vector products. In practice, we found that $k = 4$ or 5 is sufficient to yield an effective upper bound of $\sigma(H)$. Larger k values (e.g., $k > 10$) are not necessary.

In the end we can see that the extra work associated with computing bounds for constructing the Chebyshev polynomials is negligible. The major cost of filtering is in the three-term recurrences in Algorithm 4.3, which involve matrix-vector products. The polynomial degree m is left as a free parameter. Our experience indicates that an m between 8 and 20 is good enough to achieve overall fast convergence in the SCF loop.

Algorithm 4.4 *Estimating an upper bound of $\sigma(H)$ by k-step Lanczos:*

---

1. *Generate a random vector v, set $v \leftarrow v/\|v\|_2$;*
2. *Compute $f = Hv$; $\alpha = f^T v$; $f \leftarrow f - \alpha v$; $T(1,1) = \alpha$;*
3. *Do $j = 2$ to $min(k, 10)$*
4. $\beta = \|f\|_2$;
5. $v_0 \leftarrow v$; $v \leftarrow f/\beta$;
6. $f = Hv$; $f \leftarrow f - \beta v_0$;
7. $\alpha = f^T v$; $f \leftarrow f - \alpha v$;
8. $T(j, j-1) = \beta$; $T(j-1, j) = \beta$; $T(j, j) = \alpha$;
9. *End Do*
10. *Return $\|T\|_2 + \|f\|_2$ as the upper bound.*

---

5 Diagonalization in the first SCF iteration

Within CheFSI, the most expensive SCF step is the first one, as it involves a diagonalization in order to compute a good initial subspace to be used for latter filtering. In principle, any effective eigenvalue algorithms can be used. PARSEC originally had three diagonalization methods: DIAGLA, which is

a preconditioned Davidson method [31, 35]; the symmetric eigensolver in ARPACK [34, 22]; and the Thick-Restart Lanczos algorithm called TRLan [40, 41]. For systems of moderate sizes, Diagla works well, and then becomes less competitive relative to ARPACK or TRLan for larger systems when a large number of eigenvalues are required. TRLan is about twice as fast as the symmetric eigensolver in ARPACK, because of its reduced need for re-orthogonalization. In [45], TRLan was used for the diagonalization at the first SCF step.

For very large systems, memory can become a severe constraint. One has to use eigenvalue algorithms with restart since out-of-core operations can be too slow. However, even with standard restart methods such as ARPACK and TRLan, the memory demand can still surpass the capacity of some supercomputers. For example, the $Si_{9041}H_{1860}$ cluster by TRLan or ARPACK would require more memory than the largest memory allowed for a job at the Minnesota Supercomputing Institute in 2006. Hence it is important to develop a diagonalization method that is less memory demanding but whose efficiency is comparable to ARPACK and TRLan. The Chebyshev-Davidson method [44, 43] is developed with these two goals in mind.

It is generally accepted that for the implicit filtering in ARPACK and TRLan to work efficiently, one needs to use a subspace with dimension about twice the number of wanted eigenvalues. This leads to a relatively large demand in memory when the number of wanted eigenvalues is large. The block Chebyshev-Davidson method discussed in [43] introduced an *inner-outer restart* technique. The *outer restart* corresponds to a standard restart in which the subspace is truncated to a smaller dimension when the specified maximum subspace dimension is reached. The *inner restart* corresponds to a standard restart restricted to an active subspace, it is performed when the active subspace dimension exceeds a given integer act_{max} which is much smaller than the specified maximum subspace dimension. With *inner-outer restart*, the subspace used in Chebyshev-Davidson is about half the dimension of the subspace required by ARPACK or TRLan.

We adapted the Chebyshev filters discussed in Section 4.2 into a Davidson-type eigenvalue algorithm. Although no Ritz values are available from previous SCF steps to be used as lower bounds, the Rayleigh-Ritz refinement step within a Davidson-type method can easily provide a suitable lower bound at each iteration. The upper bound can again be estimated by Algorithm 4.4, and it is computed only once. These two bounds are sufficient for constructing a filter at each Chebyshev-Davidson iteration. The constructed filter magnifies the wanted lower end of the spectrum and dampens the unwanted higher end, therefore the filtered block of vectors have strong components in the wanted eigensubspace, which results in an efficiency that is comparable to that of ARPACK or TRLan. The main structure of this Chebyshev-Davidson method is sketched in Algorithm 5.1, we refer interested readers to [43] for algorithmic details.

The first step diagonalization by the block Chebyshev-Davidson method, together with the Chebyshev-filtered subspace method (Algorithm 4.2), enabled us to perform SCF calculations for a class of large systems, including the silicon cluster $Si_{9041}H_{1860}$ for which over 19000 eigenvectors of a Hamiltonian with dimension around 3 million were to be computed. These systems are practically infeasible with the other three eigensolvers (ARPACK, TRLan and Diagla) in PARSEC, using the current supercomputer resources available to us at the Minnesota Supercomputing Institute (MSI).

6 Numerical Results

PARSEC has been applied to study a wide range of material systems (e.g. [1, 20, 10]). The focus of this section is on large systems where relatively few numerical results exist because of the infeasibility of eigenvector-based methods. We mention that [42] contains very interesting studies on clusters containing up to 1100 silicon atoms, using the well-known efficient plane-wave DFT package VASP [19, 18]; however, it is stated in [42] that a cluster with 1201 silicon atoms is "too computationally intensive". As a comparison, PARSEC using CheFSI, together with the currently developed symmetric operations of

Algorithm 5.1 *Structure outline of the block Chebyshev-Davidson method*

---

1. *Compute* b_{up} *using Algorithm 4.4,*
   set b_{low} = *median of the eigenvalues of* T *from Algorithm 4.4.*
   Make the given initial size-k block V_1 *orthonormal, set* $V = [V_1]$.
2. $[\ V_f\ ] =$ `Chebyshev_filter`$(V_1, m, b_{low}, b_{up})$.
3. *Augment the basis* V *by* V_f*:* $V \leftarrow [\ V,\ V_f\ ]$, *make* V *orthonormal.*
4. *Inner-restart if active subspace dimension exceeds a given integer* act_{max}.
5. *Rayleigh-Ritz refinement: update matrix* M *s.t.* $M = V^T H V$;
   do eigendecomposition of M*:* $M\ Y = Y D$; *updated basis* V*:* $V \leftarrow V\ Y$.
6. *Compute residual vectors, determine convergence;*
   perform deflation if some eigenpairs converge.
7. *If all wanted eigenpairs converged, stop; else, adapt* $b_{low} = max(diag(D))$,
   set $V_1 = [$ the first k non-converged Ritz vectors *in* $V\ ]$.
8. *Outer-restart if size of* V *exceeds maximum subspace dimension.*
9. *Continue from step 2.*

---

real-space pseudopotential methods [37], can now routinely solve silicon clusters with five thousand atoms.

The hardware used for the computations is the SGI Altix cluster at MSI, it consists 256 Intel Itanium processors at CPU rates of 1.6 GHz, sharing 512 GB of memory (but a single job is allowed to request at most 250 GB memory).

The goal of the computations is not to study the parallel scalability of PARSEC, but rather to use PARSEC to do SCF calculation for large systems that were not studied before. Therefore we do not use different processor numbers to solve the same problem. Scalability is studied in [35] for the preconditioned Davidson method, we mentioned that the scalability of CheFSI is better than eigenvector-based methods because of the reduced reorthogonalizations.

In the reported numerical results, the `total_eV/atom` is the total energy per atom in electron-volts, this value can be used to assess accuracy of the final result; the `#SCF` is the iteration steps needed to reach self-consistency; and the `#MVp` counts the number of matrix-vector products. Clearly `#MVp` is not the only factor that determines CPU time, the orthogonalization cost can also be a significant component.

For all of the reported results for CheFSI, the first step diagonalization used the Chebyshev-Davidson method (Algorithm 5.1). In Tables 2–8, the `1st CPU` denotes the CPU time spent on the first step diagonalization by Chebyshev-Davidson; the `total CPU` counts the total CPU time spent to reach self-consistency by CheFSI.

The first example (Table 1) is a relatively small silicon cluster $Si_{525}H_{276}$, which is used to compare the performance of CheFSI with two eigenvector-based methods. All methods use the same symmetry operations [37] in PARSEC.

For larger clusters $Si_{2713}H_{828}$ (Table 2) and $Si_{4001}H_{1012}$ (Table 3), Diagla became too slow to be

method	#MVp	#SCF steps	total_eV/atom	CPU(secs)
CheFSI	189755	11	-77.316873	542.43
TRLan	149418	10	-77.316873	2755.49
Diagla	493612	10	-77.316873	8751.24

Table 1: $Si_{525}H_{276}$, using 16 processors. The Hamiltonian dimension is 292584, where 1194 states need to be computed at each SCF step. The first step diagonalization by Chebyshev-Davidson cost 79755 #MVp and 221.05 CPU seconds; so the total #MVp spent on CheFS in CheFSI is 110000. The polynomial degree used is $m = 17$ for Chebyshev-Davidson and $m = 8$ for CheFS. The fist step diagonalization by TRLan requires 14909 #MVp and 265.75 CPU seconds.

practical. However, we could still apply TRLan for the first step diagonalization for comparison, but we did not iterate until self-consistency was reached since that would cost a significant amount of our CPU quota. Note that with the problem size increasing, Chebyshev-Davidson compares more favorably over TRLan. This is because we employed an additional trick in Chebyshev-Davidson, which corresponds to allowing the last few eigenvectors not to converge to the required accuracy. The number of the non fully converged eigenvectors is bounded above by act_{max}, which is the maximum dimension of the active subspace. Typically $30 \leq act_{max} \leq 300$ for Hamiltonian size over a million where several thousand eigenvectors are to be computed. The implementation of this trick is rather straightforward since it corresponds to applying the CheFS method to the subspace spanned by the last few vectors in the basis that have not converged to required accuracy.

dim. of H	n_{state}	#MVp	#SCF	total_eV/atom	1st CPU	total CPU
1074080	5843	1400187	14	-86.16790	7.83 hrs.	19.56 hrs.

Table 2: $Si_{2713}H_{828}$, using 16 processors. $m = 17$ for Chebyshev-Davidson; $m = 10$ for CheFS. (First step diagonalization by TRLan cost 8.65 hours, projecting it into a 14-steps SCF iteration cost around 121.1 hours.)

dim. of H	n_{state}	#MVp	#SCF	total_eV/atom	1st CPU	total CPU
1472440	8511	1652243	12	-89.12338	18.63 hrs.	38.17 hrs.

Table 3: $Si_{4001}H_{1012}$, using 16 processors. $m = 17$ for Chebyshev-Davidson; $m = 8$ for CheFS. (First step diagonalization by TRLan cost 34.99 hours, projecting it into a 12-steps SCF iteration cost around 419.88 hours.)

For even larger clusters $Si_{6047}H_{1308}$ (Table 4) and $Si_{9041}H_{1860}$ (Table 5), it became impractical to apply TRLan for the first step diagonalization because of too large memory requirements. For these large systems, using an eigenvector-based method for each SCF step is clearly not feasible. We note that the cost for the first step diagonalization by Chebyshev-Davidson is still rather high, it took close to 50% of the total CPU. In comparison, the CheFS method (Algorithm 4.2) saves a significant amount of CPU for SCF calculations over diagonalization-based methods, even if very efficient eigenvalue algorithms are used.

Once the DFT problem, Eq. (1), is solved, we have access to several physical quantities. One of them is the ionization potential (IP) of the nanocrystal, defined as the energy required to remove one electron from the system. Numerically, we use a ΔSCF method: perform two separate calculations,

dim. of H	n_{state}	#MVp	#SCF	total_eV/atom	1st CPU	total CPU
2144432	12751	2682749	14	-91.34809	45.11 hrs.	101.02 hrs.

Table 4: $Si_{6047}H_{1308}$, using 32 processors. $m = 17$ for Chebyshev-Davidson; $m = 8$ for CheFS.

dim. of H	n_{state}	#MVp	#SCF	total_eV/atom	1st CPU	total CPU
2992832	19015	4804488	18	-92.00412	102.12 hrs.	294.36 hrs

Table 5: $Si_{9041}H_{1860}$, using 48 processors. $m = 17$ for Chebyshev-Davidson; $m = 8$ for CheFS.

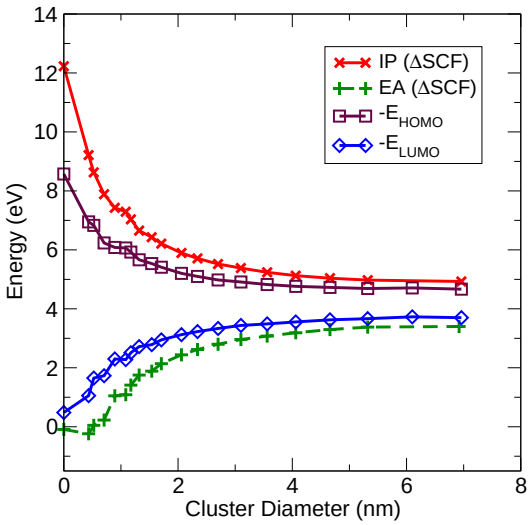

Figure 2: Ionization potential (IP, crosses) and electron affinity (EA, "plus" signs) for various clusters with diameters ranging from 0 nm (SiH_4) to 7 nm ($Si_{9041}H_{1860}$). Squares denote the negative of the highest occupied eigenvalue energy (E_{HOMO}) of the neutral cluster. Diamonds denote the negative of the lowest unoccupied eigenvalue energy (E_{LUMO}).

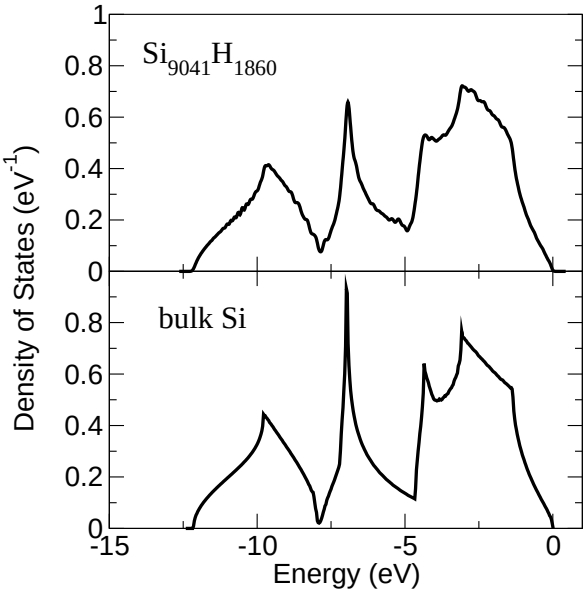

Figure 3: Density of states (DOS) of the cluster $Si_{9041}H_{1860}$ (upper panel) compared with periodic crystalline silicon (lower panel). As a consequence of the large size, the DOS of the $Si_{9041}H_{1860}$ cluster is very close to that of bulk silicon (the infinite-size limit).

one for the neutral cluster and another for the ionized one, and observe the variation in total energy between these calculations. Figure 2 shows the IP of several clusters, ranging from the smallest possible (SiH_4) to $Si_{9041}H_{1860}$. For comparison, we also show the eigenvalue of the highest occupied Kohn-Sham orbital (E_{HOMO}). A known fact of DFT-LDA is that the minus E_{HOMO} energy is lower than the IP in clusters [23], which is confirmed in Figure 2. In addition, the figure shows that the IP and $-E_{HOMO}$ approach each other in the limit of extremely large clusters.

Figure 2 also shows the electron affinity (EA) of the various clusters. The EA is defined as the energy released by the system when one electron is added to it. Again, we calculate it by performing SCF calculations for the neutral and the ionized systems (negatively charged instead of positively charged now). In PARSEC, this sequence of SCF calculations can be done very easily by reusing previous information: The initial diagonalization in the second SCF calculation is waived if we reuse eigenvectors and eigenvalues from a previous calculation as initial guesses for the ChebFSI method. Figure 2 shows that, as the cluster grows in size, the EA approaches the negative of the lowest-unoccupied eigenvalue energy. This implies that the ΔSCF method, if applied to bulk silicon, would predict energies for removal/addition of electrons equal to the negative of the Kohn-Sham eigenvalues at the valence/conduction bands edges respectively.

The properties of large silicon clusters are expected to be similar to the ones of bulk silicon, which is equivalent to a nanocrystal of "infinite size". Figure 3 shows that the density of states already assumes a bulk-like profile in clusters with around ten thousand atoms. The presence of hydrogen atoms on the surface is responsible for subtle features in the DOS at around -8 eV and -3 eV. Because of the discreteness of eigenvalues in clusters, the DOS is calculated by adding up normalized Gaussian distributions located at each calculated energy eigenvalue. In Figure 3, we used Gaussian functions with dispersion of 0.05 eV. More details are discussed in [39].

We also applied PARSEC to some large iron clusters. Tables 6–8 contain three clusters with more than 300 iron atoms. The number of states, n_{state}, is multiplied by two because spin effect is considered. These metallic systems are well-known to be very difficult for DFT calculations, because of the "charge sloshing" [25, 18]. The LDA approximation used to get exchange-correlation potential V_{XC} is also known not to work well for iron atoms. However, PARSEC was able to reach self-consistency for these large metallic clusters within reasonable time length. Physical significance of the computed data will be discussed in [38]. It took more than 100 SCF steps to reach self-consistency, which is generally considered too high for SCF calculations, but we observed (from calculations performed on smaller iron clusters) that eigenvector-based methods also required a similar number of SCF steps to converge, thus the slow convergence is associated with the difficulty of DFT for metallic systems. Without CheFS, and under the same hardware conditions as listed in Tables 6–8, over 100 SCF steps using eigenvector-based methods would have required months to complete for each of these clusters.

H size	n_{state}	#MVp	#SCF	total_eV/atom	1st CPU	total CPU
2790688	1812×2	9377435	110	-795.18064	16.16 hrs.	112.44 hrs.

Table 6: Fe_{302}, using 16 processors. $m = 20$ for Chebyshev-Davidson; $m = 19$ for CheFS.

H size	n_{state}	#MVp	#SCF	total_eV/atom	1st CPU	total CPU
2985992	1956×2	10241385	119	-795.19898	11.62 hrs.	93.15 hrs.

Table 7: Fe_{326}, using 24 processors. $m = 20$ for Chebyshev-Davidson; $m = 19$ for CheFS.

H size	n_{state}	#MVp	#SCF	total_eV/atom	1st CPU	total CPU
3262312	2160×2	12989799	146	-795.22329	16.55 hrs.	140.68 hrs.

Table 8: Fe_{360}, using 24 processors. $m = 20$ for Chebyshev-Davidson; $m = 17$ for CheFS.

7 Concluding Remarks

We developed and implemented the parallel CheFSI method for DFT SCF calculations. Within CheFSI, only the first SCF step requires a true diagonalization, and we perform this step by the block Chebyshev-Davidson method. No diagonalization is required after the first step; instead, Chebyshev filters are adaptively constructed to filter the subspace from previous SCF steps so that the filtered subspace progressively approximates the eigensubspace corresponding to occupied states of the final Hamiltonian. The method can be viewed as a nonlinear subspace iteration method which combines the SCF iteration and diagonalization, with the diagonalization simplified into a single step Chebyshev subspace filtering.

CheFSI significantly accelerates the SCF calculations, and this enabled us to perform a class of large DFT calculations that were not feasible before by eigenvector-based methods. As an example of physical applications, we discuss the energetics of silicon clusters containing up to several thousand atoms.

Acknowledgments

We thank the staff members at the Minnesota Supercomputing Institute, especially Gabe Turner, for the technical support. There were several occasions where our large jobs required that the technical support staff change certain default system settings to suit our needs. The calculations would not have been possible without the computer resource and the excellent technical support at MSI.

References

[1] M. M. G. Alemany, M. Jain, L. Kronik, and J. R. Chelikowsky. Real-space pseudopotential method for computing the electronic properties of periodic systems. *Phys. Rev. B*, 69:075101–1–6, 2004.

[2] R. Baer and M. Head-Gordon. Chebyshev expansion methods for electronic structure calculations on large molecular systems. *J. Chem. Phys.*, 107:10003–10013, 1997.

[3] R. Baer and M. Head-Gordon. Electronic structure of large systems: Coping with small gaps using the energy renormalization group method. *J. Chem. Phys.*, 109:10159–10168, 1998.

[4] S. Baroni and P. Giannozzi. Towards very large scale electronic structure calculations. *Europhys. Lett.*, 17:547–552, 1992.

[5] F. L. Bauer. Das verfahren der treppeniteration und verwandte verfahren zur losung algebraischer eigenwertprobleme. *Z. Angew. Math. Phys.*, 8:214–235, 1957.

[6] T. L. Beck. Real-space mesh techniques in density-functional theory. *Rev. Mod. Phys.*, 72(4):1041–1080, 2000.

[7] C. Bekas, Y. Saad, M. L. Tiago, and J. R. Chelikowsky. Computing charge densities with partially reorthogonalized Lanczos. *Comp. Phys. Comm.*, 171:175–186, 2005.

[8] J. R. Chelikowsky and M. L. Cohen. Ab initio pseudopotentials for semiconductors. In *Handbook on Semiconductors*, volume 1, page 59. Elsevier, Amsterdam, 1992.

[9] J. R. Chelikowsky, N. Troullier, and Y. Saad. Finite-difference-pseudopotential method: Electronic structure calculations without a basis. *Phys. Rev. Lett.*, 72:1240–1243, 1994.

[10] J. R. Chelikowsky, N. Troullier, K. Wu, and Y. Saad. Higher-order finite-difference pseudopotential method: An application to diatomic molecules. *Phys. Rev. B*, 50:11355–11364, 1994.

[11] J. Daniel, W. B. Gragg, L. Kaufman, and G. W. Stewart. Reorthogonalization and stable algorithms for updating the Gram–Schmidt QR factorization. *Math. Comp.*, 30:772–795, 1976.

[12] B. Fornberg and D. M. Sloan. A review of pseudospectral methods for solving partial differential equations. In A. Iserles, editor, *Acta Numerica*, number 3, pages 203–268. Cambridge Univ. Press, 1994.

[13] S. Goedecker. Linear scaling electronic structure methods. *Rev. Mod. Phys.*, 71:1085–1123, 1999.

[14] S. Goedecker and L. Colombo. Efficient linear scaling algorithm for tight-binding molecular dynamics. *Phys. Rev. Lett.*, 73:122–125, 1994.

[15] P. Hohenberg and W. Kohn. Inhomogeneous electron gas. *Phys. Rev.*, 136:B864–B871, 1964.

[16] L. O. Jay, H. Kim, Y. Saad, and J. R. Chelikowsky. Electronic structure calculations using plane wave codes without diagonlization. *Comput. Phys. Comm.*, 118:21–30, 1999.

[17] W. Kohn and L. J. Sham. Self-consistent equations including exchange and correlation effects. *Phys. Rev.*, 140:A1133–A1138, 1965.

[18] G. Kresse and J. Furthmüller. Efficient iterative schemes for *ab initio* total-energy calculations using a plane-wave basis set. *Phys. Rev. B*, 54(16):11169–11186, 1996.

[19] G. Kresse and J. Hafner. Norm-conserving and ultrasoft pseudopotentials for first-row and transition elements. *J. Phys.: Condens. Matter*, 6(40):8245–8257, 1994.

[20] L. Kronik, A. Makmal, M. Tiago, M. Alemany, M. Jain, X. Huang, Y. Saad, and J. Chelikowsky. PARSEC – the pseudopotential algorithm for real-space electronic structure calculations: recent advances and novel applications to nano-structures. Technical report, (In progress).

[21] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Research Nat. Bur. Standards*, 45:255–282, 1950.

[22] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK USERS GUIDE: Solution of Large Scale Eigenvalue Problems by Implicitly Restarted Arnoldi Methods*. SIAM, Philadelphia, 1998. Available at `http//www.caam.rice.edu/software/ARPACK/`.

[23] R. M. Martin. *Electronic structure : Basic theory and practical methods*. Cambridge University Press, 2004.

[24] B. N. Parlett. *The Symmetric Eigenvalue Problem*. Number 20 in Classics in Applied Mathematics. SIAM, Philadelphia, PA, 1998.

[25] M. C. Payne, M. P. Teter, D. C. Allan, T. A. Arias, and J. D. Joannopoulos. Iterative minimization techniques for ab initio total-energy calculations: molecular dynamics and conjugate gradients. *Rev. Mod. Phys.*, 64:1045–1097, 1992.

[26] J. C. Phillips. Energy-band interpolation scheme based on a pseudopotential. *Phys. Rev.*, 112:685–695, 1958.

[27] J. C. Phillips and L. Kleinman. New method for calculating wave functions in crystals and molecules. *Phys. Rev.*, 116:287–294, 1959.

[28] H. Rutishauser. Computational aspects of F. L. Bauer's simultaneous iteration method. *Numer. Math.*, 13:4–13, 1969.

[29] H. Rutishauser. Simultaneous iteration method for symmetric matrices. In J. H. Wilkinson and C. Reinsh, editors, *Handbook for Automatic Computation (Linear Algebra)*, volume II, pages 284–302. Springer-Verlag, 1971.

[30] Y. Saad. *Numerical Methods for Large Eigenvalue Problems*. John Wiley, New York, 1992. Available at `http://www.cs.umn.edu/~saad/books.html`.

[31] Y. Saad, A. Stathopoulos, J. Chelikowsky, K. Wu, and S. Öğüt. Solution of large eigenvalue problems in electronic structure calculations. *BIT*, 36(3):563–578, 1996.

[32] O. F. Sankey, D. A. Drabold, and A. Gibson. Projected random vectors and the recursion method in the electronic-structure problem. *Phys. Rev. B*, 50(3):1376–1381, 1994.

[33] 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(20):14057–14061, 1995.

[34] D. C. Sorensen. Implicit application of polynomial filters in a k-step Arnoldi method. *SIAM J. Matrix Anal. Appl.*, 13:357–385, 1992.

[35] A. Stathopoulos, S. Öğüt, Y. Saad, J.R. Chelikowsky, and H. Kim. Parallel methods and tools for predicting materials properties. *IEEE Computing in Science and Engineering*, 2:19–32, 2000.

[36] U. Stephan, D. A. Drabold, and R. M. Martin. Improved accuracy and acceleration of variational order-N electronic structure computations by projection techniques. *Phys. Rev. B*, 58(20):13472–13481, 1998.

[37] M. L. Tiago and J. R. Chelikowsky. Technical report, Univ. Texas, Austin, (in preparation).

[38] M. L. Tiago, Y. Zhou, M. Alemany, Y. Saad, and J. R. Chelikowsky. The evolution of magnetism in iron from the atom to the bulk. Technical report, Univ. of Texas at Austin and Univ. of Minnesota, 2006 (submitted).

[39] M. L. Tiago, Y. Zhou, Y. Saad, and J. R. Chelikowsky. Electronic properties and energetics of nanometer-size silicon nanocrystals. Technical report.

[40] K. Wu, A. Canning, H. D. Simon, and L.-W. Wang. Thick-restart Lanczos method for electronic structure calculations. *J. Comput. Phys.*, 154:156–173, 1999.

[41] K. Wu and H. Simon. Thick-restart Lanczos method for large symmetric eigenvalue problems. *SIAM J. Matrix Anal. Appl.*, 22:602–616, 2000.

[42] Y. Zhao, M.-H. Du Y.-H. Kim, and S.B. Zhang. First-principles prediction of icosahedral quantum dots for tetravalent semiconductors. *Phys. Rev. Lett.*, 93(1):015502–1–4, 2004.

[43] Y. Zhou. Block-wise polynomial filtered Davidson-type subspace iteration. Technical report, Minnesota Supercomputing Institute, Univ. of Minnesota, (in revision).

[44] Y. Zhou and Y. Saad. A Chebyshev-Davidson algorithm for large symmetric eigenvalue problems. Technical report, Minnesota Supercomputing Institute, Univ. of Minnesota, (submitted).

[45] Y. Zhou, Y. Saad, M. L. Tiago, and J. R. Chelikowsky. Self-consistent-field calculation using Chebyshev polynomial filtered subspace iteration. *J. Comput. Phys.*, (to appear).