9 SCIENTIFIC HIGHLIGHT OF THE MONTH

Using Chebyshev-Filtered Subspace Iteration Methods to Solve the Kohn-Sham Problem

James R. Chelikowsky[1] and Yousef Saad[2]

[1] Center for Computational Materials, Institute for Computational Engineering and Sciences
Departments of Physics and Chemical Engineering
University of Texas, Austin, TX 78712, USA

[1]Department of Computer Science and Engineering
University of Minnesota, Minneapolis, MN 55455, USA

Abstract

The ground state electronic properties of a material can be obtained using density functional theory as embodied by the Kohn-Sham equation. Typically, one employs eigensolver-based approaches to solve this equation. These approaches can be computationally demanding and have largely limited the applicability of the Kohn-Sham framework to systems of no more than a few hundred atoms. Here we discuss a different approach based on a nonlinear Chebyshev-filtered subspace iteration, which avoids computing explicit eigenvectors except to initiate the process. Our method centers on solving the original nonlinear Kohn-Sham equation by a nonlinear form of the subspace iteration technique, without emphasizing the intermediate linearized Kohn-Sham eigenvalue problems. The method achieves self-consistency within a similar number of self-consistent field iterations as eigensolver-based approaches. *However, replacing the standard diagonalization at each self-consistent iteration by a Chebyshev subspace filtering step results in a significant speedup over methods based on standard dagonalization, often by more than an order of magnitude.* Algorithmic details of a parallel implementation of this method are discussed. Numerical results are presented to show that the method enables one to perform a class of highly challenging applications that heretofore were not feasible.

1 Introduction

Electronic structure calculations based on first principles use often employ a very successful combination of *density functional theory* (DFT) [1, 2] and *pseudopotential theory* [3–6]. DFT reduces the original multi-electron Schrödinger equation into an effective one-electron Kohn-Sham equation, where the non-classical electronic interactions are replaced by a functional of the charge density. Pseudopotential theory further simplifies the problem by replacing the "all electron" 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, but more importantly the energy and length scales are set solely by the valence states. As such, species such as a carbon and lead can be treated on equal footing. However, even with these simplifications, solving the Kohn-Sham equation remains computationally challenging when the systems of interest contain a large number, *e.g.*, more than a few hundred, atoms.

Several approaches have been advocated for solving the Kohn-Sham equations. They can be classified in two major groups: basis-free or basis-dependent approaches, according to whether they use an explicit basis set for electronic orbitals or not. Among the basis-dependent approaches, plane wave methods are frequently used in applications of DFT to periodic systems where plane waves can easily accommodate the boundary conditions [7, 8]. In contrast, localized basis sets such as Gaussian orbitals are very popular in quantum-chemistry applications [6, 9]. Special basis sets have also been designed for all-electron DFT calculations, which do not make use of pseudopotentials. These basis sets include: linearized augmented plane waves, muffin-tin orbitals, projector-augmented waves. A survey of advantages and disadvantages of these explicit-basis methods can be found in [6, 10].

Here we will focus on a different approach based on *real space* methods, which are "basis free." Real space methods have gained ground in recent years [11–14] owing in great part to their simplicity and ease of implementation. In particular, these methods are readily implemented in parallel environments. A second advantage is that, in contrast with a plane wave approach, real space methods do not impose artificial periodicity in non-periodic systems. In contrast, plane wave basis techniques can be applied to clusters (or molecules) by placing the system of interest in a large supercell. Provided the supercell is sufficiently large so that the cluster of interest is removed from neighboring replicants, the electronic structure solution will correspond to that of the isolated cluster. However, the potentials from neighboring cells can be an issue. This makes supercell solutions converge slowly with the size of the cell [15]. A related, and perhaps more significant issue, is that supercells complicate the handling of systems that are not electronically neutral. Charged systems can be handled within plane wave methods by including a compensating uniform charge [15]. Real space methods need not address such complications. A third advantage is that the application of the Hamiltonian to electron wave-functions is performed directly in real-space. Although the Hamiltonian matrix in real space methods is typically much larger than with plane waves, 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.

As in plane wave methods, the chief impediment to solving the Kohn-Sham problem is "diagonalizing" the Hamiltonian and obtaining a self-consistent field (SCF) solution. We present examples of a recently developed nonlinear Chebyshev-filtered subspace iteration (CheFSI) method, implemented in our own DFT solution package called PARSEC (Pseudopotential Algorithm for Real-Space Electronic Calculations) [11, 12]. Although described in the framework of real-space DFT, CheFSI can be employed to other SCF iterations. The subspace filtering method takes advantage of the fact that intermediate SCF iterations do not require accurate eigenvalues and eigenvectors of the Kohn-Sham equation.

The "standard" SCF iteration framework is used in CheFSI, and a self-consistent solution is obtained as with previous work, which means that CheFSI has the *same* accuracy as other standard DFT approaches. Unlike, some so-called "order-N" methods [16,17] CheFSI is equally applicable to metals and insulators. One can view CheFSI as a technique to tackle directly the original nonlinear Kohn-Sham eigenvalue problems by a form of nonlinear subspace iteration, without emphasizing the intermediate linearized Kohn-Sham eigenvalue problems. In fact, within CheFSI, explicit eigenvectors are computed only at the first SCF 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.

When compared with calculations based on efficient eigenvalue packages such as ARPACK [18] and TRLan [19,20] an order of magnitude speed-up is usually observed.

CheFSI enabled us to perform a class of highly challenging DFT calculations, including clusters with over ten thousand atoms, which were not feasible before without invoking additional approximations in the Kohn-Sham problem [21–24].

2 Eigenvalue problems in density functional calculations

The Kohn-Sham equation as defined in density functional theory is given by

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

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. (We will often use atomic units: $\hbar = m = e = 1$ in the following discussion.)

The *total potential* V_{total}, is the sum of 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, if the system is non-magnetic. Eq. (3) can be easily generalized 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 Eq. (1). Since V_{total} depends on the charge density $\rho(r)$, which in turn depends on the wave functions Ψ_i, Eq. (1), can be viewed as a *nonlinear eigenvalue problem*. The SCF iteration is a general technique used to solve this nonlinear eigenvalue problem. The iteration process begins with an initial guess of the charge density usually constructed from a superposition of free atomic charge densities, then obtains the initial V_{total} and solves Eq. (1) for $\Psi_i(r)$'s to update $\rho(r)$ and V_{total}. Then the Kohn-Sham (Eq. (1)) is solved again for the new $\Psi_i(r)$'s and the process is iterated until V_{total} (and also the wave functions) becomes stationary. The standard SCF process is described in Algorithm 2.1 and illustrated in Fig. 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. *Obtain new Hartree potential V_H by solving: $\nabla^2 V_H(r) = -4\pi\rho(r)$.*
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 can be computationally demanding. This is compounded by the fact that Hamiltonian matrices can be of very large size.

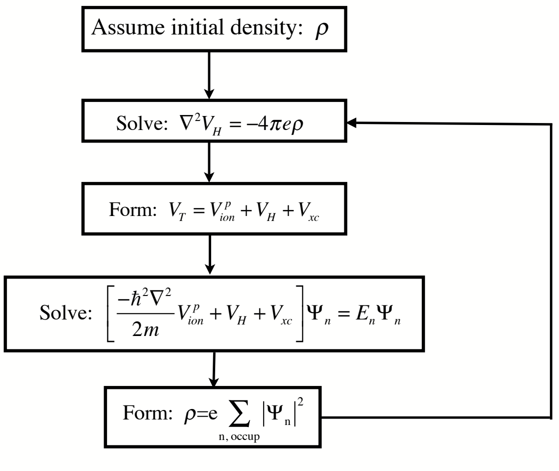
Assume initial density: ρ
Solve: $\nabla^2 V_H = -4\pi e\rho$
Form: $V_T = V_{ion}^p + V_H + V_{xc}$
Solve: $\left[\frac{-\hbar^2\nabla^2}{2m} V_{ion}^p + V_H + V_{xc}\right]\Psi_n = E_n\Psi_n$
Form: $\rho = e\sum_{n, occup} |\Psi_n|^2$

Figure 1: Flow diagram for obtaining a self-consistent solution of the Kohn-Sham equation.

For this reason, one hopes to lessen the burden of solving Eq. 4 in the SCF iteration. There are several options here. One could use some physical arguments to reduce the matrix size or zero some existing elements. Or, one could attempt to avoid diagonalization altogether, as is done in work represented by linear-scaling or order-N methods (see e.g. [16,17]). 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 can be difficult to implement in real-space discretizations or for systems where the decay properties are not optimal, *e.g.*, in metals. Another option is to use better (faster) diagonalization routines. However, this approach is limited as most diagonalization software is quite mature.

Our approach *avoids standard diagonalizations*, but otherwise makes no new approximations to the Hamiltonian. 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 invariant subspace.

3 Numerical methods for parallel platforms

The motivation and original ideas behind our real space method (PARSEC) go back to the early 1990s, see [11, 12]. 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 [25] on this grid. Wave functions are expressed as values on grid positions. Outside a specified sphere boundary that encloses the physical system, wave functions 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 [26].

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 [27]. The reader is referred to [28, 29] for details and the rationale of the parallel implementation. The PARSEC software can be obtained from

http://parsec.ices.utexas.edu/

The following is a brief summary of the most important points. PARSEC allows for either parallel or sequential excecutions. When run in the parallel mode, 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. Fig. 2 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 (owing 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

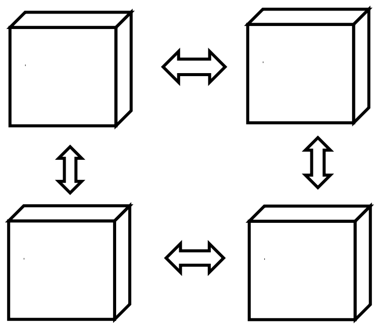

Figure 2: Sample decomposition of a physical domain used in the PARSEC package.

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.

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 required 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 decomposition and mapping, the data required by the program is completely distributed. In other words, the code runs in the so-called "Single Program Multiple Data" approach. For large problems it is quite important to be able to distribute memory loads among processors on high performance computers. For example, certain large jobs can simply not be run on a small number of processors on good-size distributed memory machines.

Parallelizing subspace methods for the linearized eigenvalue problems (represented as Eq. 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. In this way, all vector updates (e.g., linear combinations of vectors), can be executed locally (i.e., without communication). Matrix-vector products, and matrix-matrix products, can be easily executed in parallel but may require some communication with a few neighbors. 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

Since the Hamiltonians of the intermediate SCF steps are approximate, there is no need to compute eigenvectors of the intermediate Hamiltonians to a high accuracy. Moreover, as observed in Refs. [13, 17, 22, 30–32], 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 wave functions 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 article 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 [33], and later refined by Rutishauser [34, 35][4].

Chebyshev polynomial filtering has long been utilized in electronic structure calculations (see e.g. [30, 36–40]), focussing primarily on approximating the Fermi-Dirac operator.

Chebyshev polynomials of rather high degree were necessary and additional techniques were required to suppress the Gibbs phenomena. In contrast, the polynomials used in our approach are of relatively low degree (say < 20). They exploit the fast growth property of Chebyshev polynomials outside the interval $[-1, 1]$ to filter out undesired eigencomponents.

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

[4]Rutishauser published an Algol routine called *ritzit* in the volume: "Handbook for automatic computations: linear algebra", see [35]. This volume was largely at the origin of the EISPACK package (which later became a part of LAPACK), but Rutishauser's *ritzit* Algol routine was not translated into EISPACK.

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* [41] (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 (Eq. (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 [42], which scales better than the standard modified Gram-Schmidt algorithm. This process is illustrated in Fig. 3.

The estimated complexity of the algorithm is similar to that of the sequential CheFSI method in [22]. 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 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 (Eq. 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 performs 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 [18] and TRLan [19, 20].

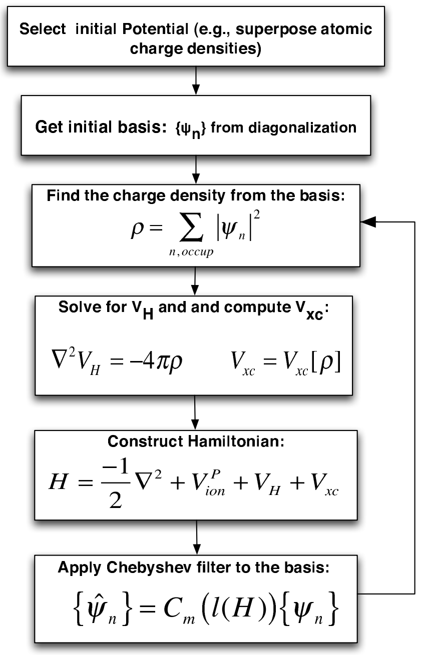
Select initial Potential (e.g., superpose atomic charge densities)
Get initial basis: $\{\psi_n\}$ from diagonalization
Find the charge density from the basis: $\rho = \sum_{n,occup} |\psi_n|^2$
Solve for V_H and and compute V_{xc}: $\nabla^2 V_H = -4\pi\rho \quad V_{xc} = V_{xc}[\rho]$
Construct Hamiltonian: $H = \frac{-1}{2}\nabla^2 + V_{ion}^P + V_H + V_{xc}$
Apply Chebyshev filter to the basis: $\{\hat{\psi}_n\} = C_m(l(H))\{\psi_n\}$

Figure 3: Flow diagram for obtaining a self-consistent solution of the Kohn-Sham equation using damped Chebyshev subspace filtering.

In summary, a standard SCF method has 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 essentially bypasses the second loop, or rather it merges it into a single outer loop, which can be considered as a *nonlinear subspace iteration algorithm*. The inner diagonalization loop is replaced by a single Chebyshev subspace filtering step.

4.2 Chebyshev filters and estimation of bounds

Chebyshev polynomials of the first kind are defined, for $k = 0, 1, \cdots$, by (see e.g., [41, p.371], or [43, p.142]):

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

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 wave func-
   tions; goto step 3.
```

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

```
1. Get the lower bounds $b_{low}$ and $\gamma$ 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 $\Phi$,
   where $\Phi$ contains the discretized wave functions of $\Psi_i(r)$, $i = 1, ..., s$:
   $\Phi$ = Chebyshev_filter($\Phi$, $m$, $b_{low}$, $b_{up}$, $\gamma$).
4. Ortho-normalize the basis $\Phi$ 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 cor-
       responding eigenvectors;
   (c) 'Rotate' the basis as $\Phi := \Phi Q$; return $\Phi$ and D.
```

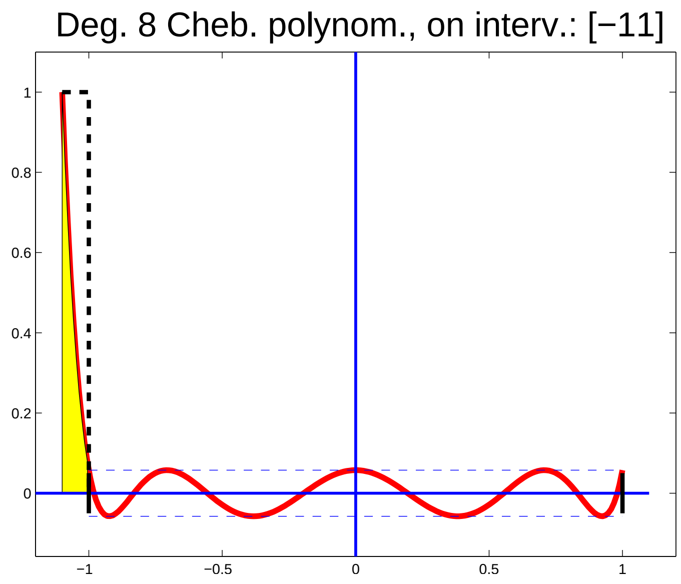

Figure 4: Degree 8 Chebyshev polynomial on the interval [-1, 1] scaled to one at $\gamma = -0.2$. The shaded area corresponds to eigen-components that will be amplified relative to the other eigencomponents, those corresponding to the interval $[-1, 1]$, which will be dampened.

Note that $C_0(t) = 1, C_1(t) = t$. The following important 3-term recurrence is easy to derive from properties of the cosine function,

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

By filtering we mean a process applied to a vector that has the effect of magnifiant desired eigen-components of this vector relative to other, undesirable, components. If the process is repeated indefinitely, the resulting vector will have zero components in the undesirable part of the spectrum. In our context, we need to filter out all components associated with the non-occupied states, or, equivalently to enhance the components associated with occupied states, relative to other components.

Filtering can be readily achieved by exploiting well-known properties of Chebyshev polynomials. It is known that among all polynomials of degree k, which have value one at a certain point $|\gamma| > 1$, the polynomial $C_k(t)/C_k(\gamma)$ is the one whose maximum absolute value in the interval $[-1, 1]$ is minimal. Thus, $C_k(t)/C_k(\gamma)$ can be viewed as an optimal polynomial if one wishes to dampen values of the polynomial in $[-1, 1]$ among all polynomials p of degree k, scaled so that $p(\gamma) = 1$. The 8th degree Chebyshev polynomial scaled at $\gamma = -0.2$ is shown in Figure 4.

Assume that the full spectrum of H (denoted by $\Lambda(H)$) is contained in $[\gamma, b]$. Then, in order to approximate the eigensubspace associated with the lower end of the spectrum, say $[\gamma, a]$ with $\gamma < a < b$, 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 [22]. 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 $X_1 = (H - cI)X_0$.

The above iteration is without any scaling. In the case of the interval $[-1, 1]$ we scaled the polynomial by $C_k(\gamma)$ in order to ensure that the value of the polynomial at γ equals one. For general intervals, this leads to the scaled sequence of polynomials [43]

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

Thus, the scaling factor is $\rho_j = C_j[\frac{2}{e}(\gamma - cI)]$. Clearly this requires an estimate for γ which, in our case, is the smallest eigenvalue of the Hamiltonian. However, since this is used for scaling, for the purpose of avoiding overflow, 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 γ. 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 [43] is shown in Algorithm 4.3 (the tildes and vector subscripts are omitted).

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 owing 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 $\Lambda(H)$. As seen in Algorithm 4.3, a desired filter can be easily controlled by adjusting two endpoints that bound the higher portion of $\Lambda(H)$.

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 [23], 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 $[a, b]$

Algorithm 4.3 $[Y] = \texttt{Chebyshev\_filter}(X, m, a, b, \gamma)$.

Purpose: Filter column vectors of X by an m degree Chebyshev polynomial in H that dampens on the interval $[a, b]$. Output the filtered vectors in Y.

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

with $b < \lambda_{max}(H)$ is mapped into $[-1, 1]$, then the $[b, \lambda_{max}(H)]$ portion of the spectrum will also be magnified, which will cause the procedure to fail. Therefore, it is imperative that the bound b be larger than $\lambda_{max}(H)$. On the other hand it should not be too large as this would result 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 $\lambda_{max}(H)$.

An inexpensive way to estimate an upper bound of $\Lambda(H)$ by the standard Lanczos [44] method is described in Algorithm 4.4, to which a safeguard step is added. The largest eigenvalue $\tilde{\lambda}$ 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. [41]). 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 $\Lambda(H)$. Larger k values (e.g., $k > 10$) are not necessary in general.

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 $\Lambda(H)$ 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 subspace to initiate the nonlinear SCF loop. This section discusses options available for this task.

In principle, any effective eigenvalue algorithms can be used for the first SCF step. PARSEC originally had three diagonalization methods: Diagla, which is a preconditioned Davidson method [28, 29]; the symmetric eigensolver in ARPACK [18, 45]; and the Thick-Restart Lanczos algorithm called TRLan [19, 20]. 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 [22], TRLan was used for the diagonalization at the first SCF step.

Another option suggested and tested in [32] but not implemented in PARSEC, is to resort to the Lanczos algorithm with partial reorthogonalization. Partial reorthogonalization Lanczos would run the Lanczos algorithm without restarting, reorthogonalizing the vectors only when needed, see [41]. This is a very effective procedure, some would even say optimal in some sense, except that it typically requires an enormous amount of memory. As illustrated in [32] the method can be 5 to 7 times faster than ARPACK for moderate size problems. It is possible to address the memory problem by resorting to secondary storage, though parallel implementations would be tedious.

At the other extreme when considering memory usage, one can use the Chebyshev filtered subspace iteration *in its linear implementation*. This means that we will now add an outer loop to the procedure described by Algorithm 4.2 and test convergence for the same Hamiltonian (the initial one) without updating potential from one outer loop to the next. Practically, this is simply as a variant of

Algorithm 4.1, whereby step 2 is replaced by as many filtering steps of Algorithm 4.2 as are required for the subspace to converge. This procedure is the most economical in terms of memory, so it is recommended if memory is an issue. However, it is well-known that subspace iteration methods (linear) are not as effective as the Lanczos algorithm, and other Krylov-based methods, see, e.g., [41, Chap. 14].

Even with standard restart methods such as ARPACK and TRLan, the memory demand can still remain too high in some cases. 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 [23, 24] was developed with these two goals in mind. Details can be found in [23, 24]. The principle of the method is to simply build a subspace by a procedure based on a form of Block-Davidson approach. The Block-Davidson approach builds a subspace by adding a 'window' of preconditioned vectors. In the Chebyshev-Davidson approach, these vectors are built by exploiting Chebyshev polynomials.

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 19,000 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).

Though results obtained with the Chebyshev-Davidson method in the first step diagonalization are satisfactory, there is still much work to be done in this area. We do not know for example how accurate the subspace must be in order to be a good initial guess to ensure convergence. It may possible to further reduce execution times by changing the stopping criterion needed in the first SCF step. It may be also possible to exploit well-known "global convergence" strategies utilized for non-linear iterations (such as continuation, or damping) to avoid completely the first step diagonalization.

6 Numerical Results

PARSEC has been applied to study a wide range of material systems (*e.g.* [12, 26, 27]). 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 Ref. [46] contains very interesting studies on clusters containing up to 1100 silicon atoms, using the well-known efficient plane wave DFT package VASP [8, 47]; however, it is stated in Ref. [46] 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 real-space pseudopotential methods [48], can now routinely solve silicon clusters with several thousands of atoms.

The hardware used for the computations is the SGI Altix cluster at MSI, it consists of 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 [29] 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. In Tables 4–11, 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.

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 4: $Si_{2713}H_{828}$, 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.)

The first example (Table 5) 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 [48] in PARSEC.

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 5: $Si_{525}H_{276}$, 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.

For larger clusters $Si_{2713}H_{828}$ (Table 4) and $Si_{4001}H_{1012}$ (Table 6), Diagla became too slow to be 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
1472440	8511	1652243	12	-89.12338	18.63 hrs.	38.17 hrs.

Table 6: $Si_{4001}H_{1012}$, 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 8) and $Si_{9041}H_{1860}$ (Table 7), 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.

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

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

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, one for the neutral cluster and another for the ionized one, and observe the variation in total energy between these calculations. Fig. 5 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 negative of the E_{HOMO} energy is lower than the IP in clusters [6], which is confirmed in Figure 5. In addition, the figure shows that the IP and $-E_{HOMO}$ approach each other in the limit of extremely large clusters.

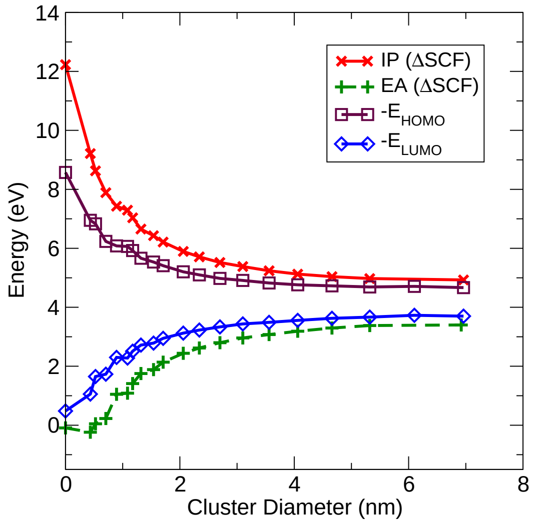

Figure 5: Ionization potential, IP, (crosses) and electron afÃđnity, EA, ("plus" signs), for various clusters with diameters ranging from 0 nm (SiH_4) to 7 nm ($Si_{9041}H_{1860}$). "Squares" denote the negative of the highest occupied molecular orbital ($-E_{HOMO}$) eigenvalue energy of the neutral cluster. "Diamonds" denote the negative of the lowest unoccupied molecular orbitaleigenvalue energy ($-E_{LUMO}$).

Fig. 5 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. Fig. 5 shows that, as the cluster grows in size, the EA approaches the negative of the lowest-unoccupied eigenvalue energy. A power-law analysis in Fig. 5 indicates that both the ionization potential and the electron affinity approach their bulk values according to a power-law decay R^n with $n \approx 1$. The numerical fits are:

$$\mathrm{IP} = \mathrm{IP}_0 + A/D^{\alpha} \tag{7}$$

$$\mathrm{EA} = \mathrm{EA}_0 - B/D^{\beta} \tag{8}$$

with $\mathrm{IP}_0 = 4.50$ eV, $\mathrm{EA}_0 = 3.87$ eV, $\alpha = 1.16$, $\beta = 1.09$, $A = 3.21$ eV, $B = 3.13$ eV. These values for A and B assume a cluster diameter D given in nanometers. The difference between ionization potential and electron affinity is the electronic gap of the nanocrystal. As expected, the value of the gap extrapolated to bulk, $\mathrm{IP}_0 - \mathrm{EA}_0 = 0.63$ eV, is very close to the energy gap predicted in various DFT calculations for silicon, which range from 0.6 eV to 0.7 eV [6,49]. Owing to the slow power-law decay, the gap at the largest crystal studied is still 0.7 eV larger than the extrapolated value.

Other properties of large silicon clusters are also expected to be similar to the ones of bulk silicon, which is equivalent to a nanocrystal of "infinite size". Fig. 6 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 Fig. 6, we used Gaussian functions with dispersion of 0.05 eV. More details are discussed in [50].

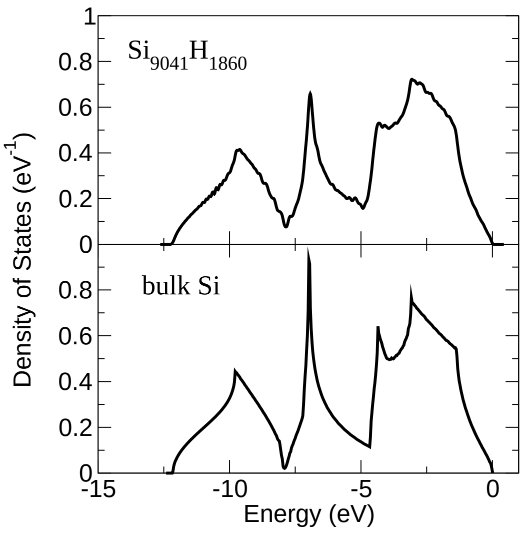

Figure 6: Density of states (DOS) of the cluster $Si_{9041}H_{1860}$ (upper panel) compared with periodic crystalline silicon (lower panel). As a consequence of the large size, the DOS of the $Si_{9041}H_{1860}$ cluster is very close to that of bulk silicon (the inÃđnite-size limit).

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 9: 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 10: Fe_{326}, using 24 processors. $m = 20$ for Chebyshev-Davidson; $m = 19$ for CheFS.

We also applied PARSEC to some large iron clusters. Tables 9–11 contain three clusters with more than 300 iron atoms. The number of states, n_{state}, is multiplied by two because these clusters are magnetized and spin degeneracy is broken. These metallic systems are well-known to be very difficult for DFT calculations, because of the "charge sloshing" [7, 8]. 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 [51]. It took more than 100 SCF steps

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

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 9–11, over 100 SCF steps using eigenvector-based methods would have required months to complete for each of these clusters.

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.

Additional tests not reported here, have also shown that the subspace filtering method is robust with respect to the initial subspace. Besides self-consistency, it can be used together with molecular dynamics or structural optimization, provided that atoms move by a small amount. Even after atomic displacements of a fraction of the Bohr radius, the CheFSI method was able to bring the initial subspace to the subspace of self-consistent Kohn-Sham eigenvectors for the current position of atoms, with no substantial increase in the number of self-consistent cycles needed.

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.

8 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. This work was supported by the MSI, by the National Science Foundation under grants ITR-0551195, ITR-0428774, DMR-013095 and DMR-0551195 and by the U.S. Department of Energy under grants DE-FG02-03ER25585, DE-FG02-89ER45391, and DE-FG02-03ER15491.

References

[1] P. Hohenberg and W. Kohn, Inhomogeneous electron gas. Phys. Rev. **136**, B864 (1964).

[2] W. Kohn and L. J. Sham, Self-consistent equations including exchange and correlation effects. Phys. Rev. **140**, A1133 (1965).

[3] J. C. Phillips, Energy-band interpolation scheme based on a pseudopotential. Phys. Rev. **112**, 685 (1958).

[4] J. C. Phillips and L. Kleinman, New method for calculating wave functions in crystals and molecules. Phys. Rev. **116**, 287 (1959).

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

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

[7] 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 (1992).

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

[9] W. Koch and M. C. Holthausen, *A chemist's guide to density functional theory* (Wiley-VCH, 2000).

[10] Y. Saad, J. R. Chelikowsky and S. Shontz,"Numerical methods for electronic structure calculations," SIAM [submitted].

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

[12] 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 (1994).

[13] 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).

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

[15] M.M.G. Alemany, M. Jain, J.R. Chelikowsky and L. Kronik: “A real space pseudopotential method for computing the electronic properties of periodic systems,” *Phys. Rev. B* **69**, 075101 (2004).

[16] T. Otsuka, T. Miyazaki, T. Ohno, D. R. Bowler and M. J. Gillan,“Accuracy of order-N density-functional theory calculations on DNA systems using CONQUEST,” J. Phys.:Condens. Matter **20**, 294201 (2008).

[17] S. Goedecker, Linear scaling electronic structure methods. Rev. Mod. Phys. **71**, 1085 (1999).

[18] 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/`.

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

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

[21] Y. Zhou, Y. Saad, M. Tiago, and J. Chelikowsky, Parallel self-consistent-Ãđeld calculations via Chebyshev- Ãđltered subspace acceleration. Phys. Rev. E **74**, 066704 (2006).

[22] Y. Zhou, Y. Saad, M. L. Tiago, and J. R. Chelikowsky, Self-consistent-field calculation using Chebyshev polynomial filtered subspace iteration. J. Comput. Phys. **243**, 1063 (2006).

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

[24] Y. Zhou. A Chebyshev-Davidson algorithm for large symmetric eigenvalue problems. Technical report, Minnesota Supercomputing Institute, Univ. of Minnesota, (in preparation).

[25] B. Fornberg and D. M. Sloan, A review of pseudospectral methods for solving partial differential equations. *Acta Numerica*, editor A. Iserles, number 3, p. 203. (Cambridge Univ. Press, 1994).

[26] 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 (2004).

[27] 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. Phys. Status Solidi B **243**, 1063 (2006).

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

[29] 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 (2000).

[30] 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**, 13472 (1998).

[31] S. Baroni and P. Giannozzi, Towards very large scale electronic structure calculations. Europhys. Lett. **17**, 547 (1992).

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

[33] F. L. Bauer, Das verfahren der treppeniteration und verwandte verfahren zur losung algebraischer eigenwertprobleme. Z. Angew. Math. Phys. **8**, 214 (1957).

[34] H. Rutishauser, Computational aspects of F. L. Bauer's simultaneous iteration method. Numer. Math. **13**, 4 (1969).

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

[36] 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**, 1376 (1994).

[37] S. Goedecker and L. Colombo, Linear scaling electronic structure methods. Phys. Rev. Lett. **73**, 122 (1994).

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

[39] 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 (1998).

[40] 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 (1999).

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

[42] 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 (1976).

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

[44] 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 (1950).

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

[46] 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**, 015502 (2004).

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

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

[49] W. G. Aulbur, L. Jönsson, and J. W. Wilkins, Quasiparticle calculations in solids *Solid State Physics*, eds. F. Seitz, D. Turnbull, and H. Ehrenreich, vol. **54**, 1, Academic, New York (2000).

[50] M. L. Tiago, Y. Zhou, Y. Saad, and J. R. Chelikowsky, Electronic properties and energetics of nanometer-size silicon nanocrystals. Technical report, and T.-L. Chan, M. L. Tiago, E. Kaxiras and J.R. Chelikowsky, "Size Limits on Doping Phosphorus into Silicon Nanocrystals," *Nano Letters* **8**, 596 (2008).

[51] 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, *Phys. Rev. Lett.* **97**, 147201 (2006), and to be published.