Chebyshev-filtered subspace iteration method free of sparse diagonalization for DFT calculations

Yunkai Zhou[*], James R. Chelikowsky[†], and Yousef Saad[‡]

January 27, 2014

Abstract

The Kohn-Sham equation in first-principles *density functional theory* (DFT) calculations is a nonlinear eigenvalue problem. Solving the nonlinear eigenproblem is usually the most expensive part in DFT calculations. Sparse iterative diagonalization methods that compute explicit eigenvectors can quickly become prohibitive for large scale problems. The Chebyshev-filtered subspace iteration (CheFSI) method avoids most of the explicit computation of eigenvectors, which results in significant speedup over iterative diagonalization methods for the DFT self-consistent field (SCF) calculations. However, the original CheFSI method utilized a sparse iterative diagonalization at the first SCF iteration to provide initial vectors for latter subspace filtering, and this diagonalization is expensive for large scale problems. We develop a new initial filtering step to fully avoid this first step diagonalization, thus making the CheFSI method free of sparse iterative diagonalizations. The new approach saves memory usage and can be two to three times faster than the original CheFSI method.

Key words: Density functional theory, self-consistency, Chebyshev filters, Hamiltonian, diagonalization, nonlinear eigenvalue problem, subspace filtering

1 Introduction

Density functional theory (DFT) [11, 15] plays a critical role in electronic structure calculations based on first principles. DFT greatly simplifies the multi-electron Shrödinger equation into the Kohn-Sham equation, which is effectively one-electron, where all non-classical electronic interactions are replaced by a functional of the charge density. Further aided by *pseudopotential theory* (e.g. [28, 4, 23]), which replaces the true atomic potential with a much smoother *pseudopotential* that can take into account the effect of core electrons, DFT significantly reduces the number of one-electron wave-functions that need to be computed. Nevertheless, even with the simplifications resulted from using pseudopotentials, it is still computationally challenging to solve the Kohn-Sham equation related to complex systems containing many thousands of atoms.

[*]Department of Mathematics, Southern Methodist University, Dallas, TX 75275, USA (`yzhou@smu.edu`.) Supported in part by the NSF under grant DMS-1228271 and a J. T. Oden fellowship at Univ. of Texas at Austin

[†]Center for Computational Materials, Institute for Computational Engineering and Science, and Department of Physics and Chemical Engineering, University of Texas, Austin, TX 78712, USA (`jrc@.ices.utexas.edu`.) Supported in part by

[‡]Department of Computer Science and Engineering, University of Minnesota, MN 55455, USA (`saad@cs.umn.edu`.) Supported in part by

Traditional approaches for solving the Kohn-Sham equation include methods using explicit bases, especially the planewave methods (e.g., [26, 16]) which expand wave-functions for periodic structures using planewave bases; and methods without using explicit bases, e.g., the real-space methods [5, 6, 32, 3].

Here we focus on solving the Kohn-Sham equation in the real-space setting. The advantages of a real-space approach include that parallel implementation is conceptually simple, that no super-cells is necessary for non-periodic structures, and that application of potentials onto electron wave-functions can be performed directly in real-space. Although Hamiltonian matrices from a real-space discretization is larger than that from a planewave expansion, they are very sparse and need not be stored explicitly, and they only need to be accessed via matrix-vector products that represent the application of the Hamiltonians on wave-functions.

It is well-known that the intrinsic nonlinearity in the Kohn-Sham equation can be addressed by utilizing a self-consistent-field (SCF) iteration, which involves solving a sequence of linearized Kohn-Sham equation. The most computationally expensive part of DFT calculations is the repeated solving of a linearized eigenvalue problem at each SCF iteration step.

Diagonalization methods are needed to solve the linear eigenvalue problem. Dense diagonalization methods that compute all eigenvectors quickly become impractical for relatively large systems due to the cubic order complexity, thus sparse iterative diagonalization methods such as [31, 19, 40, 13, 46] are used. However, larger systems can still render sparse diagonalization methods impractical.

We developed a nonlinear Chebyshev-filtered subspace iteration (CheFSI) method to alleviate the computational cost ([48, 47]). CheFSI tries to directly address the nonlinearity of the original Kohn-Sham equation by constructing a nonlinear subspace iteration. Explicit eigenvectors of the intermediate linearized Kohn-Sham eigenvalue problems no longer need to be computed, instead they are replaced by approximate basis vectors of a progressively refined subspace. Since approximate basis vectors are much easier to compute than eigenvectors, CheFSI significantly reduces the diagonalization cost. An order of magnitude speedup over eigenvector based methods was routinely obtained (see e.g. [47, 38]).

The CheFSI approach seeks a self-consistent solution via an SCF iteration, therefore it has the same accuracy as other SCF DFT approaches that are based on diagonalizations.

However, CheFSI as proposed in [48, 47] is not free of sparse diagonalization, mainly because its first SCF iteration step calls an iterative diagonalization solver in order to provide a good initial subspace for further refinement. For large problems, this first diagonalization step can be quite expensive and may constitute about 35%–45% of the total SCF CPU time, even if we call state-of-the-art solvers such as ARPACK [19], TRLanc [40], and ChebDav [46, 43].

This paper develops a filtering technique that completely avoids sparse diagonalization at the first SCF step. The filters used are again based on Chebyshev filters. In contrast to our former filtering approach in CheFSI, where the filtering lower bounds can be obtained from the Ritz values computed at the previous SCF iteration, here we do not have Ritz values from a previous SCF step since it is at the first SCF iteration. We overcome this difficulty by utilizing Ritz values generated from a Lanczos procedure. Since this procedure is needed in order to estimate a filter upper bound, by using its byproduct Ritz values, we can obtain a filter lower bound for the first SCF step, with only minor extra computation.

Recent progresses in [9] used localized nonorthogonal wave functions to reduce the cost in reorthogonalization, which showed improvement over the filtering approach proposed in [48]. Localized orbitals (e.g. [10]) are also utilized in e.g. [25, 22, 21] for accelerating DFT calculations.

We mention that similar localization techniques can be employed to further accelerate the method proposed in this paper, which will be part of our future works. Other interesting works that target to avoid diagonalizations include [24, 33, 20], they are based on approximating the Fermi-Dirac operator.

The organization of the paper is as follows: Section 2 briefly summarizes the eigenproblems in DFT calculations via SCF iteration; Section 3 discusses in some details the Chebyshev filters that play a significant role in our acceleration schemes; Section 4 presents the framework of the CheFSI method; Section 5 presents the filtering algorithm that avoids the first step diagonalization in an SCF iteration; and Section 6 presents numerical results and some discussions.

2 Eigenvalue problems in DFT calculations

The multi-electron Schrödinger equation is simplified into the following Kohn-Sham equation using density functional [11, 15],

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

where $\Psi_i(r)$'s are orthogonal wavefunctions also called eigenfunctions, λ_i's are Kohn-Sham eigenvalues, $\hbar$ is the Planck constant, and M is the electron mass. When atomic units are used, $\hbar = M = 1$.

The $\rho(r)$ in (1) is the *charge density*, which is defined by the wavefunctions,

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

Here n_{occ} is the number of occupied states, which equals to half of the number of *valence electrons* in the system. Equation (2) can be easily extended to address spin multiplicity or situations where the highest occupied states have fractional occupancy.

The operator inside the brackets in (1) is called Hamiltonian, it contains a kinetic energy term and a *total potential* term V_{total}. The V_{total}, also referred to as the *effective potential*, is defined as

$$V_{total}(\rho(r), r) = V_H(\rho(r), r) + V_{ion}(r) + V_{XC}(\rho(r), r). \tag{3}$$

The V_{total} contains three potentials: V_H the Hartree potential, which describes the electron-electron Coulomb repulsion; V_{ion} the ionic potential, which is approximated by pseudo-potentials; and V_{XC} the exchange-correlation potential, which includes all the many-particle interactions. V_{XC} is often approximated by LDA, GGA[27], OEP [18], or their variants such as LDA+U and GGA+U (e.g. [1, 7, 17]). Interested readers may consult recent books (e.g. [14, 23, 8]) for more details on DFT.

Both the Hartree potential and the exchange-correlation potential depend on the charge density $\rho(r)$. The ionic potential V_{ion} does not depend on $\rho(r)$, therefore it only needs to be calculated once in the SCF loop.

The most computationally expensive part of first principles DFT calculations is in solving the Kohn-Sham equation (1). The Hamiltonian in (1) depends on the charge density $\rho(r)$, which in turn depends on the wavefunctions of (1). Therefore (1) is essentially a nonlinear eigenvalue problem.

The nonlinearity in (1) is often addressed by a standard SCF iteration. At the first SCF iteration, an initial guess of the charge density is provided, from which one can calculate an initial V_{total}. This fixed V_{total} is substituted in (1), and after discretization it becomes a linearized eigenproblem of form $H\psi_i = \lambda_i\psi_i$. The H is the discretized Hamiltonian, it is a hermitian matrix, and ψ_i is the eigenvector corresponding to the discretized wavefunction $\Psi_i(r)$. The first n_{occ} wavefunctions of the linearized eigenproblem are then used to update $\rho(r)$ and V_{total} to obtained a new linearized eigenproblem of (1), which is solved again for the new $\Psi_i(r)$'s. The process is iterated until V_{total} (and hence the $\Psi_i(r)$'s) becomes stationary, at which stage the self-consistency is reached.

To be more specific, at each SCF step, say the ℓ-th step, the linearized eigenproblem

$$H^{[\ell]}\psi_i^{[\ell]} = \lambda_i^{[\ell]}\psi_i^{[\ell]}\ , \qquad \ell = 1, 2, \ldots, \tag{4}$$

is solved for at least number n_{occ} of smallest eigen-pairs. Let the dimension of each of these eigenproblems be n. For large n, dense eigensolvers are impractical due to their $O(n^3)$ complexity. Sparse eigensolvers can also become too expensive for two reasons: First, it has $O(n^2 n_{occ})$ complexity, and n_{occ} is often relatively large when n is large, moreover, the scalar implicit in the big-O notation in $O(n^2 n_{occ})$ is usually not small; second, the SCF iteration requires solving (4) several times, not just once. All these can make the accumulated sparse diagonalizations overwhelmingly expensive for the SCF loop to reach self-consistency.

The goal of the CheFSI acceleration is to significantly reduce the computational cost related to the sparse diagonalizations.

3 Spectrum filters for acceleration

We focus on Chebyshev polynomials (of the first kind) as spectrum filters due to their significant properties in introducing favorable gaps among wanted eigenvalues.

The degree-m Chebyshev polynomial is defined as (see e.g. [2, p.180])

$$C_m(x) = \begin{cases} \cos(m\ \cos^{-1}(x)), & |x| \le 1, \\ \cosh(m\ \cosh^{-1}(x)), & x > 1, \\ (-1)^m \cosh(m\ \cosh^{-1}(-x)), & x < -1. \end{cases} \tag{5}$$

Associated with this orthogonal polynomial is the following 3-term recurrence

$$C_{m+1}(x) = 2x\ C_m(x) - C_{m-1}(x), \quad m = 1, 2, ..., \tag{6}$$

which can be used to compute higher degree C_m from $C_0(x) = 1$ and $C_1(x) = x$.

To accelerate convergence of an invariant subspace, we exploit one of the most significant properties of the Chebyshev polynomials, namely the exponential growth of C_m outside the $[-1, 1]$ interval. This property is illustrated in Figure 1. Under comparable conditions, the exponential growth rate of C_m outside the $[-1, 1]$ interval is the *fastest* among all polynomials with degree $\le m$ ([29, p.31]).

The idea of using polynomial filters for accelerating eigenvalue calculations is based on the following well-known observation: Denote the eigenbasis of the Hamiltonian H as ψ_i's, with corresponding eigenvalues λ_i's, i.e., $H\psi_i = \lambda_i\psi_i$. Then any initial vector x_0 can be expanded in the eigenbasis as

$$x_0 = \alpha_1\psi_1 + \alpha_2\psi_2 + \cdots + \alpha_n\psi_n.$$

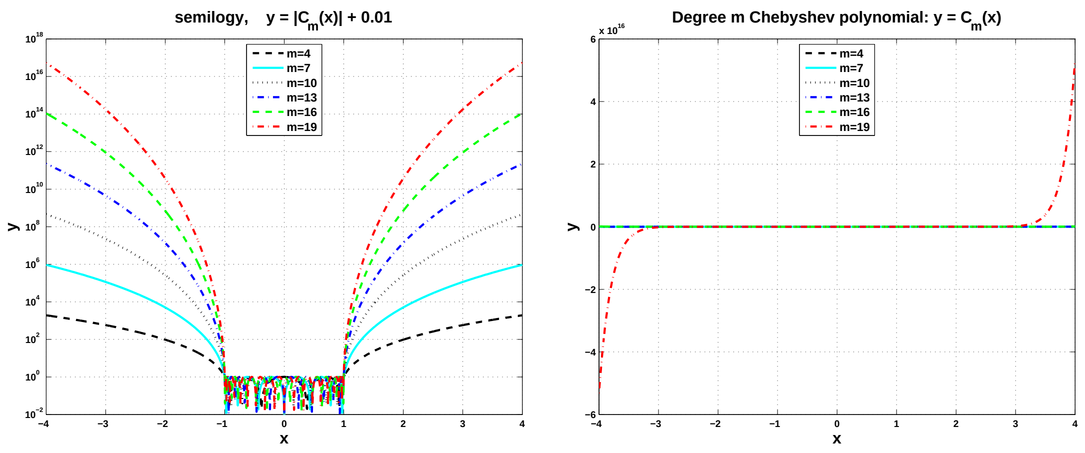


Figure 1: The fast growth of $C_m(x)$ outside the $[-1, 1]$ interval. The first figure uses semilog for the y-axis, for which we take the absolute value of $C_m(x)$ and add a small shift 0.01. The second figure does not use semilog, which shows the complete dominance of the highest degree polynomial over lower degree counterparts outside the $[-1, 1]$ interval.

Applying a polynomial filter $p(x)$ to H through a matrix-vector product leads to

$$p(H)x_0 = \alpha_1 p(\lambda_1)\psi_1 + \alpha_2 p(\lambda_2)\psi_2 + \cdots + \alpha_n p(\lambda_n)\psi_n,$$

where it is assumed that $\alpha_1 \neq 0$, which is almost always true in practice if x_0 is a random vector.

If the goal is to compute ψ_1 as fast as possible, then a suitable filter would be a $p(x)$ such that $p(\lambda_1)$ dominates $p(\lambda_j), j \neq 1$. That is, the filter introduces a significant gap between the wanted eigenvalue and the unwanted ones, so that after normalization $p(H)x_0$ will be mostly parallel to ψ_1. This is the well-known fact that the convergence rate of an eigen-algorithm mainly depends on the gap-ratio.

Our main approach for acceleration thus centers on utilizing the exponential growth property *outside* $[-1, 1]$ of the Chebyshev filters to introduce optimal gap-ratio of the filtered matrix. Interestingly, to construct a suitable filter we mainly concern what is *inside* the $[-1, 1]$ interval instead of what is outside of it. This is because for all $x \in [-1, 1]$ we have $|C_m(x)| \leq 1$. If we apply an affine mapping to map unwanted spectrum inside the $[-1, 1]$ interval, then the wanted spectrum will be automatically mapped outside $[-1, 1]$, the exponential growth property of C_m will then be automatically applied to the wanted spectrum, introducing better gap-ratio of wanted eigenvalues for faster convergence.

Therefore, the main concern is to figure out the unwanted part of the spectrum. More precisely, to find the two bounds that enclose the unwanted spectrum. If we have these two bounds, then it becomes easy to map the unwanted spectrum into $[-1, 1]$ for damping. The effectiveness of the constructed filter depends on these two filtering bounds.

In DFT calculations, the wanted eigenvalues are located at the lower end of the spectrum, since smaller eigenvalues correspond to the more stable lower energy *occupied states*.

The upper bound of the unwanted spectrum should be a value larger than all the eigenvalues of a given Hamiltonian. This is because we do not want to map any unwanted eigenvalues to be larger than 1, otherwise better gap-ratio will be introduced by the filter for the unwanted eigenvalues located at the higher end (as shown in Figure 1), this will prevent the algorithm from converging to the wanted invariant subspace corresponding to occupied states.

The upper bound can be obtained via the estimator developed in [48]. Here we use the refined estimator in [45], which provides a sharper upper bound and better filter performance in general. It is a Lanczos estimator with a safe-guard step to ensure that a true upper bound of the full spectrum is obtained, we refer to [45] for details.

The lower bound of the unwanted spectrum would be a value corresponding to the energy related to the highest occupied state. In theory this value is not easy to find, however, for the practical purpose of filtering that targets at improving gap-ratio, we only need a filtering lower bound that is greater than the energy of the highest occupied state. We will discuss how to obtain this lower bound in the next two sections.

Now we assume that the two filtering bounds are available and denote them as a and b, with $a < b$. The standard affine mapping that maps the interval $[a, b]$ into $[-1, 1]$ is

$$\mathcal{L}(x) = \left(x - \frac{b-a}{2}\right) / \left(\frac{b+a}{2}\right), \qquad x \in \mathbb{R}. \tag{7}$$

This same mapping will map the wanted eigenvalues located to the left of $[a, b]$ into a region to the left of $[-1, 1]$, for which the exponential growth property of the filter can be automatically utilized simply by calling the 3-term recurrences (6).

Since the upper bound b obtained using the estimator in [45] bounds the full spectrum of the Hamiltonian from above, the higher end of the spectrum that contains unwanted eigenvalues will be mapped inside $[-1, 1]$ for damping. Only the wanted lower end of the spectrum that is mapped to the left of $[-1, 1]$ will be magnified by the filter.

The affine mapping $\mathcal{L}(x)$ in (7) maps the smaller eigenvalues further away from $[-1, 1]$ than it does to larger eigenvalues, therefore the smaller eigenvalues will be magnified more by the polynomials (as seen from figure 1).

With suitably chosen filtering bounds, one can readily construct a degree-m Chebyshev polynomial $C_m(\cdot)$ to achieve desired filtering. Starting from $X_0 \in \mathbb{R}^{n\times d}$ that contains d initial vectors, let $p(H)$ be H pre-processed by (7) with the suitable bounds a and b, and then filtered by $C_m(\cdot)$, i.e.,

$$p(H) = C_m(\mathcal{L}(H)),$$

then the filtered matrix-vector product $p(H)X_0$ can quickly approximate a subspace spanned by the eigenvectors associated with the d smallest eigenvalues of H.

The filtered matrix-vector product $p(H)X_0 = C_m(\mathcal{L}(H))X_0$ can be computed via the 3-term recurrences as

$$X_{k+1} = \frac{4}{b+a}(H - \frac{b-a}{2}I)X_k - X_{k-1}, \qquad k = 1, 2, \ldots, m-1\ . \tag{8}$$

A pseudocode implementing the iteration (8) is listed in Algorithm 3.1.

One can introduce a scaling factor a_L which is smaller than a and apply the scaled filter

$$\tilde{p}(H) = p(H)/p(a_L). \tag{9}$$

The reason for the scaling is to prevent potential overflow, which may happen if m is large or if the smallest eigenvalue of H, denoted as $\lambda_{\min}(H)$, is mapped far away from -1. Both of the potential overflow situations can be addressed by choosing a_L as a rough estimate of $\lambda_{\min}(H)$.

Algorithm 3.1: $[Y] =$ `Chebyshev_filter`(X, m, a, b)

```
$e = (b - a)/2;$   $c = (a + b)/2;$
$Y = (H * X - c * X)/e;$
for  $i = 2$ to $m$  do
    $Y_t = (H * Y - c * Y) * (2/e) - X;$
    $X = Y;$   $Y = Y_t;$
```

The filter used in [48] is essentially the same as the Algorithm 3.1. Note that a_L is set to a in [48], which corresponds to $p(a_L) = 1$. This corresponds to a non-scaled Chebyshev filter. The non-scaled filter has its advantage as long as the two conditions for potential overflow are avoided. However, the scaled version with $a_L < a$ is the more stable filter, especially when a suitable a_L can be obtained with no extra computation. Since we can easily use the smallest Ritz value computed at each SCF step as a_L, we realize the scaled filtering with only marginal overhead.

By some algebra based on [30], we see that the 3-term recurrences using the scaled filter (9) are

$$X_{k+1} = \frac{2\sigma_{k+1}}{e}(H - cI)X_k - \sigma_{k+1}\sigma_k X_{k-1}, \qquad k = 1, 2, \dots, m-1 \;, \tag{10}$$

where c and e are respectively the center and half-width of the interval $[a, b]$, and the σ_i's are updated from $\sigma_1 = e/(c - a_L)$ as $\sigma_{k+1} = 1/\left(\frac{2}{\sigma_1} - \sigma_k\right)$. See [44] for derivation and analysis of iteration (10).

The pseudocode for computing the filtered vectors $Y = \tilde{p}(H)X$ using iteration (10) is listed in Algorithm 3.2. This algorithm as well as Algorithm 3.1 plays a significant role in the CheFSI framework.

Algorithm 3.2: $[Y] =$ `Chebyshev_filter_scaled`$(X,\ m,\ a,\ b,\ a_L)$

```
$e = (b - a)/2;$   $c = (a + b)/2;$   $\sigma = e/(c - a_L);$   $\tau = 2/\sigma;$
$Y = (H * X - c * X) * (\sigma/e);$
for  $i = 2$ to $m$ do
    $\sigma_{new} = 1/(\tau - \sigma);$
    $Y_t = (H * Y - c * Y) * (2 * \sigma_{new}/e) - (\sigma * \sigma_{new}) * X;$
    $X = Y;$   $Y = Y_t;$   $\sigma = \sigma_{new};$
```

A closer examination of Algorithms 3.2 and 3.1 reveals that three matrices X, Y, Y_t are used, each contains d columns. This clearly is *not desirable memory-wise*, especially if d is large, i.e., if X contains many columns. The merit of our approach is that the filtered subspace approach can be implemented in a memory economical way by performing *block-wise filtering*. Since essentially we only need the filtered subspace Y, we can overwrite X with Y. In a more memory economical implementation, we only need two temporary work arrays of size $n \times d_b$, where $d_b \ll d$, and do block-by-block filtering over X with block size no greater than d_b. If $d_b = 1$, then essentially we do column by column filtering over each column of X, the memory requirement of Algorithms 3.2 and 3.1 in this case is only $n \times d$ plus the memory for two length-n temporary vectors. This is to be contrasted with LOBPCG [13] that requires $n \times 3d$ memory since its subspace contains three blocks, each of size d. Our method uses about one third of the memory required by LOBPCG, thus more memory feasible when d is very large.

4 CheFSI framework for solving the Kohn-Sham equation

The main acceleration of CheFSI is achieved by computing basis vectors of an invariant subspace instead of eigenvectors. This is because the Hamiltonians during the intermediate SCF iterations are not exact (since the charge density is unknown), therefore there is no need to compute eigenvectors to high accuracy, but one should be careful not to miss wanted eigenvalues.

Clearly the invariant subspace of interest is the one that includes occupied states. Chebyshev filters can be designed to effectively compute this invariant subspace without risk of missing wanted eigenvalues.

Approximate eigenvectors can be obtained by a subspace rotation step, which is called Rayleigh-Ritz refinement. This rotation is of $O(n)$ complexity in contrast to the $O(n^2)$ complexity for sparse diagonalization. With targeted filtering and subspace rotation, one can gradually refine the computed basis vectors and converge them to the eigenvectors corresponding to occupied states. This is why CheFSI approach has the same accuracy as approaches that perform diagonalization at each SCF steps, but with much reduced computational cost.

Algorithm 4.1: CheFSI framework for DFT SCF calculations:

```
Start from an initial guess of $\rho(r)$, compute $V_{total}(\rho(r), r)$;
Solve the linearized Kohn-Sham equation $\left[-\frac{\hbar^2}{2M}\nabla^2 + V_{total}(\rho(r), r)\right]\Psi_i(r) = E_i\Psi_i(r)$ for
$\Psi_i(r),\ i = 1, 2, ..., s$, (where $s$ is an integer slightly larger than $n_{occ}$);
Compute new charge density $\rho(r) = \sum_{i=1}^{n_{occ}} |\Psi_i(r)|^2$;
Solve for new Hartree potential $V_H$ from $\nabla^2 V_H(r) = -4\pi\rho(r)$;
Update $V_{XC}$ and $V_H$; Compute the new total potential $\tilde{V}_{total}$ (with a potential-mixing
step) $\tilde{V}_{total}(\rho, r) = V_H(\rho, r) + V_{ion}(r) + V_{XC}(\rho, r)$;
If $\left\|\tilde{V}_{total} - V_{total}\right\| < tol$, then **stop** (self-consistency reached).
$V_{total} \leftarrow \tilde{V}_{total}$; Construct and apply Chebyshev filter to compute $s$ number of
approximate wavefunctions as follows:
7.1 Compute $b_{up}$ by calling the Lanczos estimator in [45];
7.2 Set $b_{low}$ and $a_L$ as the largest and smallest Ritz values from the previous iteration,
    respectively;
7.3 Apply degree-$m$ Chebyshev filter to $\Psi$ as $\Psi$ = Chebyshev_filter($\Psi$, $m$, $b_{low}$, $b_{up}$),
    or $\Psi$ = Chebyshev_filter_scaled($\Psi$, $m$, $b_{low}$, $b_{up}$, $a_L$), (here $\Psi$ is a matrix whose $s$
    columns are the discretized wavefunctions of $\Psi_i(r),\ i = 1, ..., s$);
7.4 Ortho-normalize the $s$ columns of $\Psi$ by an iterated Gram-Schmidt method;
7.5 Perform the Rayleigh-Ritz refinement step: (i) Compute the projection $\hat{H} = \Psi^T H \Psi$;
    (ii) Compute the eigendecomposition of $\hat{H}$: $\hat{H}Q = QD$, where $D$ contains eigen-
    values of $\hat{H}$ in nonincreasing order, and $Q$ contains the corresponding eigenvectors;
    (iii) Refine the basis as $\Psi := \Psi Q$.
Goto step 3 (use the first $n_{occ}$ columns of $\Psi$ as the discretized wavefunctions)
```

Algorithm 4.1 lists the main steps of the CheFSI framework for solving the Kohn-Sham equation using an SCF iteration. The main difference from a diagonalization-based approach SCF loop is that, after the first diagonalization at step 4.1 used to generate an initial basis vectors for filtering, CheFSI avoids diagonalization by replacing it with a subspace filtering step, as describe in step 4.1. The updated Hamiltonian H is implicitly applied throughout step 4.1 via matrix-vector products with the updated H.

As mentioned in the introduction, Algorithm 4.1 is not free of iterative sparse diagonalization, because step 4.1 calls a sparse diagonalization solver in order to provide a good initial subspace for refinement. This diagonalization can be expensive especially for large problems.

In the next section we propose a method to avoid this iterative diagonalization at the first SCF step.

5 First SCF step diagonalization by subspace filtering

There were two concerns that prevented us from using Chebyshev filters to replace the first diagonalization (step 4.1) in Algorithm 4.1. The first concern was that the filter bounds (especially the lower bound) were not available at the first SCF step. The second was that we wanted to generate a good initial basis Ψ_0 such that latter filtering could quickly refine it into the basis of the subspace associated with occupied states.

The first concern is not an issue after the first SCF step, since bounds to construct filters are readily available from the Ritz values computed at the previous SCF step. While at the first SCF step there are no previously computed Ritz values.

These two concerns made us apply iterative diagonalization to solve the linearized Kohn-Sham equation at the first SCF step in [48, 47].

However, by a careful reexamination of the structure of CheFSI, we realize that both concerns can be addressed and that we can apply Chebyshev filtered subspace method at the first SCF step, without using a sparse iterative diagonalization.

We can address the difficulty related to filter bounds by fully exploiting the modified Lanczos procedure [45, 43] used to estimate upper bound. This estimator computes a k-step Lanczos decomposition

$$HV_k = V_k T_k + f_k e_k^{\mathrm{T}}, \tag{11}$$

where $V_k^{\mathrm{T}} V_k = I_k,\ f_k^{\mathrm{T}} V_k = \mathbf{0}$, and e_k is the k-th canonical basis of $\mathbb{R}^k$. As proved in [45], the filtering upper bound can be set as

$$b_{up} := \max_{i \in \{1,\cdots,k\}} \lambda_i(T_k) + \|f\|_2\,, \tag{12}$$

where $\lambda_i(T_k)$ are the eigenvalues of the $k \times k$ tridiagonal matrix T_k. This bound only requires a small k in decomposition (11), in practice we use $4 \leq k \leq 10$.

To obtain a valid filtering lower bound b_{low}, we reuse the byproduct from the Lanczos bound estimator, namely the Ritz values $\lambda_i(T_k)$. The lower bound we use is

$$b_{low} := \beta \min_i \lambda_i(T_k) + (1-\beta) \max_i \lambda_i(T_k), \qquad \text{where } \beta \in [0.5, 1)\ . \tag{13}$$

The choice of bound (13) is quite natural if we notice that the variational property of eigenvalues guarantees that

$$\min_i \lambda_i(H) \leq \min_i \lambda_i(T_k) \leq \max_i \lambda_i(T_k) \leq \max_i \lambda_i(H). \tag{14}$$

In addition, by a well-known property of the standard Lanczos procedure, we know that the extreme eigenvalues of H (instead of the interior ones) are approximated much faster by $\lambda_i(T_k)$'s. Therefore we can set $\beta \approx 0.5$ in (13), which approximately corresponds to dampening the higher half of the spectrum of H. If β is set close to 1, then the filter will dampen $\lambda_i(H)$ located in the interval $[\min_i \lambda_i(T_k), \max_i \lambda_i(H)]$.

In reality this very first b_{low} is not particularly important. As long as it is not too small, which can cause to miss some occupied states, or too large, which may magnify unoccupied states, then it works fine. From this understanding and also from practice, we observe that $\beta = 0.5$ is a reasonable parameter and we use it in our computations reported in Section 6.

To address the second concern, we need to generate initial basis vectors that can capture a relatively significant portion of the wavefunctions corresponding to occupied states. This so called "good" initial basis will facilitate convergence for the latter subspace filtering.

The algorithm for this purpose turns out to be quite simple, as listed in Algorithm 5.1. The main idea is to apply Chebyshev filtering a few (say, `itmax`) times, each time with an automatically updated filtering lower bound b_{low}.

Algorithm 5.1: $[\Psi]$ = first_SCFstep_filter(`itmax`)

```
Call upper-bound estimator (use formula (12) from [43]) to get $b_{up}$ and Ritz values
$\lambda_i(T_k), i = 1, \ldots, k$;  set  $a_L = \min_i \lambda_i(T_k)$;
Set  $b_{low} = \beta \min_i \lambda_i(T_k) + (1-\beta) \max_i \lambda_i(T_k)$, where $\beta \in [0.5, 1)$ is a parameter usually
set to 0.5 ;
Set  $\Psi$  as a random matrix with $s$ column vectors  $(s > n_{occ})$;
for  $i = 1$ to itmax do
    Call  $\Psi$ = Chebyshev_filter_scaled($\Psi$, $m$, $b_{low}$, $b_{up}$, $a_L$);
    Orthonormalize $\Psi$ by iterated Gram-Schmidt ;
    Compute Rayleigh quotient: $G = \Psi^T * (H * \Psi)$;
    Compute eigen-decomposition of $G$:  $G = QDQ^T$;
    Sort the current Ritz values in increasing order:   $[R, indx]$ = sort(diag(D));
    Exit this for-loop if difference between previous and current Ritz values is small;
    Reset $b_{low}$ and $a_L$ as  $b_{low} = \max_i R$,  $a_L = \min_i R$;
Perform subspace rotation:  $\Psi = \Psi * Q(:, indx)$;
```

Note that step 5.1 in Algorithm 5.1 computes an eigendecomposition of the size-s Rayleigh quotient matrix G, which is used for subspace rotation at step 5.1.

The Ritz values contained in D from step 5.1 can be readily used to adaptively update the filtering lower bound b_{low}. We base the update rule (step 5.1) on the variational principle of eigenvalues, which is also called the Courant-Fisher min-max Theorem [12]. The principle is similar to (14) but with T_k replaced by G.

The first b_{low} is initially set according to (13), then updated according to step 5.1 in Algorithm 5.1. Applying the variational principle, and the fact that $s > n_{occ}$, we can guarantee that all occupied states will be magnified, (since b_{low} is always greater than all the wanted n_{occ} eigenvalues;) and that unwanted higher end of the spectrum of H will be dampened. A few iterations will result in a Ψ returned from Algorithm 5.1 to be the "good" initial basis which is suitable for latter subspace refinement.

The iteration number `itmax` at step 5.1 can be quite small thanks to the optimal magnifying and damping property of the Chebyshev polynomials. In practice three to four iterations suffice to generate a reasonably good initial Ψ.

As for the bound a_L, it is used only for scaling purpose as in (9), therefore any $a_L \leq b_{low}$ can serve this purpose. The one chosen in Algorithm 5.1 clearly satisfies this condition. In practice, the filtering Algorithm 3.1 without using a_L (or equivalently setting $a_L = b_{low}$) has almost identical performance as Algorithm 3.2 that uses $a_L < b_{low}$, especially so if the polynomial degree m is moderate.

The Lanczos procedure is required to obtain the upper bound b_{up}, this procedure also produces some Ritz values. Moreover, a dense eigen-decomposition (step 5.1 in Algorithm 5.1) is computed for the purpose of subspace rotation. Therefore by utilizing the available Ritz values, we obtain the filtering lower bounds with essentially no extra computations.

We apply Algorithm 5.1 to replace the expensive first SCF diagonalization (step 2) in Algorithm 4.1. With this replacement, the CheFSI approach for DFT calculation via an SCF loop becomes free of sparse iterative diagonalization. The whole process may be understood using as follows:

Starting from an initial charge density ρ_0, we call Algorithm 5.1 to generate an initial basis $\Psi_0 = [\ \psi_1, ..., \psi_s\ ]$, where s is an integer slightly larger than n_{occ}. After this step, we perform SCF iteration within a Do-loop,

Do $\ell = 1,\ 2, ...,\ n_{scf}$

(i) Get new charge density $\rho_\ell = 2\ \mathrm{diag}(\ \sum_{i=1}^{n_{occ}} \psi_i \psi_i^{\mathrm{T}}\ )$;

(ii) Implicitly construct a degree-m Chebyshev filter $p_m^{[\ell]}(H_\ell)$ and apply it to $\Psi_{\ell-1}$ to get $\Psi_\ell = p_m^{[\ell]}(H_\ell)\Psi_{\ell-1}$, (the Hamiltonian $H_{\ell-1}$ is implicitly updated to H_ℓ by modifying related matrix-vector products);

(iii) Ortho-normalize Ψ_ℓ to get the new basis $[\psi_1, ..., \psi_s]$.

At the final SCF step (where the n_{scf} denotes the iteration steps to reach self-consistency), Ψ_ℓ provides a good approximation to the basis of the eigensubspace associated with occupied states. The final basis may be expressed as

$$\Psi_{n_{scf}} = \prod_{\ell=1}^{n_{scf}} p_m^{[\ell]}(H_\ell)\Psi_0\ . \tag{15}$$

The basis from real computations may be different from the one in (15), this is because of subspace truncations, in which a dimension-s subspace is truncated into a dimension-n_{occ} subspace by throwing away larger eigen-components that correspond to unoccupied states.

Without consider the implementation details of truncation, the CheFSI method can be understood as a nonlinear subspace iteration (15), in which the iteration matrix $p_m^{[\ell]}(H_\ell)$ is updated at each step (in contrast, the iteration matrix remains unchanged in a standard subspace iteration). That is, we apply a nonlinear subspace iteration to directly address the nonlinearity in the Kohn-Sham equation (1), without emphasizing intermediate eigenvectors. It is this shift of emphasis from eigenvectors to basis vectors that provides the significant speedup of CheFSI over diagonalization-based approaches.

6 Numerical Results

We implement the various "diagonalization" algorithms in a Matlab package called RSDFT, where RS stands for *Real Space*. RSDFT is based on [5, 6], it aims to utilize the convenient features of Matlab for fast prototyping of new algorithms, as well as for teaching DFT calculations. (Another Matlab DFT package having similar purpose but based on planewave is [42].)

We use finite difference of eighth order to discretize the Kohn-Sham equation. The test problems used for the numerical experiments include three benzene molecules (C_6H_6), a buckminsterfullerene (C_{60}), three perovskite molecules ($MgSiO_3$) [37, 39], fifteen silicon dioxide molecules (SiO_2), forty two water molecules, and fifty two water molecules. Table 1 lists the dimensions of the Hamiltonian matrices for these molecules and their associated number of occupied states n_{occ}.

material	3 (C_6H_6)	C_{60}	3 ($MgSiO_3$)	15 (SiO_2)	42 (H_2O)	52 (H_2O)
Hamiltonian dim.	97,336	85,184	314,432	238,328	343,000	343,000
n_{occ}	94	244	76	244	340	420

Table 1: Dimension of the Hamiltonian matrix and its associated number of occupied states (n_{occ}) for each of the test problems.

For the numerical experiments, we compare the new algorithm with two other algorithms. The new algorithm is Algorithm 4.1 with its step 2 diagonalization replaced by the filtering Algorithm 5.1. We denote this algorithm as `1stFilt+CheFSI`. For the two algorithms to be compared with, one is SCF by iterative diagonalization at each SCF step, the other calls iterative diagonalization only at the first SCF step, and then it calls CheFSI for the remaining SCF steps.

Since the Matlab iterative eigensolver `eigs` is readily available, and it calls the ARPACK [19] — a *de facto* Fortran eigenvalue package, we choose to call `eigs` for the iterative diagonalizations. Because of this, we denote the method that does iterative diagonalization at each SCF step as `eigsSCF`, and the other method that calls diagonalization only at the first SCF step as `eigs+CheFSI`. There are certainly other eigensolvers written in Matlab, which may be used for the numerical experiments, but the numerical efficiency as well as robustness usually favor the Fortran ARAPCK package via `eigs`.

The hardware used for the numerical tests is a Linux workstation with 72GB RAM memory and dual 12-core Intel Xeon X5650 processors, each core has 2.67GHz CPU. This computer is dedicated for the reported computations. Note that Matlab can utilize the multi cores automatically, therefore the reported CPU time is the sum of CPU time on all cores. Although the `cputime` command in Matlab is known to have slight variance in measurement, our point here is that the difference in CPU time is quite significant for the three methods being compared, making the slight measuring variance of the `cputime` command negligible. Note that all the reported CPU time in this section refers to the total CPU time for "diagonalizations" only, which is usually over 90% of the total CPU time for the whole SCF simulation.

Figure 2 shows CPU time comparisons among `1stFilt+CheFSI`, `eigsSCF`, and `eigs+CheFSI`. The Chebyshev polynomial degree used for `1stFilt+CheFSI` and `eigs+CheFSI` is $m = 10$. The `itmax` in Algorithm 5.1 is set to 4. For the `eigs` calls in `eigsSCF` and `eigs+CheFSI`, we set the tolerance to 5×10^{-5}, this prevents `eigs` from taking too long to converge to unnecessarily high precision. However we cannot use a looser tolerance for `eigs` since it can risk missing some

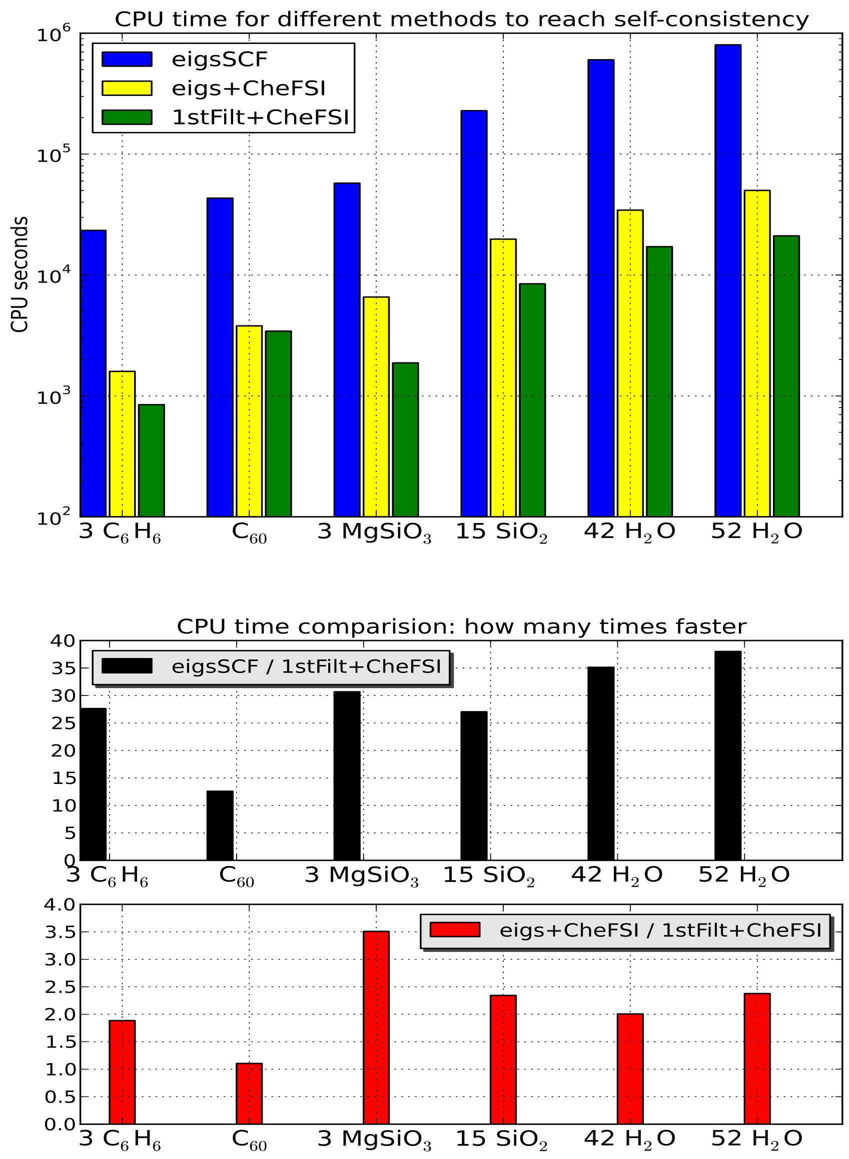


Figure 2: CPU time comparisons. The top figure shows the total CPU time the three methods took to reach self-consistency for the test problems listed in Table 1. Here $m = 10$ for all the tests. We used log scale for the y-axis since otherwise the bar length for `eigsSCF` would be overly dominant and make the bars for the other two methods hard to observe. To make the comparisons easier to observe, we plot the CPU time of `eigsSCF` and `eigs+CheFSI` over the CPU time of `1stFilt+CheFSI`. The bottom figure shows that `1stFilt+CheFSI` is easily over an order of magnitude faster than `eigsSCF`, and over twice faster than `eigs+CheFSI` on the larger problems from Table 1.

wanted eigenvalues and lead to no SCF convergence.

The comparisons shown in Figure 2 is quite impressive, considering that `1stFilt+CheFSI` is purely Matlab code, while `eigs+CheFSI` and `eigsSCF` both call the optimized Fortran code `ARPACK` via `eigs`.

Indeed the majority of the speedup is obtained via the CheFSI framework, which replaces the expensive diagonalization at each SCF step by a subspace filtering. This leads to the order of magnitude speedup, as seen from Figure 2. Our current method further reduces the expensive first step diagonalization that remained in the original CheFSI framework [47], this leads further to about twice speedup for larger problems.

For the tests shown in Figure 2, we simply set $m = 10$ as the degree for the filter polynomials. Clearly the impact of m on the efficiency of our algorithm cannot be seen from this figure. Because theoretic result guiding the choice of an optimal m is not available, we did many further numerical experiments with varying degree m to evaluate the impact of this parameter.

We report three representative results in Figure 3, where the convergence history of all three methods are shown.

Figure 3 shows that for the test problem (3 $MgSiO_3$), as m increases from 8 to 14, the total SCF steps decrease. However, this trend is not necessary true for other test problems, especially when n_{occ} is relatively large, as seen in Figures 4 and 5.

In Figures 4 and 5, we focus only on the `1stFilt+CheFSI` method, so that more choices of m can be presented in the same figure without overcrowding it.

There are two reasons for total SCF steps not necessarily going down with increasing degree m: One is the randomness in the initial vectors; the other is the difference in the automatically adjusted filtering bounds (the difference may be sizable when n_{occ} is relatively large), which can affect the filter efficiency.

Our conclusion is that it is better to use a moderate m, especially when the wanted eigenvalues are not clustered. Since a larger m implies more computation at each SCF filtering step, and it may not necessarily reduce the total SCF steps, the effect of a relatively large m (say, $m \geq 50$) often does not contribute to reducing the overall CPU time. In fact, for the two problems in Figures 4 and 5, the smallest $m = 8$ consumes the least CPU time for each problem respectively. We do not present the results for $m > 20$ for these two problems but mention that they do not further reduce the total SCF steps, hence they tend to use more CPU time with increasing $m > 20$.

Although choosing an "optimal" m appears to be a tricky theoretical problem, in practice it is quite easy to choose an m that performs well for a wide variety of problems. In general we recommend an m satisfying $8 \leq m \leq 40$; the larger m in this range is preferred only when the wanted part of the spectrum is more clustered. We can get some insight from the exponential growth of the Chebyshev polynomials as shown in Figure 1. This figure implies that m around 20 would be quite sufficient to introduce favorable spectrum gap into the filtered problems, therefore we think m in the range of 8 and 40 should be sufficient for most problems, including difficult problems for which the wanted eigenvalues are highly clustered.

Another advantage of our current method is the memory requirement. The standard Lanczos algorithm usually requires far more memory than the implicit restart Lanczos implemented in ARPACK, it can become impractical even for moderately large problems. In fact we implemented the Lanczos algorithm in RSDFT for the first step diagonalization, Matlab always exited with an "*Out of memory*" message when calling Lanczos for the larger problems listed in Table 1.

Other well-known iterative diagonalization methods, such as the ARPACK [19], Thick-restart

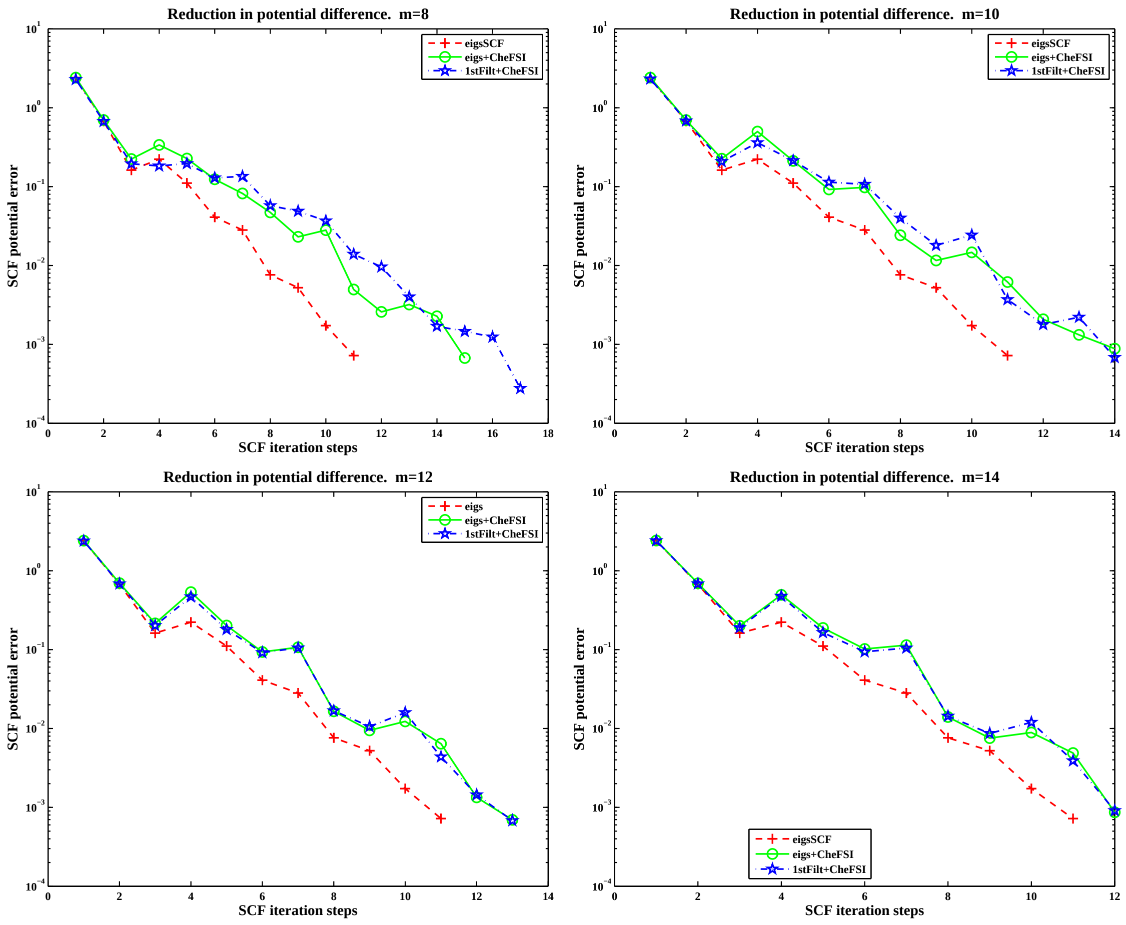
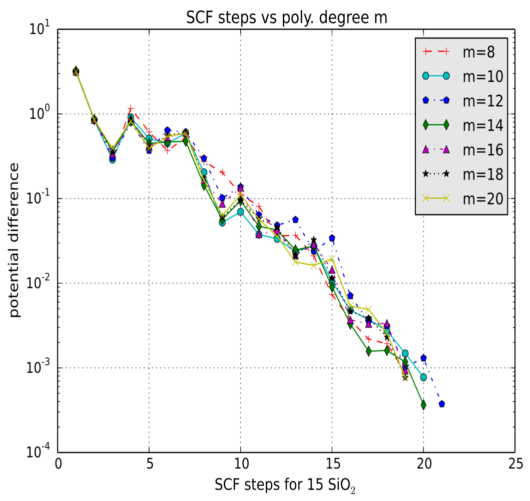


Figure 3: Effect of the polynomial degree m on the number of total SCF steps for the three perovskite molecules $MgSiO_3$. For this test problem, as m increases, the total SCF steps for the filtering methods `eigs+CheFSI` and `1stFilt+CheFSI` decrease. (In this figure the same convergence history of `eigsSCF` is shown in each subfigure as a quick reference.)



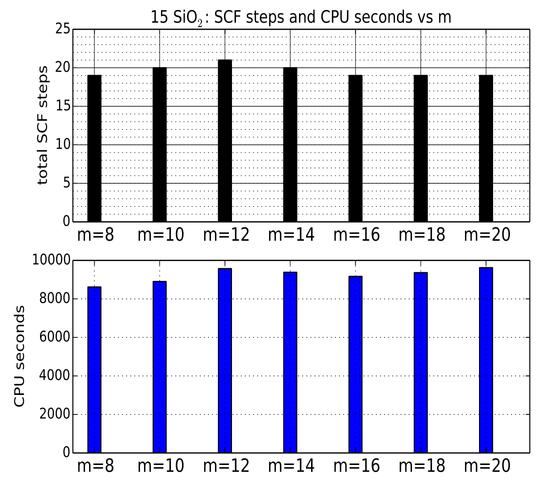
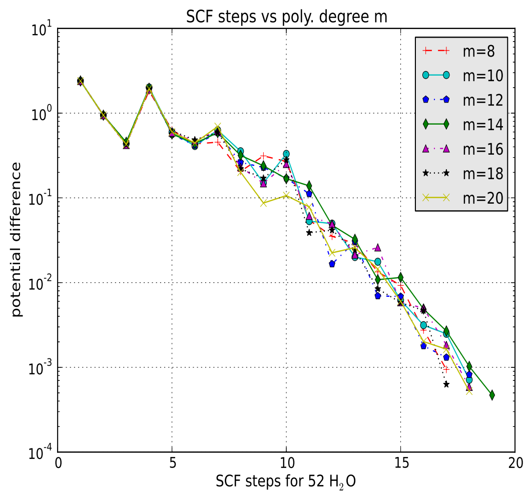


Figure 4: Left figure shows the convergence history of the `1stFilt+CheFSI` method with varying m applied to the 15 SiO_2 molecules. Right figure shows the total SCF steps and the total CPU time to reach self-consistency for each m.



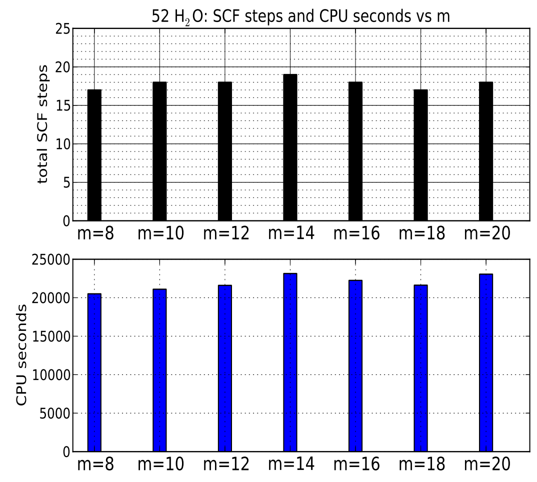


Figure 5: Left figure shows the convergence history of the `1stFilt+CheFSI` method with varying m applied to the 52 H_2O molecules. Right figure shows the total SCF steps and the total CPU time to reach self-consistency for each m.

Lanczos [40, 41], Davidson-type methods [34, 36, 35], and LOBPCG [13], all require at least $2n_{occ}$ as the dimension of the iterative subspace in order to effectively approximate wanted eigenvectors. By contrast our current method, now solely based on subspace filtering, requires the iterative subspace dimension to be only slightly larger than n_{occ}. This saves significant memory, especially for problems with a very high dimension n and a large n_{occ}.

7 Concluding Remarks

In our previous CheFSI method [48, 47], the diagonalization at the first SCF step can be quite expensive for large problems. We present a filtering method that avoids this diagonalization, thus making the CheFSI method free of sparse iterative diagonalizations at all SCF iteration steps. Our current method maintains the strength of the previous CheFSI method, with the added advantage that the first SCF step is vastly accelerated and that memory requirement is reduced by half comparing with standard iterative diagonalization methods. The numerical results obtained via the RSDFT Matlab package show the promise of our current method. We are in the process of implementing the algorithm in our Fortran package called PARSEC.

References

[1] V. I. Anisimov, F. Aryasetiawan, and A. I. Lichtenstein. First-principles calculations of the electronic structure and spectra of strongly correlated systems: the LDA+U method. *J. Phys.: Condens. Matter*, 9:767–808, 1997.

[2] O. Axelsson. *Iterative Solution Methods*. Cambridge Univ. Press, 1994.

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

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

[5] 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.

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

[7] M. Cococcioni and S. de Gironcoli. Linear response approach to the calculation of the effective interaction parameters in the $LDA+U$ method. *Phys. Rev. B*, 71(3):035105, 2005.

[8] B. Engel and R. M. Dreizler. *Density Functional Theory: An Advanced Course*. Theoretical and Mathematical Physics. Springer, 2011.

[9] C. J. García-Cervera, J. Lu, Y. Xuan, and Weinan E. Linear-scaling subspace-iteration algorithm with optimally localized nonorthogonal wave functions for Kohn-Sham density functional theory. *Phys. Rev. B*, 79(11):1–13, 2009.

[10] S. Goedecker and M. P. Teter. Tight-binding electronic-structure calculations and tight-binding molecular dynamics with localized orbitals. *Phys. Rev. B*, 51:9455–9464, 1995.

[11] P. Hohenberg and W. Kohn. Inhomogeneous electron gas. *Phys. Rev.*, 136:B864–871, 1964.

[12] R. A. Horn and C. R. Johnson. *Matrix Analysis.* Cambridge University Press, 1985.

[13] A. V. Knyazev. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM J. Sci. Comput.*, 23(2):517–541, 2001.

[14] W. Koch and M. C. Holthausen. *A chemist's guide to density functional theory.* Wiley-VCH, 2nd edition, 2001.

[15] W. Kohn and L. J. Sham. Self-consistent equations including exchange and correlation effects. *Phys. Rev.*, 140:A1133–1138, 1965.

[16] 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.

[17] H. J. Kulik, M. Cococcioni, D. A. Scherlis, and N. Marzari. Density Functional Theory in Transition-Metal Chemistry: A Self-Consistent Hubbard U Approach. *Phys. Rev. Lett.*, 97(10):103001, 2006.

[18] S. Kümmel and J. P. Perdew. Optimized effective potential made simple: Orbital functionals, orbital shifts, and the exact Kohn-Sham exchange potential. *Phys. Rev. B*, 68(3):035103, 2003.

[19] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK User's Guide: Solution of Large Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods.* SIAM, 1998.

[20] L. Lin, J. Lu, L. Ying, and Weinan E. Pole-based approximation of the Fermi-Dirac function. *Chinese Ann. Math. - Ser. B*, 30:729–742, 2009.

[21] L. Lin, J. Lu, L. Ying, and Weinan E. Adaptive local basis set for KohnSham density functional theory in a discontinuous Galerkin framework I: Total energy calculation. *J. Comput. Phys.*, 231(4):2140–2154, 2012.

[22] L. Lin, J. Lu, L. Ying, and Weinan E. Optimized local basis set for KohnSham density functional theory. *J. Comput. Phys.*, 231(13):4515–4529, 2012.

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

[24] T. Ozaki. Continued fraction representation of the Fermi-Dirac function for large-scale electronic structure calculations. *Phys. Rev. B*, 75(3):035123, 2007.

[25] T. Ozaki. Efficient low-order scaling method for large-scale electronic structure calculations with localized basis functions. *Phys. Rev. B*, 82(7):075131, 2010.

[26] 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.

[27] J. P. Perdew, K. Burke, and M. Ernzerhof. Generalized gradient approximation made simple. *Phys. Rev. Lett.*, 77(18):3865–3868, 1996.

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

[29] T. J. Rivlin. *An Introduction to the Approximation of Functions.* (1969 1st edition), Dover, 2003.

[30] Y. Saad. Chebyshev acceleration techniques for solving nonsymmetric eigenvalue problems. *Math. Comp.*, 42(166):567–588, 1984.

[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] 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.

[33] R. B. Sidje and Y. Saad. Rational approximation to the Fermi-Dirac function with applications in density functional theory. *Numerical Algorithms*, 56(3):455–479, 2011.

[34] G. L. G. Sleijpen and H. A. van der Vorst. A Jacobi–Davidson iteration method for linear eigenvalue problems. *SIAM J. Matrix Anal. Appl.*, 17:401–425, 1996.

[35] A. Stathopoulos and J. R. McCombs. Nearly optimal preconditioned methods for hermitian eigenproblems under limited memory. part II: Seeking many eigenvalues. *SIAM J. Sci. Comput.*, 29(5):2162–2188, 2007.

[36] A. Stathopoulos, Y. Saad, and K. Wu. Dynamic thick restarting of the Davidson and the implicitly restarted Arnoldi methods. *SIAM J. Sci. Comput.*, 19:227–245, 1998.

[37] L. Stixrude and B. B. Karki. Structure and freezing of $MgSiO_3$ liquid in earth's lower mantle. *Science*, 310:297–299, 2005.

[38] M. L. Tiago, Y. Zhou, M. Alemany, Y. Saad, and J. R. Chelikowsky. Evolution of magnetism in iron from the atom to the bulk. *Phys. Rev. Lett.*, 97:147201, 2006.

[39] K. Umemoto, R. M. Wentzcovitch, and P. B. Allen. Dissociation of $MgSiO_3$ in the cores of gas giants and terrestrial exoplanets. *Science*, 311:983–986, 2006.

[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] C. Yang, J. C. Meza, B. Lee, and L.-W. Wang. KSSOLV — a MATLAB Toolbox for Solving the Kohn-Sham Equations. *ACM Trans. Math. Softw.*, 36(2):10:1–35, 2009.

[43] Y. Zhou. A block Chebyshev-Davidson method with inner-outer restart for large eigenvalue problems. *J. Comput. Phys.*, 229(24):9188–9200, 2010.

[44] Y. Zhou. Practical acceleration for computing the HITS ExpertRank vectors. *Journal of Computational and Applied Mathematics*, 236(17):4398–4409, 2012.

[45] Y. Zhou and R.-C. Li. Bounding the spectrum of large hermitian matrices. *Linear Algebra Appl.*, 435(3):480–493, 2011.

[46] Y. Zhou and Y. Saad. A Chebyshev-Davidson algorithm for large symmetric eigenvalue problems. *SIAM J. Matrix Anal. Appl.*, 29(3):954–971, 2007.

[47] Y. Zhou, Y. Saad, M. L. Tiago, and J. R. Chelikowsky. Parallel self-consistent-field calculations using Chebyshev-filtered subspace acceleration. *Phys. Rev. E*, 74(6):066704, 2006.

[48] Y. Zhou, Y. Saad, M. L. Tiago, and J. R. Chelikowsky. Self-consistent-field calculation using Chebyshev-filtered subspace iteration. *J. Comput. Phys.*, 219(1):172–184, 2006.