

RESCU: a Real Space Electronic Structure Method

Vincent Michaud-Rioux,* Lei Zhang,[†] and Hong Guo
*Center for the Physics of Materials, Department of Physics,
McGill University, Montreal, Canada H3A 2T8*

In this work we present RESCU, a powerful MATLAB-based Kohn-Sham density functional theory (KS-DFT) solver. We demonstrate that RESCU can compute the electronic structure properties of systems comprising many thousands of atoms using modest computer resources, e.g. 16 to 256 cores. Its computational efficiency is achieved from exploiting four routes. First, we use numerical atomic orbital (NAO) techniques to efficiently generate a good quality initial subspace which is crucially required by Chebyshev filtering methods. Second, we exploit the fact that only a subspace spanning the occupied Kohn-Sham states is required, and solving accurately the KS equation using eigensolvers can generally be avoided. Third, by judiciously analyzing and optimizing various parts of the procedure in RESCU, we delay the $O(N^3)$ scaling to large N , and our tests show that RESCU scales consistently as $O(N^{2.3})$ from a few hundred atoms to more than 5,000 atoms when using a real space grid discretization. The scaling is better or comparable in a NAO basis up to the 14,000 atoms level. Fourth, we exploit various numerical algorithms and, in particular, we introduce a partial Rayleigh-Ritz algorithm to achieve efficiency gains for systems comprising more than 10,000 electrons. We demonstrate the power of RESCU in solving KS-DFT problems using many examples running on 16, 64 and/or 256 cores: a 5,832 Si atoms supercell; a 8,788 Al atoms supercell; a 5,324 Cu atoms supercell and a small DNA molecule submerged in 1,713 water molecules for a total 5,399 atoms. The KS-DFT is entirely converged in a few hours in all cases. Our results suggest that the RESCU method has reached a milestone of solving thousands of atoms by KS-DFT on a modest computer cluster.

PACS numbers: 31.15.E-, 71.15.-m, 02.70.Bf, 31.15.xr,

I. INTRODUCTION

Density functional theory (DFT)¹ based numerical programs are nowadays the standard tool for predicting and understanding the structural and electronic properties of materials that involve many electrons. The idea of treating complicated many-body interactions in real materials by a self-consistent mean field theory appeared in the early days of quantum mechanics. In 1927, Thomas and Fermi proposed a semiclassical model^{2,3} in which electrons in an external potential are described using only the electronic density. Subsequent calculations are simplified since the complicated many-body wavefunction is avoided. The Thomas-Fermi model was later improved by Dirac who included an exchange energy functional⁴ and by von Weizsäcker who added a gradient correction to the kinetic energy functional⁵. Nearly four decades later, Hohenberg and Kohn (HK) put DFT on firm theoretical footing by proving that the ground-state expectation values are functionals of the density and that the ground-state density can be calculated by minimizing an energy functional¹. Certain assumptions of the original HK theorems, such as the ground-state non-degeneracy, were later relaxed or eliminated⁶. These theories proved that the ground-state properties of any electronic system can in principle be calculated - if not necessarily understood - without using many-body wave functions. For practical applications, Kohn and Sham (KS) demonstrated that the problem of minimizing the total energy of a system with respect to the electronic density could take the form of a non-interacting electron problem⁷. In the

KS formulation, the kinetic energy is evaluated via single particle wave functions which is more accurate than using kinetic energy functionals that depend explicitly on the density. KS-DFT allows one to analyze a variety of physical systems and performing a DFT calculation today is all but synonymous to solving the KS equation⁷.

Various approaches for solving the KS equation have emerged such as the full potential all-electron methods^{8,9} and the *ab initio* pseudopotential methods¹⁰⁻¹⁴. In KS-DFT solvers, several bases have been used to express quantum mechanical operators including real space Cartesian grids, finite elements, planewaves, wavelets, numerical atomic orbitals (NAO), Gaussian orbitals, muffin-tin orbitals and some others. The goal is to predict structural and electronic properties of real materials reaching the required accuracy for the given research topic and KS-DFT is playing a prominent role in materials physics and engineering.

At present, a major issue of practical DFT methods is their limited capability of solving material problems involving large number of atoms using a small computer cluster (e.g. 16 to 256 cores). For instance, algorithms implemented in state-of-the-art electronic packages such as VASP^{12,13} and AbInit¹⁵ can comfortably solve systems comprising of a few hundred atoms - but not many thousands on such small computer clusters. With the widening accessibility of supercomputers and the developments of advanced parallel computing algorithms, heroic KS-DFT calculations at the level of 10,000 atoms became possible in recent years, but at the expense of using thousands or even tens of thousands of computing cores¹⁶⁻¹⁸. Nevertheless, for practical material research and innova-

tion, many small research groups in the world do not have access, cannot afford or simply wish not to use supercomputers. An urgent and very important task is to develop a KS-DFT method that can solve the KS equation without degrading the solution accuracy, at the level of several thousand atoms or more on a small computer cluster. It is the purpose of this work to report and describe such a KS-DFT solver and its associated software implementation.

To see why it is still possible to gain computational efficiency in traditional eigenvalue-based KS-DFT approaches, we note - as others had noted before us^{19,20} - that the solution process of the KS-DFT is a self-consistent procedure where one numerically converges the Hamiltonian step by step by solving the KS equation repeatedly and accurately. However, it appears unclear why one has to solve accurately the KS equation for the not-yet-converged Hamiltonian in the intermediary steps. Another observation is that, in the eigensolver-based KS-DFT methods, different parts in the computation scale differently as a function of electron number N , some $O(N)$, others $O(N^2)$ and eventually these are dominated by the $O(N^3)$ parts. If one is able to “delay” the crossover to $O(N^3)$ scaling, larger systems can potentially be solved using small computers. It turns out that these computational gains can be realized as we present below.

Our KS-DFT method combines NAO and the real space finite-differences plus Chebyshev filtering (CF) technique introduced by Zhou *et al.*^{19,20}. We found it is key to generate efficiently a proper initial subspace in the Chebyshev filtering framework, and this is achieved by the use a NAO basis. We advance efficient parallelization, a partial Rayleigh-Ritz (pRR) method for the computation of the density matrix and careful optimization of the solution process, and we have reached our goal of solving solid state physics problems consisting of thousands of atoms using 16 to 256 cores. Our code is called RESCU - which stands for Real space Electronic Structure Calculator - and it is implemented in the technical computing platform MATLAB. We use our own MPI and ScaLAPACK interfaces to harness efficiently the computational power of the cores. As such, RESCU combines the vocations of a prototyping code and a production code. In particular, the pRR allows us to compute the single particle density matrix in problems involving an exceedingly large number of electrons by taking advantage of the quasi-minimal property of basis sets built from CF. In the present paper, we will present the algorithmic and implementation advancements achieved during the development of RESCU. As practical examples, we demonstrate the following KS-DFT calculations: we simulate 5,832 Si atoms (23,328 electrons) on a real space grid, converging the entire KS-DFT calculation using 256 cores for about 5.5 hours; we simulate 4,000 Al atoms (12,000 electrons) on a real space grid, converging the entire KS-DFT calculation using 64 cores for about 5.1 hours; we simulate a supercell consisting of 13,824 Si atoms (55,296 electrons)

using a NAO basis, converging the entire calculation using 64 cores for about 6.4 hours; we simulate a supercell consisting of 5,324 Cu atoms (58,564 electrons) using a NAO basis, converging the entire calculation using 256 cores for about 12 hours. We also consider a disordered system consisting of a small DNA molecule submerged in 1,713 water molecules, for a total of 5,399 atoms (14,596 electrons), and converge the entire KS-DFT run in 9.6 hours on 256 cores. These results are compiled in table II which is found in section VII. The scaling of the RESCU method is presented going from 16 cores to 256 cores for various tests. Finally, since RESCU is primarily a real space implementation of KS-DFT, it does not require periodicity when dealing with condensed phase materials and can thus easily treat problems involving interfaces, surfaces, defects, disordered materials, etc.

The paper is organized as follow. In section II, we briefly state the fundamentals of DFT and introduce the single particle density matrix theoretical framework which is used throughout this article. In section III, we review the state-of-the-art numerical methods for solving the KS equations and recount their advantages and disadvantages. In section IV, we describe in some detail the Chebyshev filtering method. In section V, we present a computational complexity analysis of the Chebyshev filtering method and introduce the partial Rayleigh-Ritz algorithm. We explain how it takes advantage of the Chebyshev filtered basis sets to improve on the standard Rayleigh-Ritz algorithm. In section VI, we describe the implementation of the Kohn-Sham DFT solver RESCU. In section VII, we present different benchmarks of the RESCU code. We provide evidence when the partial Rayleigh-Ritz algorithm achieves significant gains over the standard Rayleigh-Ritz algorithm. We show how to generate a good quality initial subspace efficiently. Finally, we report simulations including thousands of atoms with modest computer resources. Bottlenecks and future direction will be discussion in section VIII and IX.

II. A BRIEF DISCUSSION OF DFT

Before delving into the details of the RESCU method, we briefly discuss KS-DFT in general terms. As mentioned in the introduction, the founding result of DFT is that the Hamiltonian of a system is uniquely determined by its ground-state electronic density^{1,6}. It follows that the ground-state wave function and the associated expectation values are also determined by the ground-state density, and there exists a universal energy functional of the density which is minimized by the ground-state density^{1,6}.

In the KS-DFT, the problem of minimizing the energy with respect to the density is mapped to a non-interacting electron problem⁷. The KS-DFT formulation made it possible to develop reasonably accurate energy functionals and it became the most successful and widely applied flavor of DFT. In particular, accurate kinetic en-

ergy functionals use the Kohn-Sham orbitals and not the density *per se*. The Kohn-Sham equation is usually written as the the following set of equations

$$\lambda_i \psi_i = \left(-\frac{1}{2} \nabla^2 + V_{ext} + V_H + V_{xc} \right) \psi_i \quad (1)$$

$$\rho(\mathbf{r}) = \sum_{i=1}^{\infty} n_{FD}(\lambda_i, \mu) \psi_i^*(\mathbf{r}) \psi_i(\mathbf{r}) \quad (2)$$

$$\nabla^2 V_H = 4\pi\rho \quad (3)$$

$$V_{xc} = \frac{\delta E_{xc}[\rho]}{\delta \rho} \quad (4)$$

The density ρ is the sum of the squared norm of the Kohn-Sham wave functions weighted by the Fermi-Dirac distribution. At zero temperature, the only populated states are the N lowest lying states where N is the number of electrons in the system. The Hartree potential V_H may be obtained by solving the Poisson equation, the exchange-correlation potential V_{xc} is defined as the functional derivative of the exchange-correlation energy functional E_{xc} with respect to the density, n_{FD} denotes the Fermi-Dirac distribution and the chemical potential μ is set such that the number of electrons is N . The equations are written in atomic units, we denote the Hamiltonian $\mathbf{H} = -\frac{1}{2} \nabla^2 + V_{ext} + V_H + V_{xc}$; its dimension, which corresponds to the number of real space grid points or the number of k-space grid points, is M and the eigenvalues are indexed from smallest to largest as follows $\lambda_1 < \lambda_2 < \dots < \lambda_{M-1} < \lambda_M$. Unlike the Schrödinger equation, the KS equation is non-linear since the potential depends on the density which in turn depends on the KS eigenstates. Consequently, the KS equation must be solved by cycling through Eqs.(1) to (4) until a fixed point ρ^* is found although other convergence criteria may be used.

We introduce a more flexible framework in which Eqs.(1) to (4) are expressed in terms of the single particle density matrix defined as follows

$$\rho(\mathbf{r}, \mathbf{r}') = \sum_{i=1}^{\infty} n_{FD}(\lambda_i, \mu) \psi_i^*(\mathbf{r}) \psi_i(\mathbf{r}') . \quad (5)$$

Note that the density is simply the diagonal of the single particle density matrix. The Fermi-Dirac distribution $n_{FD}(\lambda, \mu)$ decays exponentially fast for $\lambda > \mu$. It is thus reasonable to set the occupation number to zero if $n_{FD}(\lambda_i, \mu) < \epsilon$ where ϵ is some tolerance. Consequently, the number of required Kohn-Sham states L is equal to or slightly larger than the number of electrons N and it is much smaller than the linear dimension of the Hamiltonian matrix M . In other words, the single particle density matrix is a low rank matrix. It is convenient to rewrite Eq.(5) using matrix notation. To that end, we define the populated Kohn-Sham eigenspace as

$$\Psi = [\psi_1 \psi_2 \dots \psi_{L-1} \psi_L] \quad (6)$$

where ψ_i satisfies Eq.(1). The density matrix is then expressed as follows

$$\mathbf{P} = \Psi n_{FD}(\Lambda, \mu) \Psi^\dagger \quad (7)$$

where $\Lambda_{ij} = \lambda_i \delta_{ij}$. The Kohn-Sham eigenstates are expressed in terms of basis functions $\{\phi_i\}$ as done in the following equation:

$$\psi_j = \sum_i c_{ij} \phi_i , \quad (8)$$

which translates as follow in matrix notation:

$$\Psi = \Phi \mathbf{C} . \quad (9)$$

Inserting (9) in (7) we obtain

$$\mathbf{P} = \Phi \mathbf{C} n_{FD}(\Lambda, \mu) \mathbf{C}^\dagger \Phi^\dagger . \quad (10)$$

The matrix \mathbf{C} satisfies a generalized eigenvalue equation,

$$\overline{\mathbf{H}} \mathbf{C} = \overline{\mathbf{S}} \mathbf{C} \Lambda \quad (11)$$

where $\overline{\mathbf{H}} = \Phi^\dagger \mathbf{H} \Phi$, and $\overline{\mathbf{S}} = \Phi^\dagger \Phi$ is the overlap matrix. Since the overlap matrix is symmetric positive definite, it is possible to calculate its Cholesky decomposition $\overline{\mathbf{S}} = \overline{\mathbf{U}}^T \overline{\mathbf{U}}$. Eq.(11) is usually solved by reducing the generalized eigenvalue problem to a standard eigenvalue problem

$$\widehat{\mathbf{H}} \hat{\mathbf{C}} = \hat{\mathbf{C}} \Lambda \quad (12)$$

where $\widehat{\mathbf{H}} = \overline{\mathbf{U}}^{-T} \overline{\mathbf{H}} \overline{\mathbf{U}}^{-1}$ and $\hat{\mathbf{C}} = \overline{\mathbf{U}} \mathbf{C}$. Plugging Eq.(12) in (10) we obtain:

$$\mathbf{P} = \Phi \overline{\mathbf{U}}^{-1} n_{FD}(\widehat{\mathbf{H}}, \mu) \overline{\mathbf{U}}^{-T} \Phi^\dagger \quad (13)$$

It appears that as long as Ψ is a subspace of Φ , the density matrix is unchanged and so is the electronic density. Note that the chemical potential μ satisfies

$$N = \text{Tr} \left[n_{FD}(\widehat{\mathbf{H}}, \mu) \right] . \quad (14)$$

We shall refer to the quantity $\overline{\mathbf{P}}$, defined in Eq. (15) below, as the projected density matrix even though $\overline{\mathbf{P}} \neq \Phi^\dagger \mathbf{P} \Phi$.

$$\overline{\mathbf{P}} = \overline{\mathbf{U}}^{-1} n_{FD}(\widehat{\mathbf{H}}, \mu) \overline{\mathbf{U}}^{-T} . \quad (15)$$

III. PRACTICAL ALGORITHMS FOR SOLVING THE KS EQUATION

The purpose of this section is to review briefly practical algorithms used in state-of-the-art and recent DFT solvers, with the focus on elucidating where and how one may achieve speed-ups so that larger systems can be solved by KS-DFT using a modest computer.

Using the definitions introduced in section II, we now reformulate the KS equation as described in the generic Kohn-Sham solver Algorithm 1 below. Firstly, the electronic density is initialized and the dual Hamiltonian generated. The density is often initialized using the isolated atom densities but other choices, a uniform density for example, are viable in certain systems. A subspace Φ^k which spans approximately the occupied Kohn-Sham subspace Ψ is generated. Then the Hamiltonian and identity operators are projected onto the subspace Φ^k . The projected density matrix is then evaluated, typically by solving the matrix equation (11) or (12). Next, the density is obtained from the diagonal of the real space single particle density matrix and the Hamiltonian is updated by solving the Poisson equation and evaluating the exchange-correlation potential. This step is generally preceded by a mixing of the density or followed by a mixing of the potential. Finally, the convergence of the density and possibly other quantities are monitored.

Algorithm 1 Generic Kohn-Sham solver

```

procedure GENERICSOLVER( $\delta$ )
  Initialize  $\rho_0, \mathbf{H}[\rho_0]$ 
  while  $\epsilon > \delta$  or  $k < k_{max}$  do
    Compute a subspace  $\Phi^k$  which spans  $\Psi^k$ 
    Compute the projected Hamiltonian  $\bar{\mathbf{H}}^k$  and the
    overlap matrix  $\bar{\mathbf{S}}^k$ 
    Compute the projected density matrix  $\bar{\mathbf{P}}^k$ 
    Compute the density  $\rho^k(\mathbf{r}) = \mathbf{P}^k(\mathbf{r}, \mathbf{r})$ 
    Compute  $\mathbf{H} = \mathbf{H}[\rho_k]$ 
    Calculate  $\epsilon = \|\rho_k - \rho_{k-1}\|, \epsilon = \|\mathbf{H}[\rho_k] - \mathbf{H}[\rho_{k-1}]\|$ 
  return  $\rho_k$ 

```

Many currently used DFT codes fit in the framework established in Algorithm 1. They generally differ in how to calculate the subspace Φ^k and how to compute the projected density matrix $\bar{\mathbf{P}}^k$: these are the foci of the most recent algorithmic advancements and probably those to come as we shall mention later. We now discuss how particular DFT methods translate in the above picture.

The procedure executed by state-of-the-art DFT solvers is summarized in Algorithm 2 below. The eigenvectors of the Kohn-Sham Hamiltonian are calculated directly which makes the rest of the procedure simple. The Kohn-Sham states diagonalize the Hamiltonian such that $\bar{\mathbf{H}}^k$ is diagonal and the overlap matrix is the identity \mathbf{I} by virtue of the orthogonality of the eigenvectors. Calculating the Fermi-Dirac operator $n_{FD}(\bar{\mathbf{H}}^k, \mu) = n_{FD}(\Lambda^k, \mu)$ is then trivial. At zero temperature, N Kohn-Sham states are required if there are N electrons in the system ($N/2$ if there is spin degeneracy). When using thermal smearing, more states are required since the states whose energy is close to μ are fractionally occupied. As mentioned above, the density matrix has a low rank since $n_{FD}(\lambda, \mu)$ decays exponentially fast for $\lambda > \mu$. It is thus generally sufficient to compute $L = N + N_{buf}$ KS states

where N_{buf} is a modest number. These L KS states can be thought of as forming a KS basis that diagonalize the KS Hamiltonian, and this Kohn-Sham basis is deemed quasi-minimal since $L \simeq N$. The number of Kohn-Sham basis functions is typically smaller than the dimensionality of the discretized Hamiltonian by a few orders of magnitude and therefore it is advantageous to use partial diagonalization methods such as the Arnoldi algorithm or the Lanczos algorithm to compute the required eigenvectors. These algorithms are implemented in established software libraries such as ARPACK²¹ and TRLAN²² which are used by many DFT solvers.

Algorithm 2 Diagonalization Kohn-Sham solver

```

procedure DIAGSOLVER( $\delta$ )
  Initialize  $\rho_0, \mathbf{H}[\rho_0]$ 
  while  $\epsilon > \delta$  or  $k < k_{max}$  do
    Compute the occupied Kohn-Sham subspace  $\Psi^k$ 
    The projected Hamiltonian is  $\Lambda^k$  and the overlap
    matrix  $\mathbf{I}$ 
    The projected density matrix is  $n_{FD}(\Lambda^k, \mu)$ 
    Compute the density  $\rho^{k+1}(\mathbf{r}) = \mathbf{P}^k(\mathbf{r}, \mathbf{r})$ 
    Compute  $\mathbf{H} = \mathbf{H}[\rho_k]$ 
    Calculate  $\epsilon = \|\rho_k - \rho_{k-1}\|, \epsilon = \|\mathbf{H}[\rho_k] - \mathbf{H}[\rho_{k-1}]\|$ 
  return  $\rho_{k+1}$ 

```

Even sparse diagonalization techniques are very computationally demanding if a lot of occupied states must be computed. As already mentioned at the end of the last section, the Kohn-Sham states are actually not necessary and a subspace which approximately spans the occupied Kohn-Sham subspace may solve the KS equation. For example, basis sets such as atomic orbitals^{14,23-25} and Gaussian orbitals²⁶ have been used extensively in the DFT community. The procedure using a predefined basis set is summarized in Algorithm 3.

The main disadvantage of such methods is the difficulty to systematically augment the basis to improve the simulation accuracy or validate convergence. Many research groups have put forward methods to generate basis sets that can achieve a systematic convergence for most elements from H to Rn^{14,23,27-30}. Well established DFT codes using atom-centered basis functions such as SIESTA¹⁴, FHI-AIMS²³ or Gaussian²⁶ provide tested basis sets and have been used extensively by researchers to study physical systems with a variety of chemical environments. Nevertheless, some systems (e.g. certain metals, dense structures, solids with large coordination numbers) may require special treatment where new basis functions must be generated and tested. This is in contrast with the procedure described in Algorithm 2 where the Kohn-Sham states accuracy is only determined by the underlying numerical grid, which makes the convergence with respect to the basis set relatively more transparent and straightforward.

On the other hand, predefined basis sets have many computational advantages. In the scheme of Algorithm 3, the subspace needs not be updated at every step and

hence it is generated ahead of the self-consistent loop. Other quantities such as the projected kinetic energy matrix, the projected non-local ionic potential matrix and the overlap matrix are also computed ahead of the iterative process. Only the diagonal part of the Hamiltonian corresponding to the Hartree and exchange-correlation potentials has to be updated and projected onto the subspace Φ . Among other advantages, predefined basis sets are often localized by design and the projection of diagonal operators scales linearly with respect to system size and has a relatively cheap computational cost. The basis functions may also have spherical symmetry which makes it possible to perform certain integrals analytically. The main bottleneck is the computation of the projected density matrix which is usually obtained by diagonalizing the reduced projected Hamiltonian $\bar{\mathbf{H}}^k$. Whereas these basis sets are not minimal, their dimension is generally a modest multiple of the number of electrons. For systems with less than a thousand atoms or so, these methods are competitive as the matrix $\bar{\mathbf{H}} - \lambda\bar{\mathbf{S}}$ is directly invertible or diagonalizable. In larger systems, it becomes crucial that the projected Hamiltonian matrix be made as small as possible, and these methods become less competitive.

Algorithm 3 Orbital Kohn-Sham Solver

procedure ORBITALSOLVER(δ)
 Initialize $\rho_0, \mathbf{H}[\rho_0]$
while $\epsilon > \delta$ or $k < k_{max}$ **do**
 The subspace Φ is constant
 Compute the projected Hamiltonian $\bar{\mathbf{H}}^k$; the overlap matrix $\bar{\mathbf{S}}$ is constant
 Compute the projected density matrix $\bar{\mathbf{P}}^k$
 Compute the density $\rho^{k+1}(\mathbf{r}) = \mathbf{P}^k(\mathbf{r}, \mathbf{r})$
 Compute $\mathbf{H} = \mathbf{H}[\rho_k]$
 Calculate $\epsilon = \|\rho_k - \rho_{k-1}\|, \epsilon = \|\mathbf{H}[\rho_k] - \mathbf{H}[\rho_{k-1}]\|$
return ρ_{k+1}

A way around this issue is to use polynomial approximations of the Fermi-Dirac operator or other operators simulating its effect. Goedecker and Colombo have used polynomial approximations of the Fermi operator³¹; Goedecker and Teter have used Chebyshev polynomial approximations of the complementary error function³² to simulate the action of the Fermi operator; Jay *et al.* used Chebyshev-Jackson polynomial expansions of the Heaviside function³³; etc. These techniques are free of diagonalization but have bottlenecks and limitations of their own. Polynomial approximations work only at finite temperature since the Fermi-Dirac distribution is discontinuous at $T = 0$. Another disadvantage is that the degree of the polynomial must scale as $\mathcal{O}(\beta\sigma)$, where β is the inverse temperature and σ is the valence spectral width, to achieve a given accuracy. The occupied part of the energy spectrum usually spans many eV and even tens of eV whereas room temperature corresponds to an energy of 0.025 eV - these very different energy scales demand very high order expansions.

It is also possible to compute the density matrix from

single particle Green's function using the following formula:

$$\mathbf{P} = \frac{1}{2\pi i} \int_{-\infty}^{\mu} d\lambda \mathbf{G}(\lambda) \quad (16)$$

where $\mathbf{G} = (\lambda - \mathbf{H})^{-1}$, as proposed by Baroni and P. Giannozzi³⁴. Like polynomial expansions, rational approximations generally require a lot of terms to achieve a decent accuracy. The issue has been addressed by Lin and coworkers who developed a multipole expansion method which scales as $\mathcal{O}(\log(\beta\sigma) \log(\log(\beta\sigma)))$ ^{35,36}. Their method was combined with the parallel selective inversion algorithm developed by Lin *et al.*^{37,38} to perform electronic structure calculations of systems comprising thousands of atoms^{39,40}. The rational expansion of the Fermi-Dirac operator leads to an inverse intensive method which has some disadvantages compared with diagonalization techniques. It is usually more difficult to achieve a good load balance and memory distribution in inverse algorithms. Moreover, the complexity worsens as the dimensionality of the system increases from one dimension (1D) to three dimensions (3D). This is due to the filling of the matrix factors which becomes problematic in 3D problems from both memory and processing perspectives. If the inverse is available, spectrum slicing and shift-and-invert eigensolvers may also be used to find a large number of eigenvectors efficiently⁴¹.

Having briefly reviewed the various algorithms in practical KS-DFT implementations, we present a method which focuses on building a quasi-minimal basis set in the following subsection. The adaptive basis set is constructed via the Chebyshev filtering technique first applied by Zhou, Chelikowsky, Saad and coworkers^{19,20,42,43}. The workflow is presented in Algorithm 4. First, the basis set from the previous iteration is refined using Chebyshev filtering. A Chebyshev filter is an operator which is expressed as a first kind Chebyshev polynomial of the Hamiltonian matrix. It is easily evaluated as it only requires matrix-vector products. The size of the subspace is usually comparable to N and is thus significantly smaller than the size of fixed basis sets (e.g. Gaussian orbitals or atomic orbitals). It can also be systematically refined and its accuracy is only limited by the underlying grid resolution. The downside is naturally that its computation comes at a cost and the resulting subspace is generally dense such that the memory requirement scales as $\mathcal{O}(N^2)$. The scheme is generally more costly than DFT calculations using orbital bases but much lighter than the plain diagonalization schemes. RESCU implements both Algorithms 3 for atomic orbitals and Algorithm 4 for real space grids.

IV. CHEBYSHEV FILTERING

In this section, we describe the Chebyshev filtering procedure introduced by Zhou *et al.* for KS-DFT

Algorithm 4 CFSI Kohn-Sham Solver

```

procedure CFSISOLVER( $\delta$ )
  Initialize  $\rho_0, \mathbf{H}[\rho_0]$ 
  while  $\epsilon > \delta$  or  $k < k_{max}$  do
    Compute a subspace  $\Phi^k = T_n(\mathbf{H})\Phi^{k-1}$  using the
    Chebyshev filtering
    Compute the projected Hamiltonian  $\bar{\mathbf{H}}^k$  and the
    overlap matrix  $\bar{\mathbf{S}}^k$ 
    Compute the projected density matrix  $\bar{\mathbf{P}}^k$ 
    Compute the density  $\rho^{k+1}(\mathbf{r}) = \mathbf{P}^k(\mathbf{r}, \mathbf{r})$ 
    Compute  $\mathbf{H} = \mathbf{H}[\rho_k]$ 
    Calculate  $\epsilon = \|\rho_k - \rho_{k-1}\|, \epsilon = \|\mathbf{H}[\rho_k] - \mathbf{H}[\rho_{k-1}]\|$ 
  return  $\rho_{k+1}$ 

```

calculations¹⁹ and its application in the RESCU method. Already in 2006, the simulation of a Si₉₀₄₁H₁₈₆₀ nanocluster was reported in Ref.20 using this method plus computational acceleration by symmetry considerations. The technique concentrates on building a subspace Φ which spans Ψ as defined in Eq.(6). A suitable approximation for Ψ is generated and thereafter rotated toward Ψ only using Hamiltonian-subspace products. We denote the subspace at the k^{th} self-consistent iteration Φ^k . To illustrate how this works, we shall refer to Fig.1. The dimension of Φ^k is $L = N_{oc} + N_{fr}$, where N_{oc} is the number of fully occupied states ($n_{FD} > 1 - \epsilon$) and N_{fr} is the number of fractionally occupied states ($1 - \epsilon \geq n_{FD} \geq 0$).

Consider a vector $\phi \in \mathbb{R}^M$. Since the Kohn-Sham eigenvector basis is complete we can write

$$\phi = \sum a_i \psi_i. \quad (17)$$

If we apply a spectral filter $T_n(\mathbf{H})$ to ϕ , we change the composition of the vector ϕ in the following way:

$$T_n(\mathbf{H})\phi = \sum a_i T_n(\mathbf{H})\psi_i \quad (18)$$

$$= \sum a_i T_n(\lambda_i)\psi_i. \quad (19)$$

Suppose that $T_n(\lambda_i) \gg T_n(\lambda_j)$ for $i \in \{1, \dots, L\}$ and $j \in \{L+1, \dots, M\}$, then $T_n(\mathbf{H})\phi$ has a much larger overlap with Ψ than ϕ . Applying such a filter to a whole subspace Φ^k will result in a steering of Φ^k toward Ψ . For reasons evoked previously, we would like to avoid diagonalization and inversion of the Hamiltonian so that polynomial filters are an evident choice. We seek a polynomial that assumes large values in the interval $[\lambda_1, \lambda_L]$ and small values in the interval $[\lambda_{L+1}, \lambda_M]$. Many polynomials satisfy this property but the Chebyshev polynomials of the first kind (denoted T_n where n is the degree) have the minimal ∞ -norm on the interval $[-1, 1]$ among monic polynomials, and they grow exponentially in the degree n outside $[-1, 1]$ making them the ideal candidate. The 8th order Chebyshev polynomial of the first kind T_8 is plotted on a logarithmic scale in Fig. 1. The polynomial ∞ -norm is bounded by 1 in the interval $[-1, 1]$ and strictly greater than 1 outside the interval $[-1, 1]$.

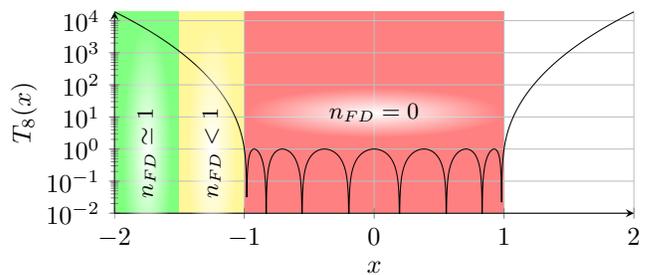


FIG. 1: 8th degree Chebyshev polynomial of the first kind $T_8(x)$ as a function of x . The y axis is logarithmic. The green region is mapped to fully occupied Kohn-Sham states. The yellow region is mapped to the fractionally occupied states (mostly unoccupied states). The $[-1, 1]$ interval is mapped to the unoccupied Kohn-Sham states. $|T_8(x)| \leq 1$ in the interval $[-1, 1]$ and hence applications of the Chebyshev filter suppress the unoccupied components. The interval $[1, \infty]$ is mapped above the spectrum and hence do not contribute.

In general, the unoccupied energies do not correspond to the interval $[-1, 1]$. In order to use T_n as a filter, we must apply an affine transformation which maps the interval $[-\infty, -1]$ to $[-\infty, \lambda_L]$ and $[-1, 1]$ to $[\lambda_{L+1}, \lambda_M]$. In this way, the components of the occupied spectrum are assuredly magnified with respect to the components of the unoccupied spectrum. This requires the knowledge of the lower and upper bounds λ_L and λ_M . The lower bound of the unoccupied spectrum λ_L can be estimated from the largest Ritz value of Φ^k which is easily obtainable. The upper bound of the spectrum can be estimated from a few step of the Lanczos algorithm. Estimates for the eigenvalue errors are derived in Templates for the solution of algebraic eigenvalue problems⁴⁴ and Zhou has studied the accuracy and robustness of a few estimators for the upper bound of the spectrum⁴⁵.

Now, consider a system of N electrons at zero temperature such that $N_{oc} = N$. The rate of convergence is roughly $T_n(\lambda_N)$ since $|T_n(\lambda_i)| \geq |T_n(\lambda_N)| \geq 1 \geq |T_n(\lambda_j)|$ for $i \in \{1, \dots, N\}$ and $j \in \{L+1, \dots, M\}$. The occupied part of the spectrum is thus magnified by at least $T_n(\lambda_N)$ at every filter application with respect to the unoccupied spectrum. However, if $\lambda_N \simeq \lambda_{L+1}$ then $|T_n(\lambda_N)| \gtrsim 1 \gtrsim |T_n(\lambda_{N+1})|$ and the convergence rate $T_n(\lambda_N) \simeq 1$ is disappointing. It is thus crucial to include enough fractionally occupied states in the subspace to separate the occupied part of the spectrum and the unoccupied part of the spectrum. The fractionally occupied Kohn-Sham states converge slowly but it does not matter since they contribute little if at all to the electronic density. The occupied, fractionally occupied and unoccupied sections of the spectrum are represented by the green, yellow and red region respectively in Fig.1. The white region does not map to any state since the largest eigenvalue is mapped below 1. A pseudocode detailing the application of a Chebyshev filter is found in Algorithm 5. There, we take advantage of the fact that Chebyshev polynomials of the first kind obey the recursive relation

$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$. Alternatively, the coefficients can be directly computed using the following formula⁴⁶

$$T_n(x) = \frac{n}{2} \sum_{k=0}^{\lfloor \frac{n}{2} \rfloor} \left(-\frac{1}{4}\right)^k 2^n \frac{(n-k-1)!}{k!(n-2k)!} x^{n-2k} \quad (20)$$

which allows the filtering implementation to use one less temporary vector.

Algorithm 5 Chebyshev Filtering

procedure CHEBFILTER($n, \lambda_M, \lambda_L, \psi_0, \mathbf{H}$)
 Compute the affine transformation parameters $e = (\lambda_M - \lambda_L)/2$ and $c = (\lambda_M + \lambda_L)/2$.
 $\psi_1 = (\mathbf{H}\psi_0 - c\psi_0)/e$
for $i = 2, \dots, n$ **do**
 $\psi_2 = 2(\mathbf{H}\psi_1 - c\psi_1)/e - \psi_0$
 $\psi_0 = \psi_1, \psi_1 = \psi_2$
return ψ_2

If N_{fr} was equal to 0, we would only need to orthonormalize the subspace Φ^k and the projected density matrix $\bar{\mathbf{P}}$ could be assumed to be the identity and therefore the density would then be trivial to evaluate. This is because the density is invariant under unitary transformations of the occupied Kohn-Sham subspace. However, using extra Kohn-Sham states is generally necessary to obtain a robust convergence for reasons we just mentioned and the density must not include the contribution from the unoccupied states. For very large systems comprising tens of thousands of electrons, certain quantities such as the total energy may not be affected significantly by including a few unoccupied states in the density. Whether this is true depends on the problem and the precision target but in general Chebyshev filtering is followed by the Rayleigh-Ritz procedure in order to populate the Kohn-Sham states correctly. The Rayleigh-Ritz procedure is summarized in Algorithm 6. It is essentially a procedure that orthonormalizes the subspace Ψ^k and computes the projected density matrix. It scales as $\mathcal{O}(N^3)$ and it is the main bottleneck in large scale computations in the Chebyshev filtering scheme. Note that it is possible to first orthonormalize Φ^k and then solve a standard eigenvalue problem or use the non-orthonormal Φ^k and solve a generalized eigenvalue problem. We have observed that the latter is generally slightly faster overall.

The Chebyshev filtering technique introduced above can be used with orthonormal discretization schemes such as finite-differencing and plane-waves. It may also be used on the projected Hamiltonian even in the case where the overlap matrix is not the identity. Implementations using Chebyshev filtering with non-orthonormal basis sets such as finite-elements^{47,48}, projector-augmented waves (PAW)¹⁸ and full-potential linearized augmented planewaves (FLAPW)⁴⁹ have been reported in the literature. We will describe how the technique also benefits basis sets methods such as NAO. As mentioned above, the NAO basis is static and localized. This leads to a sparse

Algorithm 6 Rayleigh-Ritz procedure

procedure RAYLEIGHRITZ(\mathbf{H}, Φ)
 Compute $\bar{\mathbf{H}} = \Phi^\dagger \mathbf{H} \Phi$
if $\Phi^\dagger \Phi \neq \mathbf{I}$ **then**
 Compute $\bar{\mathbf{S}} = \Phi^\dagger \Phi$
else
 $\bar{\mathbf{S}} = \mathbf{I}$
 Diagonalize $\bar{\mathbf{H}}\mathbf{C} = \bar{\mathbf{S}}\mathbf{C}\Lambda$
 Compute $\bar{\Phi} = \Phi\mathbf{C}$
 Compute $\bar{\mathbf{P}} = n_{FD}(\Lambda, \mu)$
return $\bar{\Phi}, \bar{\mathbf{P}}$

representation for $\bar{\mathbf{H}}$ and $\bar{\mathbf{S}}$ but their size is not minimal such that the eigenvalue problem of the Rayleigh-Ritz procedure is rather large. The Chebyshev filtering technique can construct and maintain an eigensubspace \mathbf{C}^k which satisfies, to a good approximation, Eq.(11). The matrix pencil $\bar{\mathbf{H}} - \lambda\bar{\mathbf{S}}$ cannot be used directly so that one must transform the generalized problem to a standard one. It is crucial to preserve to a point the sparsity of the matrix pencil since otherwise the required matrix operations do not have a significant advantage over dense diagonalization algorithm which are based on QR-decompositions. One option is to rewrite Eq.(11) as follows:

$$\bar{\mathbf{S}}^{-1}\bar{\mathbf{H}}\mathbf{C} = \mathbf{C}\Lambda \quad (21)$$

and to apply a filter $T_n(\bar{\mathbf{S}}^{-1}\bar{\mathbf{H}})$. The operator $\bar{\mathbf{S}}^{-1}\bar{\mathbf{H}}$ is no more Hermitian but in principle Eqs.(11) and (21) are equivalent and this should pose no problem. One issue is that $\bar{\mathbf{S}}^{-1}$ is generally dense and hence the matrix-vector products are computationally costly. In quasi-1D or quasi-2D systems, the Cholesky factor $\bar{\mathbf{U}}$ may still be relatively sparse depending on the system and the ordering of the overlap matrix. In this case, we suggest considering Eq.(12) instead. The reduced Hamiltonian $\hat{\bar{\mathbf{H}}} = \bar{\mathbf{U}}^{-T}\bar{\mathbf{H}}\bar{\mathbf{U}}^{-1}$ can be used in the filter to compute $\hat{\mathbf{C}}^k$ which yields \mathbf{C}^k . Each product then necessitates solving two triangular systems of equations and one symmetric matrix product. In conclusion, the efficiency of the Chebyshev filtering method vitally depends on the capacity to invert the overlap matrix in the context of NAO. There is so far no versatile and effective method to solve this issue.

V. THE PARTIAL RAYLEIGH-RITZ PROCEDURE

We begin this section by analyzing the computational complexity of the different operations performed during the self-consistent procedure in KS-DFT.

Applying a Chebyshev filter consists essentially in matrix products and the procedure scales as $\mathcal{O}(MN + N^2)$ where M is the size of the discrete Hamiltonian \mathbf{H} (i.e. the number of grid points) and N is the number of occupied Kohn-Sham states (recall $L \simeq N$).

Procedure	Scaling
Compute $\Phi := T_n(\mathbf{H})\Phi$	$\mathcal{O}(MN)$
Orthonormalize Φ	$\mathcal{O}(MN^2)$
Compute $\bar{\mathbf{H}}$ and $\bar{\mathbf{S}}$	$\mathcal{O}(MN^2)$
Solve $\bar{\mathbf{H}}\mathbf{C} = \bar{\mathbf{S}}\mathbf{C}\Lambda$	$\mathcal{O}(N^3)$
Compute $\Phi := \Phi\mathbf{C}$	$\mathcal{O}(MN^2)$

TABLE I: List of the most computationally expensive procedures in solving the Kohn-Sham equations and the associated computational complexities. M is the size of the Hamiltonian matrix and N is the number of occupied Kohn-Sham states.

The $\mathcal{O}(MN)$ scaling comes from applying the Laplacian and the $\mathcal{O}(N^2)$ term comes from applying the Kleinman-Bylander projectors in dealing with the nonlocal pseudopotentials⁵⁰.

Next, the Rayleigh-Ritz procedure scales as $\mathcal{O}(MN^2 + N^3)$ but its computational cost does not dominate until quite large system sizes as we shall demonstrate in Section VII. We further decompose the complexity of the Rayleigh-Ritz procedure in the following four operations: subspace orthonormalization, Hamiltonian (and identity) projection, eigenvalue solution and computation of the Ritz vectors. The scaling of the most computationally expensive steps in solving the Kohn-Sham equations is displayed in Table I. The non-orthonormality of Φ following the Chebyshev filtering procedure can be taken into account by the Rayleigh-Ritz procedure, and therefore the cost of orthonormalization can be absorbed in the diagonalization cost. The complexity for computing $\bar{\mathbf{H}}$ and $\bar{\mathbf{S}}$, and computing $\Phi := \Phi\mathbf{C}$ is $\mathcal{O}(MN^2)$. The former is more computationally expensive since it is a $(N \times M) \times (M \times N)$ matrix product (where $M \gg N$) whereas the latter is a $(M \times N) \times (N \times N)$ matrix product. We identify the computation of $\bar{\mathbf{H}}$ and $\bar{\mathbf{S}}$, the eigenvalue problem $\bar{\mathbf{H}}\mathbf{C} = \bar{\mathbf{S}}\mathbf{C}\Lambda$ and the computation of the Ritz vectors as the three principal bottlenecks in a large scale KS-DFT computation.

The obvious way to address the first and third bottlenecks in Table I is to construct a localized basis for Φ^k which leads to a sparse matrix representation. In atomic orbital methods, Φ^k is sparse but has known limitations in accuracy due to the inflexible nature of the basis set. In addition, even if Φ_{k-1} is localized, Φ_k is not sparse in general since the Chebyshev filtering procedure fills in the matrix. In Ref.48, Motamarri *et al.* use the localization technique introduced by Cervera⁵¹ to build a localized basis for the Chebyshev filtered subspace and the efficiency relies on the possibility to maintain a sparse basis. Their work shows that finite-elements are most appropriate to exploit the advantages of both Chebyshev filtering and basis localization. This is not possible if high-order finite-differences or plane waves are employed to compute the derivatives. When using high-order finite-differences, the density of the subspace matrix increases rapidly with the degree of the Chebyshev filter due to the far reaching high-order stencils.

Here, we address the second bottleneck in Table I by

showing that it is unnecessary to fully diagonalize $\hat{\mathbf{H}}$ in order to populate the Kohn-Sham states correctly. Suppose that N_{oc} Kohn-Sham states are fully occupied and that N_{fr} are fractionally occupied. We claim that only the N_{fr} largest eigenpairs of $\hat{\mathbf{H}}$ are actually required for the KS-DFT, and this method is named the partial Rayleigh-Ritz (pRR) procedure. To see this, consider

$$\hat{\mathbf{C}} = \begin{bmatrix} \hat{\mathbf{C}}_{oc} & \hat{\mathbf{C}}_{fr} \end{bmatrix} \quad (22)$$

$$\Lambda = \begin{bmatrix} \Lambda_{oc} & \mathbf{0} \\ \mathbf{0} & \Lambda_{fr} \end{bmatrix} \quad (23)$$

where $\hat{\mathbf{C}}$ is the matrix of eigenvectors of $\hat{\mathbf{H}}$. Then

$$n_{FD}(\Lambda) = \begin{bmatrix} \mathbf{I}_{oc} & \mathbf{0} \\ \mathbf{0} & n_{FD}(\Lambda_{fr}) \end{bmatrix} \quad (24)$$

$$= \mathbf{I} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & n_{FD}(\Lambda_{fr}) - \mathbf{I}_{frac} \end{bmatrix} \quad (25)$$

where $\mathbf{I}_{oc/fr}$ is a $N_{oc/frac} \times N_{oc/frac}$ identity matrix. Using the last equation and the fact that $\hat{\mathbf{C}}$ is unitary, Eq.(15) can be transformed into

$$\bar{\mathbf{P}} = \bar{\mathbf{U}}^{-1} \left(\mathbf{I} + \hat{\mathbf{C}}_{fr} [n_{FD}(\Lambda_{fr}) - \mathbf{I}_{fr}] \hat{\mathbf{C}}_{fr}^\dagger \right) \bar{\mathbf{U}}^{-T} \quad (26)$$

$$= \left[\bar{\mathbf{S}}^{-1} + \mathbf{C}_{fr} (n_{FD}(\Lambda_{fr}) - \mathbf{I}_{fr}) \mathbf{C}_{fr}^\dagger \right]. \quad (27)$$

From the last equation, it appears that only the N_{fr} largest Ritz-values are required to evaluate the Fermi-Dirac operator. The density matrix is the inverse of the overlap matrix plus a rank- N_{fr} correction in which the largest eigenvectors of $\hat{\mathbf{H}}$ appear. In a large system, N_{fr} is generally much smaller than N_{oc} and is more or less constant with respect to the system size. For example, our tests show that $N_{fr} \sim 8 - 32$ whereas $N_{oc} \sim 8,000 - 12,000$. Since only a few eigenvectors are required, iterative eigensolvers can be used to compute \mathbf{C}_{fr} . Like the Rayleigh-Ritz algorithm, this partial Rayleigh-Ritz procedure works whether the subspace Φ is orthonormal to begin with or not. It is generally faster to orthonormalize it inside the Rayleigh-Ritz procedure and not ahead of it. The partial Rayleigh-Ritz procedure is summarized in Algorithm 7.

Algorithm 7 Partial Rayleigh-Ritz procedure

procedure PARTIALRAYLEIGHRITZ(\mathbf{H}, Φ)

 Compute $\bar{\mathbf{H}} = \Phi^\dagger \mathbf{H} \Phi$

if $\Phi^\dagger \Phi \neq \mathbf{I}$ **then**

 Compute $\bar{\mathbf{S}} = \Phi^\dagger \Phi$

else

$\bar{\mathbf{S}} = \mathbf{I}$

 Diagonalize $\hat{\mathbf{H}}\hat{\mathbf{C}}_{fr} = \hat{\mathbf{C}}_{fr}\Lambda_{fr}$

 Compute $\Phi = \Phi\bar{\mathbf{U}}^{-1}$

 Compute $\bar{\mathbf{P}} = \mathbf{I} + \hat{\mathbf{C}}_{fr} (n_{FD}(\Lambda_{fr}) - \mathbf{I}_{fr}) \hat{\mathbf{C}}_{fr}^\dagger$

return $\Phi, \bar{\mathbf{P}}$

To close, the partial Rayleigh-Ritz algorithm differs from the traditional Rayleigh-Ritz algorithm in two

ways from a computational perspective. Firstly, in the Rayleigh-Ritz algorithm, the current Chebyshev filtered subspace is multiplied by the eigenvectors of the projected eigenvalue problems, a general matrix. In the partial Rayleigh-Ritz algorithm, the current Chebyshev filtered subspace is multiplied by the inverse of the Cholesky factor of the overlap matrix, a triangular matrix. In the case where the Chebyshev filtered subspace is already orthonormal, nothing needs to be done. This halves the computational cost associated with updating the subspace. In exact arithmetic, this could even be avoided altogether, but in practice the basis vectors of the subspace Φ^k become linearly dependent and they must be periodically orthonormalized. Whether this is necessary can be monitored by looking at the condition number of the overlap matrix. Another distinction is that only a few eigenvalues and eigenvectors of the projected eigenvalue problem are necessary to build the one-particle density matrix and obtain the electronic density for the KS-DFT. This reduces the computational complexity associated with the diagonalization from $\mathcal{O}(N^3)$ to $\sim \mathcal{O}(N^2)$.

Furthermore, we argue that the partial Rayleigh-Ritz algorithm is easier to parallelize than the Rayleigh-Ritz algorithm. Parallelizing the eigenvalue problem Eq.(11) is tantamount to re-implementing ScaLAPACK routines or some other linear algebra library for distributed memory computers, a daunting task physicists wish to avoid. In contrast, in pRR it suffices to parallelize matrix products of the type $\hat{\mathbf{H}}u$ to parallelize the eigenvalue problem in Algorithm 7. The parallel matrix-vector routine can then be used in a partial diagonalization algorithm such as LOBPCG⁵².

VI. THE RESCU IMPLEMENTATION

In this section, we report the implementation of the KS-DFT solver RESCU. The code is written in MATLAB and includes interfaces to MPI libraries, ScaLAPACK, CUDA, cuSPARSE and LibXC written in C and compiled into MEX-files.

The Kohn-Sham equation is discretized on a uniform Cartesian grid. High-order finite-differencing is used to discretize the differential operators. We generally use $\mathcal{O}(h^{16})$ stencils (49-point stencils) where h is the grid resolution. Beyond that the Vandermonde system of equations determining the stencil coefficients becomes ill-conditioned. Moreover, the entries delimiting the stencil become negligibly small and there is no gain in trying to increase the accuracy further. Matrix representations of the first and second order differential operators are generated for each coordinate. Differencing matrix-vector products take the following form

$$\frac{\partial^{(n)}}{\partial x_1^{(n)}} f(x_1^i, x_2^j, x_3^k) = \sum_l (\mathbf{D}_1^{(n)})^{i,l} f(x_1^l, x_2^j, x_3^k) \quad (28)$$

Here, the derivative is taken with respect to the first coordinate. More generally, $\mathbf{D}_j^{(n)}$ is a n^{th} order discrete differential operator operating along the j^{th} coordinate and x_i^j is the i^{th} grid point along the j^{th} coordinate. Such products can be implemented as matrix-matrix products by making the array f into a matrix in which the first dimension runs over the differentiated coordinate and the second dimension the other coordinates. N -dimensional finite-difference operators are Kronecker products of the form

$$\frac{\partial^{(n)}}{\partial x_j^{(n)}} = \bigotimes_{i=1}^{j-1} \mathbf{I}_i \otimes \mathbf{D}_j^{(n)} \bigotimes_{i=j+1}^N \mathbf{I}_i \quad (29)$$

where \mathbf{I}_i is the identity operator along the i^{th} dimension. The matrix representation of the operator defined in Eq.(29) need not be built explicitly. It suffices to perform certain manipulations on the function array, array dimension permutation and transposition for instance, to make equation 28 into a matrix product. We implement gradients and Laplacians applications as these particular Kronecker products as we found this was most efficient. We have compared it against using the multi-dimensional (sparse) Laplacian, using the stencils directly and using Fourier transforms among others. For large systems it may be advantageous to put the differencing matrices $\mathbf{D}_j^{(n)}$ in a sparse format. A sparsity of roughly 0.05 was observed to be the turning point. For example, for an $\mathcal{O}(h^{16})$ stencil it would be advantageous to use sparse differential operators if the number of points along a dimension is larger than 300.

We use a pseudopotential set generated from the Troullier-Martins scheme^{53,54} and use the Kleinman-Bylander representation⁵⁰ to model the atomic cores. A core correction is added as prescribed in Ref. 55 for elements in which the core shells overlap significantly with the valence shell. The set was developed for the NAO quantum transport package Nanocal⁵⁴ and it includes double-zeta polarized atomic orbitals. The Hartree potential of the spherically symmetric valence atomic orbital charge is added to the pseudopotentials, and hence screens long range Coulomb tails. Corrections to the Hartree potential are calculated by solving the Poisson equation for the deviation from the neutral atom density. Fourier transforms are used in periodic systems and sine transforms are used to diagonalize the finite-difference Laplacian in Dirichlet problems.

Our software implements the time-saving double-grid technique of Tomoya and Kikuchi^{56,57}. It is used to compute certain integrals, such as the projection of the Kohn-Sham states onto the Kleinman-Bylander projectors, at a lower cost. The Kohn-Sham states are typically smoother than the pseudopotentials. The idea is to express the Kohn-Sham states on a coarse grid and interpolate them on a finer grid used for the pseudopotentials when needed, saving memory and time as a result. A few exchange-correlation functionals have been implemented in MATLAB: PW92⁵⁸ (LDA), PBE⁵⁹ (GGA) and MBJ⁶⁰

(mGGA). More exchange and correlation functionals are available from LibXC⁶¹. The interface from MATLAB to LibXC is written in C and allows us to use most LDA, GGA and meta-GGA functionals implemented in the library.

The Chebyshev filtering technique originally proposed by Zhou *et al.* is significantly impeded by the initial subspace generation which requires solving for all Kohn-Sham eigenstates. In a recent paper, the authors show that starting from a random subspace and performing a few filtering steps is sufficient to obtain a suitable initial subspace⁶². In our implementation, yet another option is available: a single- or double-zeta atomic orbital basis is used as an initial subspace. The lowest energy levels of that subspace are found by partly diagonalizing the projected Hamiltonian and then a quasi-minimal initial subspace is constructed. Moreover, it is possible to reach convergence to a prescribed accuracy in the NAO basis before transposing the calculation to real space. This provides a more robust and quick convergence and alleviate significant computational cost. Our tests show that even using a single-zeta atomic orbital basis can generally give a good initial subspace.

Certain elements with d and f electrons have hard pseudopotentials and the resolution of the grid must be increased accordingly. It is generally easier to perform calculations at a low resolution since the convergence rate of non-linear accelerators generally deteriorates with respect to system size as shown by Lin and coworkers in Ref.63. We have implemented a “multi-grid” Kohn-Sham solver: it solves the equation on a coarse grid first and gradually refines the grid until some target resolution is reached. After completing the calculation at a given resolution, the code interpolates the pseudopotentials and neutral atom electronic density and interpolates the current approximation of the Kohn-Sham subspace on the refined grid. The density is then updated and a new self-consistent cycle is initiated. We implemented a few mixing schemes to accelerate the convergence of the density or the effective potential. In particular, we implemented Broyden mixing as proposed by Srivastava in Ref.64 and Johnson mixing⁶⁵ as presented by Kresse and Furthmüller in Ref. 12.

The parallelization is done with MPI and ScaLAPACK. MATLAB also naturally takes advantage of Intel’s MKL threading capabilities. The MPI-only implementation is based on the MPI+ScaLAPACK implementation, and hence we refer to ScaLAPACK in the description that follows. Our implementation uses a 2D block cyclic distribution to scatter the arrays across many processes. This is a quite general distribution scheme as far as matrices are concerned. The distribution depends on four parameters: two specifying the size of the blocks, two specifying the size of the process grid. Some examples of a 60×10 matrix shared between 4 processes are shown in Fig. 2. Fig. 2(a) shows a matrix split in 10×10 blocks distributed on a 4×1 process grid, which we refer to as a tall process grid. In contrast, a

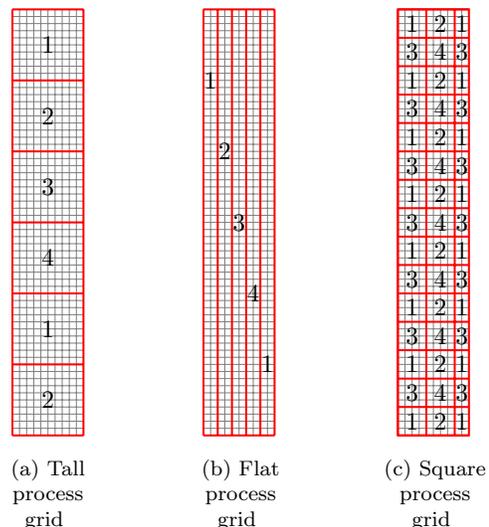


FIG. 2: The 2D block cyclic distribution is a general scheme used to distribute arrays in RESCU. The array distribution for different block sizes and process grids is depicted. The numbers indicate the rank of the process holding the submatrices.

flat process grid is distributed as shown in Fig. 2(b), where the blocks are 60×2 submatrices. Finally, a matrix distributed on a square process grid is depicted Fig. 2(c), where the blocks are 4×4 submatrices. The block size must be chosen large enough to limit communication between processes but small enough to yield a good load balance. The performance is not that sensitive to block size in our experience; a block size between 16 and 256 is usually efficient on an InfiniBand network. It is much more sensitive to the shape of the process grid however. For certain operations, the computational time can vary as much as 100%. It is more often than not favorable to take the time to redistribute optimally an array using PDGEMR2D before performing an operation. In the following description, we indicate which one of a tall, flat or square process grid is likely to yield the best performance.

The load associated with interpolating spatial variables such as potentials, densities and atomic orbitals is distributed according to a spatial partitioning which corresponds to a tall process grid (Fig. 2(a)). A good load balance is achieved since the number of real space grid points is always by far superior to the number of processes and the communication cost is negligible since interpolation is a local operation.

A significant computational cost is associated with Hamiltonian-wavefunction products. Such products occur in the Lanczos solver which is used to compute the eigenspectrum upper bound (see Section IV). The cost associated with the Lanczos solver is marginal but it can be parallelized by using ScaLAPACK with a few (e.g. four) processes or by threading the operations. Filtering the Kohn-Sham subspace is computationally expensive in

comparison. We found that parallelizing over the Kohn-Sham states is often the most efficient approach since the processes do not need to communicate during the filtering procedure. We thus distribute the subspace Φ_k array using a flat process grid (Fig.2(b)) prior to that operation. This is adequate as long as the number of electrons in the simulated system is larger than the number of processes. It is more difficult to achieve a good load balance in large systems with relatively few electrons such as ‘hollow’ molecules and 2D crystals. In this case, a parallelization scheme as described in Ref.20 will likely perform better. One other option is to use less processes but use threading to parallelize the Hamiltonian-wavefunction products.

Another computational bottleneck comes from orthonormalizing the Kohn-Sham subspace. Cholesky orthonormalization can be used, but it is performed by the more robust QR factorization routines PDGEQRF and PDORGQR if required. We have observed that matrix decomposition and diagonalization routines generally perform best on arrays split in square blocks and distributed on a square process grid (Fig.2(c)). Consequently, we redistribute the subspace as it yields a good performance in our tests. A significant cost is associated with calculating the projected Hamiltonian and the overlap matrix. A tall process grid offers the best performance for this operation when using MPI or ScaLAPACK. The Rayleigh-Ritz procedure is completed by calling PDSYEV or PDSYGVX depending on whether an orthonormalization of the subspace has taken place before. In the Rayleigh-Ritz procedure, PDSYGVX finds all the eigenvectors. If the partial Rayleigh-Ritz procedure is used, then PDSYEVX or PDSYGVX is invoked to find the few required eigenvectors. We have also modified the ARPACK²¹ interface provided with MATLAB and Knyazev’s MATLAB implementation of LOBPCG⁵² to use the parallel matrix-vector products mentioned at the end of section V.

VII. NUMERICAL TESTS

Our numerical tests are performed at McGill University’s Centre for High Performance Computing (HPC). We use nodes that consist of two Intel E5-2670 processors (8-core, 2.6 GHz, 20MB Cache) and 128 GB of DDR3 memory. The internode communication link is InfiniBand QDR. We use OpenMPI 1.8.3 and ScaLAPACK 2.0.2. In our tests, we set the number of processes equal to the number of cores and turn off threading. The timings reported below are wallclock times. The results for some of the largest systems are compiled in table II.

A. Real Space RESCU

1. Partial Rayleigh-Ritz

As mentioned earlier, the partial Rayleigh-Ritz algorithm is most competitive when the cost of the diagonalization taking place in the Rayleigh-Ritz procedure becomes significant. We evaluate the potential gain associated with diagonalization by benchmarking the ScaLAPACK routines PDSYEV, PDSYEVX and our parallel version of ARPACK, which we call ‘ScaARPACK’ here. We seek the 16 largest eigenvalues of random matrices of varying sizes. This is a typical number of buffer states required in the partial Rayleigh-Ritz algorithm in the simulation of gapped systems (e.g. Si). The time is averaged over 10 randomly generated symmetric matrices for each size. The ScaLAPACK routines use an 8×8 process grid and 32×32 blocks. ScaARPACK uses a homemade function that carries out the parallel matrix-vector products. The matrix is scattered according to a 64×1 process grid and 16×16 blocks. We have diagonalized matrices up to linear size 16,384, 32,768 or 65,536 with PDSYEV, PDSYEVX or ScaARPACK respectively. The results are plotted in Fig. 3. The scaling is $\mathcal{O}(N^{2.7})$ for the ScaLAPACK routines and almost linear for ScaARPACK. This is better than theoretical asymptotic scaling in all cases. This reflects the importance of the communication cost in the case of ScaLAPACK and the large overhead of our implementation in the case of ScaARPACK. Both ScaLAPACK routines share roughly the same scaling but the routine PDSYEVX is about an order of magnitude faster than PDSYEV since it stops when the wanted eigenpairs have been found. ScaARPACK starts winning over PDSYEVX when the matrix to be diagonalized is larger than $8,000 \times 8,000$. The speed-up for the diagonalization becomes substantial when the number of electrons is equal to or greater than 32,000 (after accounting for spin degeneracy).

2. Subspace Initialization

Next, we demonstrate the importance of the quality of the initial subspace on convergence when using Chebyshev filtering in lieu of an eigensolver. In Ref. 20, the authors suggest starting from a relatively accurate set of eigenvectors of the Hamiltonian. It was later demonstrated that using a few Chebyshev accelerated steps of the power method on a random initial subspace was sufficient to simulate many systems more efficiently⁶². We thus compare the atomic orbital (NAO) initialization against the Chebyshev filtering initialization as described in Ref.62. Single-zeta atomic orbitals are interpolated on the real space grid and the Rayleigh-Ritz procedure is performed using the resulting subspace to obtain the initial subspace. In that sense, we are performing a one shot NAO calculation and then we project the result on the real space grid. For the Chebyshev filtering initialization,

System	# of atoms (e ⁻)	N _x	N _y	N _z	Subspace (L)	Method	# cores	Time (hrs)
Si	5,832 (23,328)	140	140	140	11,672	RS	256	5.52
Al	4,000 (12,000)	110	110	110	8,044	RS	64	5.09
Al	8,788 (26,364)	141	141	141	17,596	RS	256	23.88
Cu	1,372 (15,092)	156	156	156	8,058	RS	256	9.12
DNA-H ₂ O	5,399 (14,596)	170	168	148	7,314	RS	256	9.62
Si	13,824 (55,296)	247	247	247	55,296	AO	64	6.43
Cu	5,324 (58,564)	267	267	267	95,832	AO	256	13.42

TABLE II: Some of the largest physical systems solved by KS-DFT with RESCU. The number of electrons in the system is indicated in the parentheses beside the number of atoms. The vector $[N_x, N_y, N_z]$ gives the numbers of points used along each dimension. For the AO method, $[N_x, N_y, N_z]$ is the size of the real space grid used to project the orbitals and calculate the density. It is also the grid by which the Poisson equation is solved for the Hartree potentials. The subspace dimension L corresponds to the linear dimension of the eigenvalue problem. In the method column, RS stands for real space and AO for numerical atomic orbital. The time is the total wall-clock time to converge the entire KS-DFT computation. More details about the computation are found in section VII.

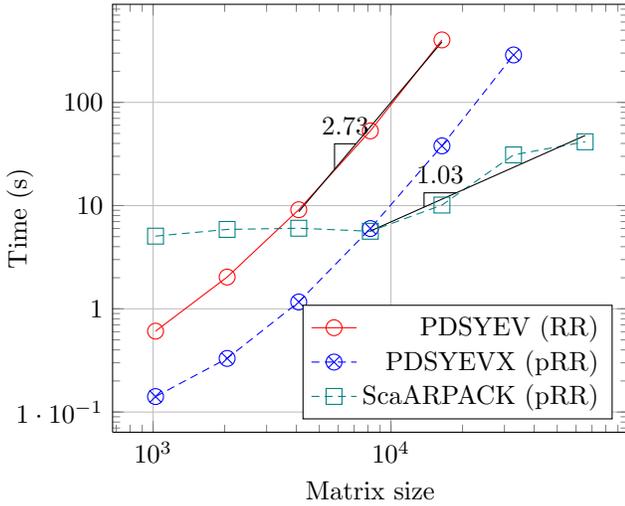


FIG. 3: Time as a function of matrix size for full and partial diagonalization (16 eigenpairs). The timings for the ScaLAPACK routines PDSYEV and PDSYEVX, used in the Rayleigh-Ritz and partial Rayleigh-Ritz algorithms, are represented by the circles and crossed circles respectively. The timings for the parallelized ARPACK function (ScaARPACK) are denoted by the squares.

4 Chebyshev filtering steps are used.

In order to compare the methods, the density of a unit cell comprising 216 silicon atoms is calculated. The number of iterations to convergence as a function of the Chebyshev filter degree (CF_{SCF}) used in the self-consistent loop is plotted in Fig. 4. We use 16 buffer states, Broyden acceleration with a mixing fraction of 0.3 and the convergence criteria are $\frac{\|\rho_k - \rho_{k-1}\|}{N} < 10^{-5}$, $\frac{\|E_k - E_{k-1}\|}{\|E_k + E_{k-1}\|} < 5 \times 10^{-6}$. The number of iterations to converge the density for $CF_0 = 16$ and $CF_{SCF} = 8$ is missing as the density did not converged within 100 iterations. $CF_{SCF} = 8$ also yields the worst performance for all initialization methods. This illustrates that the robustness is partly determined by the degree of the Chebyshev fil-

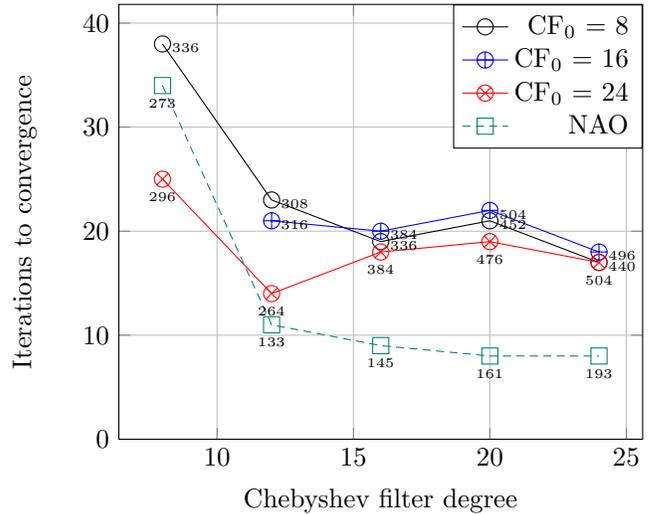


FIG. 4: Number of iterations to converge the density of a unit cell containing 216 Si atoms as a function of self-consistent Chebyshev filter degree for different initialization techniques. CF_0 stands for the degree of the Chebyshev filter used in the initialization of the subspace Φ_k (4 filtering steps are used). NAO stands for initialization from the solution of a single-zeta atomic orbital basis. The total number of Hamiltonian-wave function products is written by the data points. The circle, “plus” circle and “times” circle marks are for $CF_0 = 8$, $CF_0 = 16$ and $CF_0 = 24$ respectively. The square marks are the the NAO initialization.

ter. NAO initialization leads to a faster convergence for all values of CF_{SCF} except for $CF_{SCF} = 8$ in which case the $CF_0 = 24$ initialization leads to the smallest iteration count. However, the number of Hamiltonian-subspace products remains superior to the NAO case as indicated beside the marks in Fig. 4. Otherwise, the degree of the filter in the initialization step does not seem to impact convergence much in the present test. For CF_{SCF} larger than 12, the number of self-consistent steps to converge stagnates and the number of Hamiltonian-subspace products increases more or less linearly accordingly. Us-

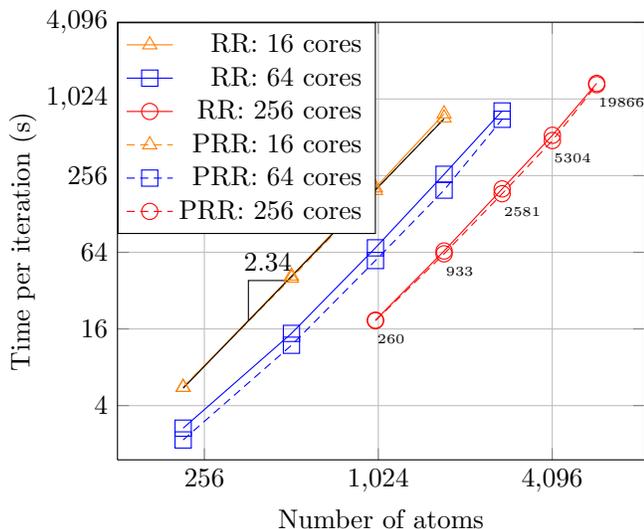


FIG. 5: Time per self-consistent step as a function of the number of Si atoms. The scaling with respect to the number of atoms is approximately $\mathcal{O}(N^{2.3})$. The triangles, squares and circles represent data points for calculations performed by 16, 64 and 256 cores respectively. The solid and dashed lines are for the Rayleigh-Ritz algorithm and the partial Rayleigh-Ritz algorithm. For the 256-core results, the numbers by the data points (red open circles) are the total wall-clock time for converging the entire KS-DFT run.

ing NAO initialization, the inflation of the number of Hamiltonian-subspace products is not as important because the number of self-consistent steps keeps decreasing although not enough to compensate the cost of a high-degree filter. The optimal filter degree remains 12 for all methods for this system. We thus recommend using NAO initialization as it may converge faster and it is appreciably cheaper than Chebyshev filtering initialization in large systems by virtue of the localized character of the atomic orbitals (see Fig. 5 and Fig. 8). The initial subspace can also be improved by using a multiple-zeta basis.

3. Bulk Silicon Supercells

Having tested various mathematical procedures, we now turn to physical systems of interest. Unit cells of silicon of varying size are simulated to test the performance of RESCU, the partial Rayleigh-Ritz algorithm and the Rayleigh-Ritz algorithm. The states are assumed to be spin-degenerate and 8 buffer states are included to separate the occupied and unoccupied spectra as explained in Section IV. Kleinmann-Bylander projectors up to angular momentum $L = 1$ are used. The grid spacing lower bound is set to 0.66 Bohr which corresponds to an energy cutoff of 300 eV. The differential operators are generated using 16th order stencils. The exchange-correlation terms are computed using Perdew and Wang’s version of LDA⁵⁸

as implemented in LibXC. We use 15 steps of the Lanczos algorithm to calculate the eigenspectrum upper bound and a Chebyshev filter of degree 16. Broyden mixing with a mixing fraction of 0.3 and a history of 20 is employed to accelerate convergence. The convergence criteria are again $\frac{\|\rho_k - \rho_{k-1}\|}{N} < 10^{-5}$, $\frac{\|E_k - E_{k-1}\|}{\|E_k + E_{k-1}\|} < 5 \times 10^{-6}$. For the partial Rayleigh-Ritz benchmark, we use the partial diagonalization capabilities of ScaLAPACK (PDSYGVX) and we find 12 hole eigenvalues (i.e. 8 buffer states plus 4 occupied states). Many of these parameters can be optimized further to yield a better performance: the parameters used here are by no means optimal - we simply used sensible values based on experience - but they already give impressive performance. Although this is suboptimal, we also keep the parameters constant while varying the number of processes to get a consistent and fair comparison.

In Fig. 5, the time per self-consistent step is plotted as a function of the number of atoms. The calculations were carried out using 16, 64 and 256 cores. We could go up to 5,832 Si atoms before running out of memory (the next cubic Si supercell contains 8,000 Si atoms). The total time to convergence (in seconds) for the 256-core runs is written by the data points in Fig.5. We observe that, at those sizes, the partial Rayleigh-Ritz algorithm leads to marginal gains. There are many reasons to this. Most importantly, the pRR gains are masked by the large cost of projecting the Hamiltonian into the filtered subspace and computing the overlap matrix. Hence, even though pRR is always faster (see Fig.3), it gives marginal gains for this particular test. We shall discuss this issue in more details in section VIII. We stress that pRR is measured against the highly efficient parallel linear algebra library ScaLAPACK. In the case where no such library is available, pRR provides a convenient way to parallelize the computation of the projected density matrix and can lead to substantial time savings in smaller physical systems. Finally, we note that the computational time scales consistently as $\mathcal{O}(N^{2.3})$ for all processor counts. We also highlight that the parallelization efficiency approaches 100% as the number of atoms in the system increases.

4. Bulk Aluminium Supercells

We now turn to a metallic system for our second test: aluminium supercells. Again, we perform the calculations using 16, 64 and 256 cores. The number of valence electrons per atom is 3 and the system is assumed to be spin-degenerate. The spatial resolution is 0.7 a.u. and the differential operator accuracy is $\mathcal{O}(h^{16})$. We compute the states occupancies using the Fermi-Dirac distribution with a temperature of 1,000K. The computation is performed within the LDA using the routines XC_LDA_X and XC_LDA_C-PW from LibXC as in the previous benchmark. We chose a Chebyshev filter of degree 16. The number of buffer states varies with respect to system size. This is necessary to open a sizeable gap

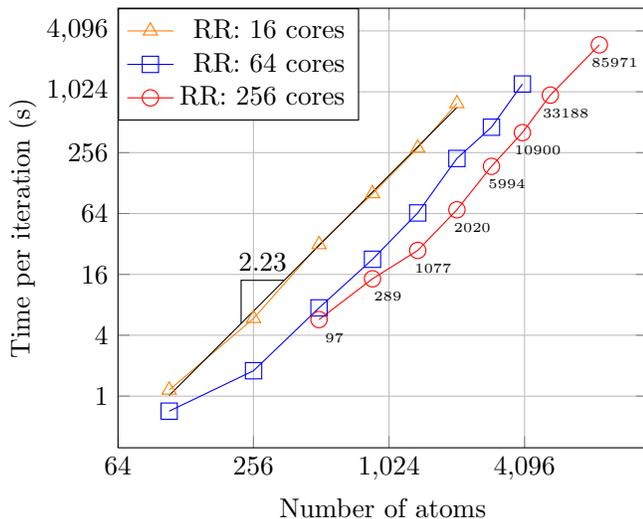


FIG. 6: Time per self-consistent step as a function of the number of Al atoms. The scaling with respect to the number of atoms is approximately $\mathcal{O}(N^2)$. For the 256-core results, the numbers by the data points (red open circles) are the total wall-clock time for converging the entire KS-DFT run.

between the occupied and unoccupied states as explained in section IV. In the present test, we set the number of extra states to 10%-30% the number of ions. Finally, we used Johnson-Kerker mixing with a mixing fraction of 0.5 and a minimal mixing fraction of 0.05. The time per self-consistent step as a function of the number of Al atoms is displayed in Fig. 6. The computational cost per self-consistent step scales almost quadratically ($\mathcal{O}(N^{2.2})$) with respect to the number of atoms. With 64 cores, the largest supercell simulated contained 4,000 Al atoms and the density and total energy where converged to one part in 10^5 in slightly over 18,000 seconds using 256 cores. A supercell containing 8,788 Al atoms is handled with 256 cores. After 29 iterations and almost 24 hours of computation, the residuals are the following: $\frac{\|\rho_k - \rho_{k-1}\|}{N} \simeq 10^{-4}$, $\frac{\|E_k - E_{k-1}\|}{\|E_k + E_{k-1}\|} \simeq 5 \times 10^{-8}$. The density convergence criterion is not as restrictive as one used in the other benchmarks. We note, however, that it is already constrained enough for band structure calculation purposes as shown in Fig.9(c) below.

5. Bulk Copper Supercells

We briefly report input parameters and results for another metal test: copper supercells. In this test we used 256 cores. The number of valence electrons per atom is 11 and the system is assumed to be spin-degenerate. The spatial resolution is 0.3 a.u. and the differential operator accuracy is $\mathcal{O}(h^{16})$. We compute the states occupancies using the Fermi-Dirac distribution with a temperature of 100K. The computation is performed within the LDA using the routines XC.LDA.X and XC.LDA.C.PW from

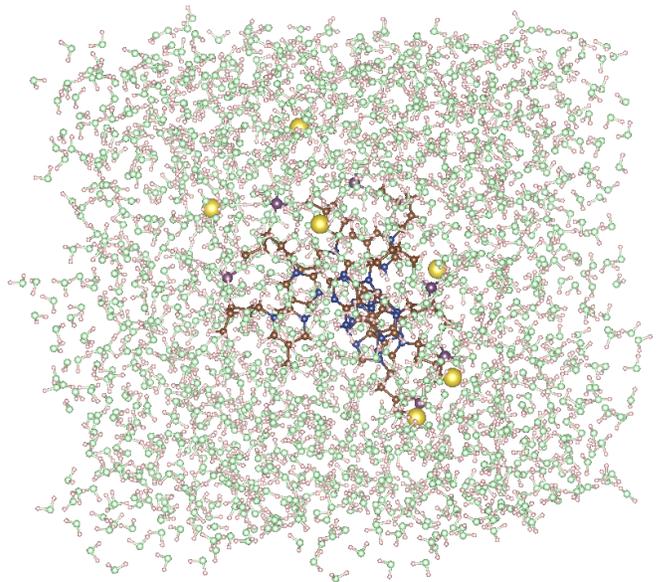


FIG. 7: DNA molecule solvated in 1,713 water molecules.

LibXC as in the previous benchmark. We chose a Chebyshev filter of degree 16. In the present test, we set the number of extra states to 50% the number of ions. Finally, we used Johnson-Kerker mixing with a mixing fraction of 0.25 and a minimal mixing fraction of 0.1. The largest supercell simulated contained 1,372 Cu atoms and the density and total energy where converged to one part in 10^5 in 32,843 seconds using 256 cores as reported in table II.

6. DNA molecule in water

As another example of testing on physical systems, RESCU is applied to calculate the electronic structure of a solvated DNA structure (5'-AAAA-3') which is a completely disordered system. The initial structure is obtained from minimizing the structure with the molecular modeling package AMBER 11. The DNA structure is initially charge neutralized with counterions by 6 Na^+ and solvated with 1,713 TIP3P water molecules⁶⁶. The system is depicted in Fig. 7. The tetragonal simulation domain has dimensions $44.5 \times 44.4 \times 39.1 \text{ \AA}^3$ and periodic boundary conditions. We use a resolution of 0.25 Bohr. A total of 5,399 atoms (14,596 electrons) were simulated using 256 cores in the setup described above. We use the LibXC XC_GGA_X_OPTPBE_VDW exchange functional and XC_GGA_C_OP_B88 correlation functional. The Laplacian is discretized using $\mathcal{O}(h^{16})$ stencils. We used a 16th order Chebyshev filter and 16 buffer states. The convergence criterion is $\frac{\|\rho_k - \rho_{k-1}\|}{N} < 10^{-5}$ and we mix the density using the Pulay method with a mixing fraction of 0.1. We initialize the subspace using a single-zeta atomic orbital basis set with an angular momentum cutoff $L = 1$. The electronic density converged in 20

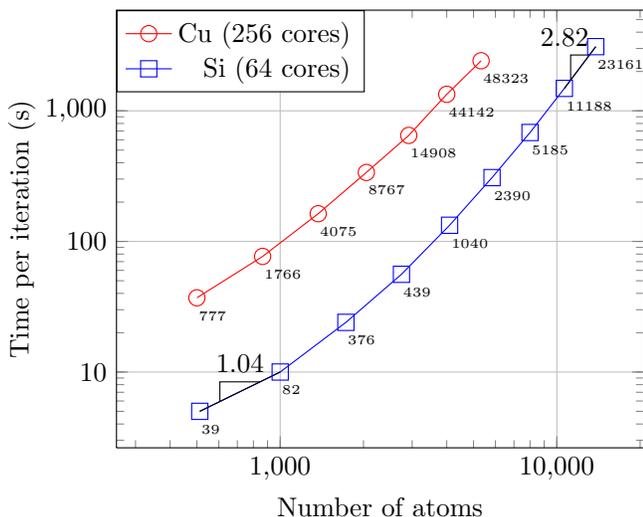


FIG. 8: Time per self-consistent step as a function of the number atoms. The circles are for the Cu benchmark which is performed using a double-zeta atomic orbital basis with angular momentum cutoff $L = 2$ and 256 cores, each having 8GB of DDR3 memory. The squares are for the Si benchmark which is performed using a single-zeta atomic orbital basis with angular momentum cutoff $L = 1$ and 64 cores, each having 4GB of DDR3 memory. The scaling with respect to the number of atoms is linear $\mathcal{O}(N^{1.04})$ for smaller systems and gradually ramps up to cubic $\mathcal{O}(N^{2.82})$. The numbers by each data point is the total wall-clock time for converging the entire KS-DFT run.

steps which took a total of 34,638 seconds for an average time of 1,732 seconds per self-consistent step. Finally, we find that the gap between the highest occupied molecular orbital (HOMO) and the lowest unoccupied molecular orbital (LUMO) shrinks to 0.6eV - the gap for the isolated DNA structure without water is 2.0eV - which indicates that the solvent plays an important role in the optical properties of wet DNA⁶⁷. We shall present the comparison in a forthcoming article⁶⁸.

B. Atomic Orbital RESCU

To demonstrate the capabilities of the atomic orbital method implementation in RESCU, we perform the same benchmark as above (bulk Si supercells) using numerical atomic orbitals. We use the same processors (E5-2670) equipped with half the memory this time (4GB/core). We perform the benchmark with 64 cores only. We also use a real space resolution of 0.5 a.u. A single-zeta basis with angular momentum cutoff $L = 1$ was used such that there are 4 atomic orbitals per atom. The results are plotted as blue squares in Fig. 8. The largest simulated Si supercell comprises 13,824 Si atoms and the KS-DFT was converged in 23,161 seconds. Even at that size, we did not run into any memory issue since the atomic orbital basis yields sparse matrices. For relatively modest su-

percells, RESCU scales linearly. The scaling deteriorates as the supercell size grows since the eigenvalue problem accounts for a larger and larger proportion of the computational cost. The scaling gradually becomes cubic since we are treating the projected eigenvalue problem as a dense eigenvalue problem. Many methods evoked in Section III could improve the efficiency of NAO calculations further. Since diagonalization performance is so important in this test, we mention that a square processor grid and 64×64 blocks are used in the block cyclic distribution of the Hamiltonian and overlap matrices.

We have also performed NAO computations for bulk copper supercells. For this benchmark, we use 256 cores with 8GB of memory per core. We use a real space resolution of 0.25 a.u and a double-zeta basis with angular momentum cutoff $L = 2$ (18 atomic orbitals per atom). We have simulated supercells including up to 5,324 Cu atoms (58,564 electrons). The density and total energy were converged in roughly 12 hours (33 iterations). The time per self-consistent step as a function of the number of Cu atoms is indicated by the red circles in Fig. 8. The number of atomic orbitals per atom is 4.5 times larger than in the Si benchmark and the number of cores 4 times larger. Consequently, the time per step for the Cu system is roughly an order of magnitude larger than the time per step for the Si system. The time per iteration scales similarly, i.e. it goes from linear in the 1,000 atom range to cubic in the 10,000 atoms range. The total time scales worse here as larger (metallic) systems tend to take more iterations to converge.

C. Accuracy

To verify the accuracy of the electronic structures calculated in our benchmark, we calculated the band structures of Al, Si and Cu using the VASP package (albeit using a primitive cell) and compared it with the band structures calculated from the density of the largest supercell simulated. The overlaid RESCU and VASP band structures are displayed in Fig.9. The agreement is impressive considering that these methods are different in many respects. In particular, the PAW method is used in VASP and pseudopotentials are used in RESCU. We have also calculated band structures of compounds by both RESCU and VASP, results for the semiconductor GaAs and the insulator MgO are presented in Fig. 10. Again, the agreement between the two methods is excellent. This demonstrates the precision of the RESCU method for band structure calculations.

A systematic approach to comparing DFT codes has been introduced by Lejaeghere et al.⁶⁹ in 2014. Its essence is to calculate the Δ -functional (see Eq.30 below) for the total energy versus volume (E vs V) equation of states (EOS) of elemental crystals. The equilibrium volume V_0 , the bulk modulus B_0 and the derivative of the bulk modulus B_1 can be extracted from the EOS using a third-order Birch-Murnaghan fit⁷⁰. The value of Δ is

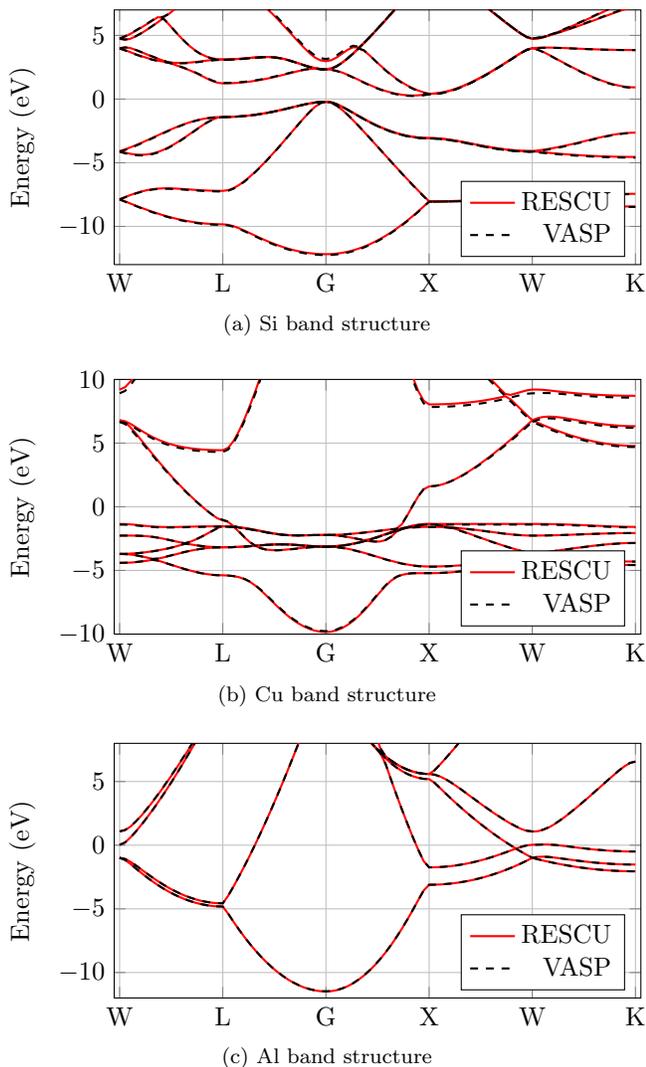


FIG. 9: Comparisons of band structures obtained by VASP and RESCU. The band structures agree to a high accuracy indicating that the resolution used in our benchmark was sufficient for the purpose of band structure calculations.

thus a sensible measure of the agreement of the structural and mechanical predictions of two DFT solvers. The Δ -functional is defined as follows

$$\Delta(E^a, E^b) = \frac{1}{|\Omega|} \int_{\Omega} dV (E^a(V) - E^b(V)) \quad (30)$$

where Ω is an interval, $E^a(V)$ is the E vs V EOS for code a and $E^b(V)$ is the E vs V EOS for code b . The interval Ω is chosen as $[0.94(V_0^a + V_0^b)/2, 1.06(V_0^a + V_0^b)/2]$.

We have calculated the value of $\Delta(E^{RESCU}, E^{WIEN2k})$ for a number of elements, where E^{RESCU} is the EOS calculated by RESCU and E^{WIEN2k} is the EOS calculated by WIEN2k⁹. The WIEN2k EOS are provided in the supplemental content of Ref. 69. In RESCU, we used a grid spacing of 0.14 Bohr and 6750/ N k-points for N -atoms unit cells.

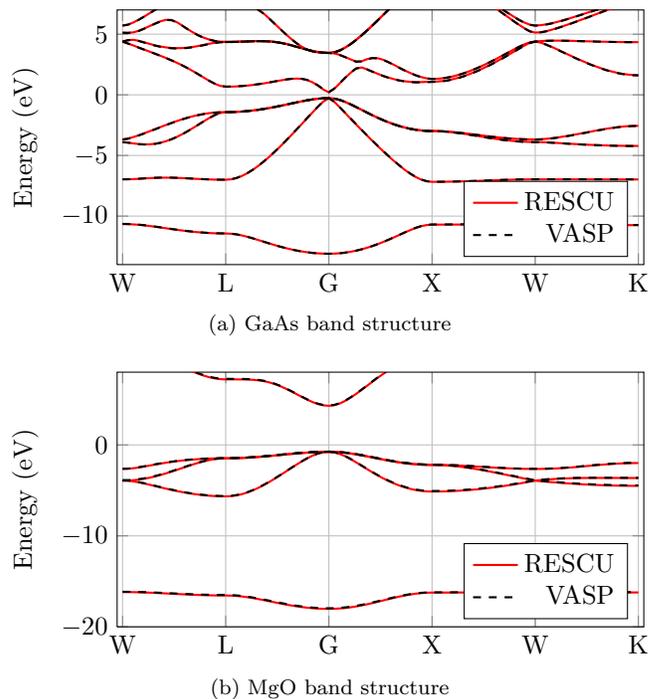


FIG. 10: Comparisons of the band structures of the GaAs and MgO compounds obtained by VASP and RESCU.

The charge convergence criterion is $10^{-5}e$ per valence electron and the energy convergence criterion is 10^{-5} Hartree per valence electron. A Fermi-Dirac smearing with a temperature of 800 K is used. The computation is performed within the GGA using the routines XC_GGA_X_PBE and XC_GGA_C_PBE from LibXC. Again, in the RESCU calculation the atomic cores are modeled by Troullier-Martins pseudopotentials^{53,54}. The obtained Δ values are listed in Table III and the differences between the EOS are quite reasonable in all cases. These Δ values are comparable to those obtained with the electronic structure package AbInit (using the Troullier-Martins pseudopotential) and WIEN2k⁷¹. Many codes include all elements from H to Rn except those between lanthanum and ytterbium in their test, but we leave such an exhaustive test to a future opportunity.

We end this subsection by emphasizing that, while using a more efficient solution process to solve the KS-DFT equation, there was no accuracy-degrading approximation in RESCU and the accuracy tests presented here strongly demonstrate its quality.

VIII. FURTHER DISCUSSIONS

Having demonstrated the power of RESCU by solving the KS equation for thousands of atoms - both insulating and metallic, both ordered and disordered, and on a modest computer cluster - we now discuss the princi-

Element	Δ (meV/atom)	Element	Δ (meV/atom)
H	0.864	Ca	2.013
Be	7.080	Cu	7.758
Mg	1.392	Rh	18.627
Al	0.434	Pd	15.581
Si	3.608	Ag	11.286
P	7.291	Cs	1.082
S	6.830		

TABLE III: The $\Delta(E^{RESCU}, E^{WIEN2k})$ values for thirteen elements.

pal bottlenecks of the real space method in RESCU. To this end, we use the results of the Si benchmark in Section VII for the discussion. We have plotted the time taken by the computationally intensive operations of Table I as a function of the number of atoms in Fig. 11: they are the Chebyshev filtering, the Hamiltonian projection onto the filtered subspace, the diagonalization of the projected pencil, the orthonormalization or/and computation of the Ritz vectors, and we add to this list the residual timing (ROC) which includes the remaining time. The timings for one self-consistent step performed by 256 cores are reported in Fig. 11. Both RR and pRR algorithms yield similar timings in all parts of the computation except for the orthonormalization. In solving the generalized eigenvalue problem, the eigensolver computes the Cholesky factor of the overlap matrix to reduce the generalized eigenvalue problem to a standard form. In the partial Rayleigh-Ritz algorithm, this factor is reused to orthonormalize the filtered subspace. In the standard Rayleigh-Ritz algorithm, there is no point in doing so since the eigenvectors for the generalized eigenvalue problem will directly provide the Ritz vectors which are orthonormal by construction. But the latter matrix is a general one whereas the former is triangular, and hence the factor of two speed up observed in Fig. 11. The diagonalization is faster in the partial Rayleigh-Ritz procedure, but it is not too significant as it is as large as the efficiency fluctuations observed in our computation tests. There are a few explanations to this. Firstly, according to the benchmark in Fig.3, direct diagonalization is relatively cheap for matrices smaller than $10,000 \times 10,000$ which is about the size of the matrices in our largest system. Secondly, in a relatively simple system of bulk silicon, the projected matrix pencil $\bar{\mathbf{H}} - \lambda \bar{\mathbf{S}}$ is more easily diagonalizable than the random matrices used in the benchmark described above since it is closer to being diagonal. Moreover, it becomes closer and closer to being “diagonal” as the electronic density - and the Kohn-Sham invariant subspace - approaches its fixed point. The scaling of the residual timing appears linear and that of the Chebyshev filtering is quadratic. The main bottlenecks are the projection and the orthonormalization which scale almost cubically. This originates from the complexity of matrix-matrix multiplication which is $\mathcal{O}(N^3)$ or $\mathcal{O}(N^{2.8})$ depending on matrix size and implementation.

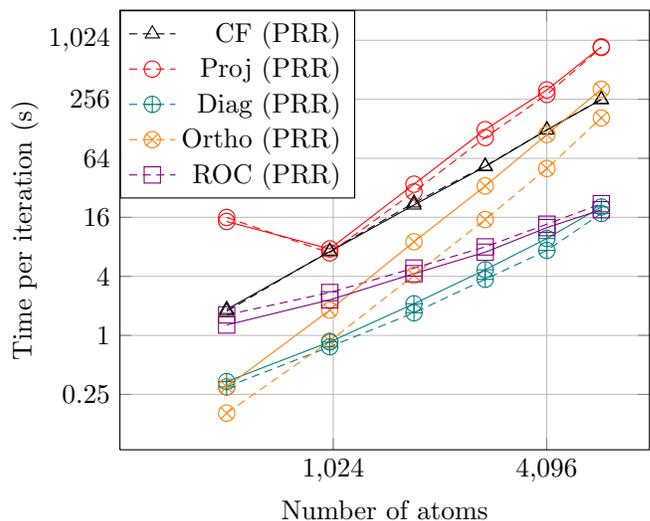


FIG. 11: Time per self-consistent step as a function of the number of Si atoms using 256 processors. The full line link data points for the Rayleigh-Ritz algorithm and the dashed lines link data points for the partial version.

The main bottleneck comes about because of the massive amount of data required to encode the subspace Φ_k . In order to improve the method described in this work, it is crucial to achieve some sort of subspace compression. One way to address this issue is to calculate a localized basis for the filtered subspace as done by Motamarri *et al.* in Ref. 48, where the authors show that localizing the basis is key to achieving a subquadratic computational scaling, in particular when evaluating the Hartree-Fock exchange functional. Tangentially, subspace compression is crucially needed because the memory requirement scales as $\mathcal{O}(N^2)$. Indeed, the RESCU method is so far more limited by the memory requirement than the computational requirement as evidenced in the Si benchmark. Unfortunately, it appears that, for the reported tests, the subspace “resists” localization as it approaches convergence and performing part of the computation with a dense subspace is still required. In a similar spirit, RESCU can use multiple-zeta numerical atomic orbital bases to solve the Kohn-Sham equations. If the memory requirements are not too severe, the NAO solutions are projected into real space and the energy may be further minimized using Chebyshev filtering. In summary, future research should focus on the following points: a vector space spanning the Kohn-Sham occupied subspace must be computed efficiently, storing such a vector space should require $\mathcal{O}(N)$ memory (the data must be compressible somehow) and the projected density matrix should be efficiently computable. In any event, even with the existing bottlenecks, RESCU has already achieved impressive computational efficiency in solving the KS-DFT problem.

IX. CONCLUSIONS

In this work, we have presented a powerful Kohn-Sham DFT solver, RESCU. The goal of the RESCU method is to predict electronic structure properties of systems comprising many thousands of atoms using moderate computer resources. The computational efficiency is gained by exploiting four routes. First, in real space, Chebyshev filtering is used to expedite the computation of an invariant Kohn-Sham subspace in large systems. This approach essentially exploits the fact that when the Hamiltonian is not yet converged, one does not need to solve the KS equation extremely accurately. Second, we developed a NAO-based method to efficiently generate a good initial subspace which is necessary in the Chebyshev filtering paradigm. Third, by judiciously analyzing various parts of the KS-DFT solution procedure, RESCU gains efficiency by delaying the $O(N^3)$ scaling to large N ; and our tests showed that RESCU scales as $O(N^{2.3})$ up to the several thousand atoms level. Fourth, RESCU gains efficiency by various numerical mathematics and, in particular, we introduced the partial Rayleigh-Ritz algorithm and showed it leads to efficiency gains for systems comprising more than 10,000 electrons. The RESCU code is implemented in MATLAB such that it provides a convenient prototyping and development environment. It is also easily installed on many platforms and architectures. Finally, we mention in passing that we have also implemented total energy and force calculation methods into RESCU, but we reserve the discussion of structural

relaxation using RESCU for the future.

We demonstrated that the RESCU method could perform large scale KS-DFT computations using computer resources ranging from 16 to 256 cores. At the 5,000-15,000 atoms level, there are many important material physics problems to be investigated and we wish to report them in the near future. From the method development point of view, to deal with even larger systems, we find it is essential to compress the Kohn-Sham subspace to achieve better computational effectiveness and to relieve computer memory requirements. Sparse matrix representations can alleviate the problem to some extent but do not solve it entirely. We think the solution may lie in the hierarchical matrix approximations which are data-sparse structures for dense matrices. We wish to present these efforts in the future. Finally, it is tempting and interesting to extrapolate the RESCU ability to much larger systems - using current supercomputers. However, we caution that a naive extrapolation may not work since for much larger N , the computational burden shifts toward the parts of the algorithm that scale as $O(N^3)$. We believe these are important topics of future research.

Acknowledgement. We gratefully acknowledge financial support by NSERC of Canada and FQRNT of Quebec (H.G.). We thank Dr. Eric Zhu and Dr. Lei Liu for their help on pseudopotentials and LCAO basis sets; Dr. Langhui Wan and Dr. Kevin Zhu for helping us on computational issues related to ScaLAPACK. We thank McGill HPC, Calcul Québec and Compute Canada for computation facilities which made this work possible.

* Electronic address: Email:vincentm@physics.mcgill.ca

† Electronic address: Email:zhanglei@physics.mcgill.ca

¹ P. Hohenberg and W. Kohn, *Physical Review* **136**, 864 (1964).

² L. H. Thomas, *Proceedings of the Cambridge Philosophical Society* **23**, 542 (1927).

³ E. Fermi, *Rend. Accad. Naz. Lincei* **6**, 602 (1927).

⁴ P. A. M. Dirac, *Proceedings of the Cambridge Philosophical Society* **26**, 376 (1930).

⁵ C. Weizscker, *Zeitschrift fr Physik* **96**, 431 (1935).

⁶ M. Levy, *Proceedings of the National Academy of Sciences* **76**, 6062 (1979).

⁷ W. Kohn and L. J. Sham, *Physical Review* **140**, 1133 (1965).

⁸ P. Blaha, K. Schwarz, P. Sorantin, and S. Trickey, *Computer Physics Communications* **59**, 399 (1990).

⁹ K. Schwarz and P. Blaha, *Computational Materials Science* **28**, 259 (2003).

¹⁰ W. E. Pickett, *Computer Physics Reports* **9**, 115 (1989).

¹¹ M. Fuchs and M. Scheffler, *Computer Physics Communications* **119**, 67 (1999).

¹² G. Kresse and J. Furthmüller, *Phys. Rev. B* **54**, 11169 (1996).

¹³ G. Kresse and J. Furthmüller, *Computational Materials Science* **6**, 15 (1996).

¹⁴ J. M. Soler *et al.*, *Journal of Physics: Condensed Matter*

14, 2745 (2002).

¹⁵ X. Gonze *et al.*, *Computer Physics Communications* **180**, 2582 (2009).

¹⁶ F. Bottin, S. Leroux, A. Knyazev, and G. Zrah, *Computational Materials Science* **42**, 329 (2008).

¹⁷ L. Lin, A. Garca, G. Huhs, and C. Yang, *Journal of Physics: Condensed Matter* **26**, 305503 (2014).

¹⁸ A. Levitt and M. Torrent, *Computer Physics Communications* **187**, 98 (2015).

¹⁹ Y. Zhou, Y. Saad, M. L. Tiago, and J. R. Chelikowsky, *Journal of Computational Physics* **219**, 172 (2006).

²⁰ Y. Zhou, Y. Saad, M. L. Tiago, and J. R. Chelikowsky, *Phys. Rev. E* **74**, 066704 (2006).

²¹ R. Lehoucq, D. Sorensen, and C. Yang, *ARPACK Users' Guide*, 1998.

²² K. Wu, A. Canning, H. Simon, and L.-W. Wang, *Journal of Computational Physics* **154**, 156 (1999).

²³ V. Blum *et al.*, *Computer Physics Communications* **180**, 2175 (2009).

²⁴ D. R. Bowler, R. Choudhury, M. J. Gillan, and T. Miyazaki, *physica status solidi (b)* **243**, 989 (2006).

²⁵ J. Enkovaara *et al.*, *Journal of Physics: Condensed Matter* **22**, 253202 (2010).

²⁶ M. J. Frisch *et al.*, *Gaussian09 Revision D.01*, gaussian Inc. Wallingford CT 2009.

²⁷ J. Junquera, O. Paz, D. Sánchez-Portal, and E. Artacho,

- Phys. Rev. B **64**, 235111 (2001).
- ²⁸ T. Ozaki and H. Kino, Phys. Rev. B **69**, 195113 (2004).
- ²⁹ G. A. Petersson, S. Zhong, J. A. Montgomery, and M. J. Frisch, The Journal of Chemical Physics **118**, 1101 (2003).
- ³⁰ A. Schfer, H. Horn, and R. Ahlrichs, The Journal of Chemical Physics **97**, 2571 (1992).
- ³¹ S. Goedecker and L. Colombo, Phys. Rev. Lett. **73**, 122 (1994).
- ³² S. Goedecker and M. Teter, Phys. Rev. B **51**, 9455 (1995).
- ³³ L. O. Jay, H. Kim, Y. Saad, and J. R. Chelikowsky, Computer Physics Communications **118**, 21 (1999).
- ³⁴ S. Baroni and P. Giannozzi, Europhys. Lett. **17**, 547 (1992).
- ³⁵ L. Lin, J. Lu, L. Ying, and W. E, Chinese Annals of Mathematics, Series B **30**, 729 (2009).
- ³⁶ L. Lin, J. Lu, R. Car, and W. E, Phys. Rev. B **79**, 115133 (2009).
- ³⁷ L. Lin *et al.*, ACM Trans. Math. Softw. **37**, 40:1 (2011).
- ³⁸ L. Lin *et al.*, SIAM Journal on Scientific Computing **33**, 1329 (2011).
- ³⁹ L. Lin, M. Chen, C. Yang, and L. He, Journal of Physics: Condensed Matter **25**, 295501 (2013).
- ⁴⁰ L. Lin, A. Garca, G. Huhs, and C. Yang, Journal of Physics: Condensed Matter **26**, 305503 (2014).
- ⁴¹ H. M. Aktulga *et al.*, Parallel Computing **40**, 195 (2014).
- ⁴² Y. Saad *et al.*, physica status solidi (b) **243**, 2188 (2006).
- ⁴³ G. Schofield, J. Chelikowsky, and Y. Saad, Using Chebyshev-Filtered Subspace Iteration and Windowing Methods to Solve the Kohn-Sham Problem, 2012.
- ⁴⁴ J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, in *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, edited by Z. Bai (Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000).
- ⁴⁵ Y. Zhou and R.-C. Li, Linear Algebra and its Applications **435**, 480 (2011).
- ⁴⁶ D. Zwillinger and C. press inc, *CRC standard mathematical tables and formulae* (CRC Press, Boca Raton, Florida, 1996).
- ⁴⁷ P. Motamarri *et al.*, Journal of Computational Physics **253**, 308 (2013).
- ⁴⁸ P. Motamarri and V. Gavini, Phys. Rev. B **90**, 115127 (2014).
- ⁴⁹ M. Berljafa and E. Di Napoli, in *Parallel Processing and Applied Mathematics, Lecture Notes in Computer Science*, edited by R. Wyrzykowski, J. Dongarra, K. Karczewski, and J. Waniewski (Springer Berlin Heidelberg, ADDRESS, 2014), pp. 395–406.
- ⁵⁰ L. Kleinman and D. M. Bylander, Phys. Rev. Lett. **48**, 1425 (1982).
- ⁵¹ C. J. García-Cervera, J. Lu, Y. Xuan, and W. E, Phys. Rev. B **79**, 115110 (2009).
- ⁵² A. Knyazev, SIAM Journal on Scientific Computing **23**, 517 (2001).
- ⁵³ N. Troullier and J. L. Martins, Phys. Rev. B **43**, 1993 (1991).
- ⁵⁴ NanoAcademic Technologies, <http://www.nanoacademic.ca/>.
- ⁵⁵ S. G. Louie, S. Froyen, and M. L. Cohen, Phys. Rev. B **26**, 1738 (1982).
- ⁵⁶ T. Ono and K. Hirose, Phys. Rev. Lett. **82**, 5016 (1999).
- ⁵⁷ T. Ono and K. Hirose, Phys. Rev. B **72**, 085115 (2005).
- ⁵⁸ J. P. Perdew and Y. Wang, Phys. Rev. B **45**, 13244 (1992).
- ⁵⁹ J. P. Perdew, K. Burke, and M. Ernzerhof, Phys. Rev. Lett. **77**, 3865 (1996).
- ⁶⁰ F. Tran and P. Blaha, Phys. Rev. Lett. **102**, 226401 (2009).
- ⁶¹ M. A. Marques, M. J. Oliveira, and T. Burnus, Computer Physics Communications **183**, 2272 (2012).
- ⁶² Y. Zhou, J. R. Chelikowsky, and Y. Saad, Journal of Computational Physics **274**, 770 (2014).
- ⁶³ L. Lin and C. Yang, SIAM Journal on Scientific Computing **35**, S277 (2013).
- ⁶⁴ G. P. Srivastava, Journal of Physics A: Mathematical and General **17**, L317 (1984).
- ⁶⁵ D. D. Johnson, Phys. Rev. B **38**, 12807 (1988).
- ⁶⁶ W. L. Jorgensen *et al.*, The Journal of Chemical Physics **79**, 926 (1983).
- ⁶⁷ A. Hübsch, R. G. Endres, D. L. Cox, and R. R. P. Singh, Phys. Rev. Lett. **94**, 178102 (2005).
- ⁶⁸ L. Zhang, V. Michaud-Rioux, and H. Guo (unpublished).
- ⁶⁹ K. Lejaeghere, V. V. Speybroeck, G. V. Oost, and S. Cottner, Critical Reviews in Solid State and Materials Sciences **39**, 1 (2014).
- ⁷⁰ F. Birch, Phys. Rev. **71**, 809 (1947).
- ⁷¹ C. for Molecular Modeling, Comparing Solid State DFT Codes, Basis Sets and Potentials, <https://molmod.ugent.be/deltacodesdft/>, [Online; accessed 19-July-2015].