An \( h \)-adaptive finite element solver for the calculations of the electronic structures

Gang Bao\textsuperscript{a,b}, Guanghui Hu\textsuperscript{b,*}, Di Liu\textsuperscript{b}

\textsuperscript{a} Department of Mathematics, Zhejiang University, Hangzhou 310027, China
\textsuperscript{b} Department of Mathematics, Michigan State University, East Lansing, MI 48824, USA

\textbf{ARTICLE INFO}

\textbf{Article history:}
Received 11 December 2011
Received in revised form 31 March 2012
Accepted 8 April 2012
Available online 16 April 2012

\textbf{Keywords:}
Density functional theory
\( h \)-Adaptive
Kohn–Sham equation
Finite element method

\textbf{ABSTRACT}

In this paper, a framework of using \( h \)-adaptive finite element method for the Kohn–Sham equation on the tetrahedron mesh is presented. The Kohn–Sham equation is discretized by the finite element method, and the \( h \)-adaptive technique is adopted to optimize the accuracy and the efficiency of the algorithm. The locally optimal block preconditioned conjugate gradient method is employed for solving the generalized eigenvalue problem, and an algebraic multigrid preconditioner is used to accelerate the solver. A variety of numerical experiments demonstrate the effectiveness of our algorithm for both the all-electron and the pseudo-potential calculations.

\( \copyright \) 2012 Elsevier Inc. All rights reserved.

1. Introduction

Density functional theory (DFT) is a quantum model that has proved very successful in real applications [1]. To numerically solve the DFT model, various approaches have been developed. The most common and successful numerical schemes include the plane-wave (PW) expansion method [2,3] which expands the governing equation by the plane-wave basis set, and linear combination of atomic orbitals (LCAO) method [4] which uses the Gaussian basis set. Other numerical schemes include the finite difference methods [5,6], the wavelet methods [7], the finite element methods [8,9,1,10]. Recently, the framework of using discontinuous Galerkin methods for the Kohn–Sham equation is also introduced in [11].

For the well-developed plane-wave expansion method, it is very popular because of its attractive features. For example, its basis functions form a complete and orthonormal set which is independent of the atomic positions. However, there are still some disadvantages for this method. Compared with methods which are based on the local basis function, the PW method is not flexible for nonperiodic boundary conditions and/or domain with complicated boundary. The large parallel computing is necessary for the practical applications. However, the data flow through the computers is huge when the orthogonalization process is implemented of the PW method. In principle, all of the above disadvantages are from the use of the nonlocal basis set. To avoid these drawbacks, the real-space methods which have local basis set are developed.

Among these real-space methods such as the finite difference methods, the wavelet methods, and the finite element methods, we focus on the finite element methods in this paper based on the following considerations. Compared with other real-space methods, the finite element methods are much more suitable for the domain with arbitrary geometry. The total energy of the ground state of the electronic structures is approximated variationally by using the finite element methods. In addition, the finite element methods have been widely used in a variety of research fields such as the elasticity, the electromagnetism, the computational fluid dynamics. There are a large number of numerical algorithms such as the multigrid
methods, the a posteriori error estimation technique we may use for the reference. Especially, there are already a lot of mature software to implement the finite element methods in the market, which save us from coding the methods from scratch.

When using the finite element methods, an important task is to divide the domain into a set of high-quality elements (triangle or rectangle in the two-dimensional case, and tetrahedron or hexahedron in the three-dimensional case). To solve the Kohn–Sham equation, the uniform mesh may not be a good choice because of the singularity of the external potential, which is a term in the Hamiltonian operator. Furthermore, in a multiphysical model where the DFT model works together with, for example, the Maxwell equations to describe the behavior of the nanooptics, the scale of the DFT model is much smaller than that of the Maxwell equations. In the above situations, to obtain accurate numerical results using the uniform mesh will result in a large number of mesh grids in the domain. To reduce the amount of the mesh grids and keep the numerical accuracy at the meantime, a few high-quality nonuniform meshes are designed for atoms and molecules, see [12,13]. Excellent numerical results are observed by using these well-designed nonuniform meshes. However, these meshes are static, which may result in difficulty for describing the dramatic change of the electronic structure in the optimization of the geometry of the electronic structures, or the time-dependent DFT with a strong laser field. To cope with the above situations by using those static meshes, a straightforward way is to regenerate the mesh according to the current electronic structures, then interpolate the numerical results from the old mesh to the new mesh. However, this may cause a big degeneration of the efficiency and/or the numerical accuracy of the algorithm. What we need to do to avoid the above issue is a dynamical mesh which changes in terms of the electronic structures, which can be achieved by using the adaptive mesh techniques.

The adaptive mesh techniques mainly include the r-adaptive methods [14,15] which redistribute the mesh grids, and h-adaptive methods which locally refine and/or coarsen the mesh grids. There are also the p-adaptive methods which locally enrich the order of the basis function, and hp-adaptive methods which combine the h- and p-adaptive methods together. All these adaptive techniques have been investigated and applied in a variety of research fields, and the electronic structure calculation is one of them. For the application of these adaptive methods in the electronic structure calculations, people may refer to [16,17] for the adaptive coordinates methods, [10,1] for the h-adaptive methods, [18] for the r-adaptive methods. Some interesting discussion about hp-adaptive method for electronic structure calculation can also be found in [19]. In [18], a finite element solver for the Kohn–Sham equation is proposed, combined with an adaptive mesh redistribution technique. Numerical results show that, compared with the fixed mesh case, the adaptive algorithm improves the efficiency and the numerical accuracy significantly. However, there is a potential difficulty for the r-adaptive methods for the electronic structure calculations, when the time-dependent DFT is considered. There are a few time propagators for the time-dependent Kohn–Sham equation such as the “classical” propagators which based on the symmetric second order differentiation scheme, the Crank–Nicolson method, etc., or some “nonclassical” propagators such as the exponential midpoint rule, the time-reversal symmetry based propagator, and the Magnus expansions. All these schemes can be found in [20]. Although the “classical” propagators are easy to implement and reliable, the “nonclassical” propagators are suggested by those authors in [20], based on their numerical experience. In these “nonclassical” propagators, the evaluation of the Hamiltonian on the time step $t + \Delta t/2$ is needed. To obtain the evaluation, a kind of extrapolation based on the Hamiltonian on the previous time steps is implemented. This extrapolation could be very time-consuming for the r-adaptive methods because of the movement of the mesh grids. For the $h$-adaptive methods, a well-designed data structure for the mesh may help on the efficient implementation of the extrapolation.

In this paper, an $h$-adaptive finite element solver is presented for solving the Kohn–Sham equation. There are two main components in the algorithm. The first one is the self-consistent filed (SCF) iteration. In the SCF iteration, the Kohn–Sham equation is discretized by using the linear finite element methods. In the Hamiltonian operator, both the all-electron and the pseudo-potential cases are considered for the external potential, where the atomic pseudo-potential engine (APE) [21] is used to generate the pseudo-potentials. The Libxc [22] is adopted for the generation of the exchange–correlation potentials. To get the Hartree potential, the Poisson equation is solved by using the highly efficient algebraic multigrid method. Since we focus on the nonperiodic boundary conditions, the Dirichlet boundary condition is adopted and the boundary values need to be determined. Unlike the wavefunctions which decay exponentially from the position of the nucleus, the Hartree potential decays linearly, which means the zero-Dirichlet boundary condition is not suitable for the Poisson problem. In our method, the multipole expansion method is utilized to generate the boundary values. Note that for the numerical accuracy of the multipole expansion method, an efficiently large domain is needed. To solve the generalized eigenvalue problem, the locally optimal block preconditioned conjugate gradient (LOBPCG) method [23,24] is used.

The second component in our algorithm is about the mesh adaptivity. One of the important issues in the $h$-adaptive methods is the so-called indicator. Generally speaking, indicators mark the regions which need to be locally refined or coarsened in the domain, and always be given by certain a posteriori error estimation. For the case that there is only one orbital in the system, it is natural to generate the indicator by using the information from this orbital. However, for the case that there are more than one orbital in the system, generating the indicator by using an individual orbital is no longer a good idea, because every orbital in the system is expected to resolve well by using the mesh grids after the adaptivity. A straightforward idea for generating indicator for multiorbital case is to use the electron density. Because the electron density is one kind of summation of those orbitals, it contains the information from every orbital. However, the difference of the amplitude among those orbitals could be very large, especially for the all-electron calculations. Direct utilization of electron density may cause the difficulty on selection of the tolerance for the adaptivity. If the tolerance is too large, the orbital with small amplitude may not be resolved well by mesh grids. On the other hand, although a small tolerance helps on resolving all orbitals, this may cause the overload on the storage, and the efficiency of the algorithm may also be affected negatively. To avoid the above
issue, in this paper we adopt the following strategy. First, the indicator for each orbital is generated individually by using certain method. Then the normalization of each indicator is implemented. The final indicator is given by a combination of those normalized indicators. With this strategy, the efficiency and the accuracy of the algorithm could be balanced well. To generate the indicator for each orbital, we simply use the $l_2$-norm of the gradient of the orbital in every mesh element. Although there are a lot of excellent a posteriori error estimators which can be used for indicator generation such as the polynomial preserving recovery (PPR) [25–29], and recent work [30] for a posteriori error estimation of the quantum physics, we found that the indicator from the simple utilization of the $l_2$-norm of the gradient of the orbital works very well in our simulations. However, we want to mention that a good indicator is crucial for the application of the adaptive method, and an accurate and robust error estimator is always needed.

Another important issue for the mesh adaptivity is about the highly efficient implementation. The operations include the mesh refinement, the mesh coarsening, and the solution update from the old mesh to the new mesh. For the solution update, it is not necessarily necessary for the DFT calculation, because a random initial guess can always be used for each SCF iteration theoretically. However, a high-quality, high-efficiency solution update may accelerate the convergence dramatically. Furthermore, the update operation is required by the time-dependent DFT. In our method, a hierarchy geometry tree (HGT) for the mesh is used for the update operation. The numerical results show that our $h$-adaptive method works effectively in all simulations.

The rest of the paper is arranged as the following. In Section 2, the Kohn–Sham equation is briefly summarized. Then the finite element discretization of the Kohn–Sham equation is described, and the detailed discussion about the Poisson equation and its boundary condition for the Hartree potential is given. Then in Section 3, the $h$-adaptive method will be introduced, and some technical details are discussed. Section 4 evaluates our solver for a variety of atoms and molecules with known properties. Finally, our current work and highlights of some new directions are summarized in Section 5.

2. The Kohn–Sham equation and its finite element discretization

2.1. The Kohn–Sham equation

The Kohn–Sham equation can be read as

$$H\psi_i(\mathbf{x}) = \varepsilon_i \psi_i(\mathbf{x}),$$

where $H$ is the Hamiltonian operator, $\psi_i(\mathbf{x})$ is the wavefunction (eigenfunction) and $\varepsilon_i$ is its corresponding eigenenergy. For the Hamiltonian operator $H$, it consists of four terms,

$$H = -\frac{1}{2} \nabla^2 + V_{\text{ext}}(\mathbf{x}) + V_{\text{Hartree}}[n](\mathbf{x}) + V_{\text{xc}}[n](\mathbf{x}).$$

In the above formula, $n$ stands for the electron density, and has the relation with the wavefunction $\psi_i(\mathbf{x})$,

$$n = \sum_i^n \text{occ}_i |\psi_i|^2,$$

where $\text{occ}_i$ means the occupation number of the $i$th orbital, $N$ means the number of the occupied orbitals.

The term $-\frac{1}{2} \nabla^2$ in (2) is the kinetic energy operator, and is a dominant term in the Hamiltonian operator. $V_{\text{ext}}(\mathbf{x})$ is the external potential. In a closed system, the external potential stands for the attraction between the nucleus and the electron. For the all-electron calculation, the external potential $V_{\text{ext}}(\mathbf{x})$ is given by

$$V_{\text{ext}}(\mathbf{x}) = \sum_{i=1}^M Z_i \frac{\mathbf{R}_i}{||\mathbf{x} - \mathbf{R}_i||}$$

where $M$ is the number of the nucleus in the system, $Z_i$ is the charge of the $i$th nucleus, and $\mathbf{R}_i = ||\mathbf{x} - \mathbf{R}_i||$ means the length away from the $i$th nucleus, $\mathbf{R}_i$ stands for the coordinates of the $i$th nucleus. Notice that there is $1/|\mathbf{R}_i|$ singularity in the $V_{\text{ext}}$, and this causes the length scale difficulty for accurately resolving the solutions in the vicinity of the nucleus. However, it can be observed that the solutions for a atomic configuration can be separated into two groups, the “core” states and the “valence” states. In most situations, only valence states are active for the formation of the chemical bonds. Consequently, the pseudo-potential methods are developed to remove the core states in the system, and so is $1/|\mathbf{R}_i|$ singularity. By using the pseudo-potential in the Hamiltonian operator, a series of the pseudo-potentials $V_i^l$ and the corresponding pseudo-wavefunctions $u_i^l(\mathbf{x})$ for the $i$-th atom which solve the single $i$-th atom are needed. Then by using the Kleinman–Bylander form [31], the $V_{\text{ext}}(\mathbf{x})$ in the Hamiltonian operator can be expressed as

$$V_{\text{ext}}(\mathbf{x}) = \sum_i^M V_{\text{loc}}(\mathbf{x}) \psi_i(\mathbf{x}) + \sum_i^M \sum_{l,m} G_{l,m} u_i^l(\mathbf{x}) \Delta V_i^l(\mathbf{x}),$$

where

$$\Delta V_i^l(\mathbf{x}) = V_i^l(\mathbf{x}) - V_i^l(\mathbf{x}).$$
and

\[ G_{i,m}^j = \frac{\int u_{i,m}^j(x) \Delta V_l^j(x) \psi(x) \, dx}{\int u_{i,m}^j(x) \Delta V_l^j(x) \, dx}. \]

\( V_{\text{loc}}(\mathbf{x}) \) is the local pseudo-potential. Theoretically, \( V_{\text{loc}} \) could be any one of \( V_l^j \)'s. In our simulations, the most repulsive one is chosen as the local pseudo-potential.

\( V_{\text{Hartree}}(\mathbf{x}) \) in (2) is the Hartree (electrostatic) potential, which stands for the repulsion among the electrons. Its analytical form is given as

\[ V_{\text{Hartree}}[n](\mathbf{x}) = \int \frac{n(\mathbf{y})}{|\mathbf{x} - \mathbf{y}|} \, d\mathbf{y}. \]  

(6)

To evaluate the Hartree potential by using the above formula will cause \( O(N^2) \) operations, which is very time-consuming. Alternatively, the Hartree potential can be obtained by solving the following Poisson equation

\[
\begin{aligned}
-\nabla^2 V_{\text{Hartree}} &= 4\pi n, \\
V_{\text{Hartree}}(\mathbf{x}) &= 0 \quad \text{as } ||\mathbf{x}|| \to \infty.
\end{aligned}
\]

(7)

Solving the above Poisson equation by a multigrid method will reduce the operations to \( O(N) \).

Finally, \( V_{\text{xc}} \) in (2) is the exchange–correlation potential. This term explains the many-body interactions in the system. Unlike \( V_{\text{ext}} \) and \( V_{\text{Hartree}} \), the exchange–correlation potential \( V_{\text{xc}} \) does not have the analytical form. Hence, the approximation for this term is needed. We refer to [32] for the approximations such as the local density approximation (LDA), the generalized gradient approximation (GGA), etc.

2.2. The Finite element discretization

Note that in (1), the Hamiltonian operator \( H \) depends on the electron density \( n \). At the meantime, the electron density \( n \) depends on the unknowns \( \psi_l(\mathbf{x}) \), which means that the Kohn–Sham equation (1) is nonlinear. In this subsection, the finite element discretization of the Kohn–Sham equation (1) is first described. We suppose that the electron density \( n \) in the Hamiltonian \( H \) is known, and \( H \) only depends on the coordinates \( \mathbf{x} \). The nonlinearity of (1) will be discussed at the end of this subsection.

Suppose that the numerical simulations are implemented in the domain \( \Omega \in \mathbb{R}^3 \). The weak form of the Kohn–Sham equation (1) takes the form: Find \( N \) eigenpairs \((\lambda_i, \psi_i(\mathbf{x}))\), \( \lambda_i \in \mathbb{R} \) and \( \psi_i(\mathbf{x}) \in \mathbb{V} - \{0\} \) such that

\[
\int_{\Omega} \left( \frac{1}{2} \nabla \psi_l \cdot \nabla \phi + (V_{\text{ext}}(\mathbf{x}) + V_{\text{Hartree}}(\mathbf{x}) + V_{\text{xc}}(\mathbf{x})) \psi_l \phi \right) \, d\mathbf{x} = \epsilon_i \int_{\Omega} \psi_l \phi \, d\mathbf{x}
\]

(8)

for any \( \phi \in \mathbb{V} \). Note that the function in \( \mathbb{V} \) should satisfy certain boundary conditions.

To discretize the weak form (8), the domain \( \Omega \) is first divided into a set of tetrahedron elements, and there is no overlap between any two tetrahedron elements. Then a linear finite element space \( \mathbb{V}_h \) is built on this tetrahedron mesh. The basis polynomials for the linear tetrahedron element are \( \{1,x,y,z\} \), and the interpolation points for the basis function are four vertices of the each tetrahedron element. For each basis function which is formed by the above basis polynomials, it is 1 at its corresponding interpolation point, and 0 at other interpolation points.

With the finite element space \( \mathbb{V}_h \), the discretized form of (8) can be read as: Find \( N \) pairs \((\lambda_i^h, \psi_i^h(\mathbf{x}))\), \( \lambda_i^h \in \mathbb{R} \) and \( \psi_i^h \in \mathbb{V}_h \), such that

\[
\sum_{K \in \Omega} \int_{K \cap \Omega} \left( \frac{1}{2} \nabla \psi_l^h \cdot \nabla \phi + (V_{\text{ext}}(\mathbf{x}) + V_{\text{Hartree}}(\mathbf{x}) + V_{\text{xc}}(\mathbf{x})) \psi_l^h \phi \right) \, d\mathbf{x} = \epsilon_i^h \sum_{K \in \Omega} \int_{K \cap \Omega} \psi_l^h \phi \, d\mathbf{x},
\]

(9)

for any \( \phi \in \mathbb{V}_h \). In the simulations, we focus on the free-space boundary condition. Because the wavefunction decays exponentially from the nucleus, the zero–Dirichlet boundary condition is adopted.

If we define the \( L^2(\Omega) \) scalar product

\[ (\psi_i, \phi) = \int_\Omega \psi_i \phi \, d\mathbf{x}, \]

and the bilinear form

\[ a(\psi_i, \phi) = \int_\Omega \left( \frac{1}{2} \nabla \psi_l \cdot \nabla \phi + (V_{\text{ext}}(\mathbf{x}) + V_{\text{Hartree}}(\mathbf{x}) + V_{\text{xc}}(\mathbf{x})) \psi_l \phi \right) \, d\mathbf{x}, \]

the discretized Eq. (9) results in the following generalized eigenvalue problem

\[ K\psi_i^h = \epsilon_i^h M\psi_i^h, \]

(10)
where \( K_{ij} = a \left( \phi_i^e, \phi_j^e \right) \), and \( M_{ij} = \left( \phi_i^e, \phi_j^e \right) \), and \( \phi_i^e \) stands for the individual basis function. It can be seen easily from \( (9) \) that the matrix \( M \) is symmetric positive definite, but the matrix \( K \) is only symmetric. For the generalized eigenvalue problem \( (10) \), the left-most \( N \) eigenpair \( (\epsilon_i^e, \psi_i^e) \), \( i = 1, 2, \ldots, N \) are needed for calculating the ground state of the electronic structure. In our solver, the locally optimal block preconditioned conjugate gradient (LOBPCG) method [24] is used to find out those eigenpairs. Compared with the traditional Krylov subspace iterative methods, the LOBPCG finds the minimization/maximization of the trace of the projected Hamiltonian in a larger subspace, which means that the more accurate eigenpairs can be expected. Note that in the algorithm [24], at each iteration, the orthonormalizations of the eigenvectors, the residuals, and the LOBPCG “directions” are implemented separately. Of course this is cheaper than orthogonalizing all these vectors together. However, it may cause the instability for the algorithm because of the absence of the orthogonality of all vectors in the subspace. To improve it, the orthonormalization of all these vectors is implemented. Although it is more time-consuming than the original one, it is more stable. Because the system we consider in this paper is not too large, the increment of the CPU-time is not significant.

A appropriate preconditioner is needed in LOBPCG method to accelerate the convergence of the eigenvalue problem. A general choice of the preconditioner for a sparse matrix is from the incomplete Cholesky decomposition. However, note that for our problem in this paper, the kinetic potential term \(-\frac{1}{2} \nabla^2\) in \( (2) \) is a dominating one. Consequently, the matrix \( K \) which is only generated by the kinetic term is a very good approximation for the matrix \( K \) in \( (10) \). That means the matrix \( T = K + \beta M \) is a very good preconditioner in our problem. Furthermore, \( T \) is symmetric positive definite, which is a desired property for the LOBPCG method. \( \beta \) is a parameter which is problem-dependent. In the numerical simulation, the preconditioner \( T \) improves the convergence of the eigenvalue problem dramatically.

As we mentioned before, the Kohn–Sham equation is actually nonlinear. In the numerical simulation, the SCF iteration is adopted to solve this nonlinear problem, and its implementation can be depicted by Algorithm 1.

In Algorithm 1, \( \psi_{i\text{initial}} \) is the initial guess for the wavefunctions, and a random initialization is used in the simulations. \( \text{MITER} \) and \( \text{TOL} \) denote the maximum number of the iteration and the tolerance, respectively. The simple mixing scheme \( n_{\text{out}} = n_{\text{out}} + (1 - \alpha) n_{\text{in}} \) is used to improve the convergence of the SCF iteration. The selection of the parameter \( \alpha \) is problem-dependent.

### Algorithm 1. The SCF iteration.

**Data:** \( \psi_{i\text{initial}}, \text{MITER}, \text{TOL} 

**Result:** \( (\epsilon_i^e, \psi_i^e), i = 1, 2, \ldots, N \)

Evaluate \( n_{\text{out}} \) with \( \psi_{i\text{initial}} \) by using \( (3) \);

while \( \| n_{\text{out}}^n - n_{\text{out}}^n \|_2 > \text{TOL} \) or \( \text{iter} < \text{MITER} \) do

\( n_{\text{in}}^n = n_{\text{out}}^n \);

Evaluate \( V_{\text{Hartree}}(\vec{R}) \), and \( V_{\text{xc}}(\vec{R}) \) by using \( n_{\text{in}}^n \);

Form the eigenvalue system \( (10) \);

Solve \( (10) \);

Evaluate \( n_{\text{out}} \) by using \( (3) \);

\( n_{\text{out}}^n = \alpha n_{\text{out}} + (1 - \alpha) n_{\text{in}}^n \);

\( \text{iter} + 1 \);

end

2.3. The boundary conditions for the Poisson problem

To evaluate the Hartree potential \( V_{\text{Hartree}} \), the Poisson problem \( (7) \) needs to be solved. However, the exact zero-boundary condition is only valid when \( |\vec{x}| \rightarrow \infty \). In the practical applications, the domain is always bounded. Unlike the wavefunction \( \psi(\vec{x}) \) which is exponentially decay from the position of the nucleus, the Hartree potential \( V_{\text{Hartree}} \) decays linearly. Consequently, appropriate methods are needed to deal with this long range effect.

There are two kinds of the techniques in the literature. The first one is to use the multipole expansion. Suppose there are \( M \) point charges \( q_i \) in the system, with the position \( \vec{X}_i, i = 1, 2, \ldots, M \). These charges are clustered in a sphere with the radius \( R \), which means \( |\vec{X}_i| < R, \forall i \). Then for any position \( \vec{X} \) with \( |\vec{X}| > R \), the electrostatic potential at this position can be approximated by using the multipole expansion. For our algorithm where the linear finite element space is used, approximating the boundary value by the expansion till quadrupole term is sufficient to keep the numerical accuracy. In [18], the multipole expansion method is adopted, and desired results are successfully obtained.

Different from evaluating the boundary value by the multipole expansion method, the second method tries to neutralize the system by introducing a positive charges. Since the system is neutralized, the electrostatic potential on the boundary becomes zero, then the zero-Dirichlet boundary condition can be used directly. By using this method, we will solve the following Poisson problem instead of \( (7) \) in the bounded domain \( \Omega \),

\[
\begin{align*}
- \nabla^2 V_{\text{Hartree}} &= 4\pi(n + \rho) & \text{in } \Omega, \\
\nabla V_{\text{Hartree}} &= 0, & \text{on } \partial\Omega.
\end{align*}
\]
where \( \rho \) is the introduced position charge. By solving the above problem, we obtain the potential \( V_{\text{Hartree}} \). The next step is to remove the electrostatic potential \( V_\rho \), which is generated by the positive charge \( \rho \), in the Hamiltonian operator.

The Gaussian charge \([1,33,13]\) is a popular choice for the charge \( \rho \). In the implementation, the charge \( \rho \) is constructed by a sum of Gaussian charges \( \rho_i, i = 1, 2, \ldots, M \), located at each position \( \vec{X}_i \) of the nucleus.

\[
\rho_i(\vec{x}) = -\frac{Z_i}{(\sqrt{\pi} r_i)} \exp \left( -\frac{|\vec{x} - \vec{X}_i|^2}{(r_i)^2} \right),
\]

where \( r_i \) is a parameter which needs to be chosen carefully. Then the electrostatic potential caused by these Gaussian charges can be presented analytically as

\[
V_\rho(\vec{x}) = \sum_{i=1}^{M} -\frac{Z_i}{|\vec{x} - \vec{X}_i|} \text{erf} \left( \frac{|\vec{x} - \vec{X}_i|}{r_i} \right).
\]

Both methods discussed above can be used for a free boundary problem. In our algorithm, the multipole expansion method is employed.

### 3. Local grid refinement and coarsening

In the last section, the Kohn–Sham equation (1) has been discretized and solved on a fixed mesh. As we discussed in the first section, an adaptive method becomes necessary for the further improvement of the numerical accuracy and the efficiency of the algorithm. In this section, the technical details are introduced about the local refinement and coarsening of the mesh grids.

To efficiently implement the \( h \)-adaptive method, two components need to be designed carefully. The first one is an effective management mechanism of the mesh grids. With this management mechanism, the mesh grids can be locally refined or coarsened flexibly, and the solutions update from the old mesh to the new mesh can also be implemented efficiently. The second one is about the effective indicators for the adaptivity. A posteriori error estimation is adopted to generate the indicators in our algorithm.

#### 3.1. The hierarchical geometry tree

A well-designed data structure for the mesh grids is needed for an effective management mechanism. In our algorithm, the hierarchy geometry tree (HGT) \([34]\) is utilized.

First, the mesh structure is described hierarchically, which means that the mesh information is given from the lowest dimension (0-D, the points) to the highest dimension (3-D in this paper, the tetrahedrons) hierarchically. An element such as a point for 0-D, an edge for 1-D, a triangle for 2-D, or a tetrahedron for 3-D is called a geometry. In the hierarchical description of a tetrahedron, all geometries have a belonging-to relationship. For example, if a triangle face is one of the faces of a tetrahedron, this triangle face belongs to this tetrahedron. Similarly, this triangle face has its own edges and vertexes. With this hierarchical structure, the geometry information of a tetrahedron can be referred flexibly, and the refinement and coarsening of a mesh can also be implemented efficiently.

By uniformly refining a tetrahedron \( T_0 \), eight sub-tetrahedrons \( \{T_{0.0}, T_{0.1}, \ldots, T_{0.7}\} \) are generated, which is demonstrated in Fig. 1. This procedure can be managed by the octree data structure, which is tree data structure in which each internal node has exactly eight children. The above uniform refinement can be expressed by the octree which is given by Fig. 2 (the top two rows). When the local refinement and coarsening techniques are adopted, only some tetrahedrons are refined. For example, in Fig. 2 (the bottom row), the tetrahedron \( T_{0.0} \) and \( T_{0.6} \) are further refined. In the above octree, we call \( T_0 \) the root node, and those node without further descents like \( T_{0.1} \) and \( T_{0.6.0} \) the leaf nodes. Suppose there is a set of root nodes
\( \{ T_i \}, \; \; i = 0, 1, 2, \ldots \) which form a mesh for a domain \( \Omega \), then from the above definition we know that a set of all leaf nodes of these HGTs also form a mesh.

The mesh refinement and coarsening can be implemented efficiently by using HGT. However, building the finite element space directly on a set of all leaf nodes is not implementable because of the hanging points in the direct neighbors of those refined tetrahedrons. To deal with these hanging points, two kinds of geometries are introduced: twin-tetrahedron and four-tetrahedron. They are demonstrated in Fig. 3. For the twin-tetrahedron geometry (Fig. 3, left one), it actually consists of two tetrahedrons \( ABED \) and \( AECD \). There are five degrees of freedom (DOF) in the twin-tetrahedron, and the interpolation points of these DOFs are \( A, B, C, D, \) and \( E \). To make the finite element space conform, the following strategy is adopted to construct the basis functions in the twin-tetrahedron geometry: For each basis function, its value is 1 at its corresponding interpolation point, and 0 at other interpolation points. The support of the basis function whose interpolation point is a common point of two sub-tetrahedrons in a twin-tetrahedron geometry such as \( A, D, \) and \( E \) in Fig. 3 (left one), is the whole twin-tetrahedron geometry. For the noncommon point like \( B \), the support of its corresponding basis function is only the tetrahedron \( ABED \). Similarly, the support of the basis function which corresponds to the point \( C \) is the tetrahedron \( AECD \). To construct the basis functions in the four-tetrahedron (Fig. 3, right one), the similar strategy is used. With the twin-tetrahedron geometry and the four-tetrahedron geometry, a conform finite element space can be built smoothly in a mesh where the local refinement is implemented.

With the HGT, the solution update from the old mesh to the new mesh after the local refinement and coarsening can also be done efficiently. As we mentioned before, a set of all leaf nodes forms a mesh. Although different mesh corresponds to different set of leaf nodes, all meshes are from the same set of the root nodes. Then the belonging-to relationship of the geometries between two meshes can be analyzed easily. As a result, the solution update can be implemented efficiently according to the relationship.

3.2. The indicator

An effective indicator generator is another important component for the \( h \)-adaptive methods, because the local refinement and coarsening of the mesh grids will be implemented according to the indicators.

In the calculations of the eigenpairs, we always hope that all the eigenpairs are described well in (10). For a real-space method, such good description is from the accurate evaluation of each term in the Hamiltonian operation on the one hand, and also from the a high quality mesh on the other hand. For a high quality mesh, it should resolve every orbital (eigenvector) well. The situation is simple when there is only one orbital in the system. However, when there are more than one orbital in the system, the indicator should be generated according to all orbitals in the system. Note that there is a relationship (3) between the electron density and the wavefunctions. A straightforward way to generate an appropriate indicator is to use the electron density \( n \). However, there is a potential difficulty for choosing the tolerance in the adaptivity. In a multi-orbital
system, the difference of the altitude among those orbitals could be very large, especially for the all-electron calculations. For the indicator directly from electron density, a too large tolerance would result in the deficiency on resolving the orbitals with small altitude, and a too small tolerance may cause the overload on the storage, and further affect the efficiency of the algorithm.

To resolve the above issue, in this paper, the following strategy is adopted. First of all, the indicator for each orbital is generated. Then these indicators are normalized, respectively. Finally, a combination of these normalized indicators is presented as the indicator. By using this way, the numerical accuracy and the efficiency of the algorithm are balanced in our simulations.

The ambiguous part in the above strategy is the way to generate the indicator for each orbital. Traditionally, a posteriori error estimation is a good choice. However, a very simple method is adopted in our method for this purpose. That is, the following form is used as the indicator in each mesh element,

\[
\text{Ind}_i^k = \left( \int_{T_i} (\nabla \psi_k)^2 \, dx \right)^{1/2},
\]

where \(T_i\) stands for the \(i\)th element, and \(k\) stands for the \(k\)th orbital.

Suppose now we have \(M\) indicators \(\text{Ind}_i^k, k = 0, 1, 2, \ldots, M\) for \(M\) orbitals. To balance the contribution of each indicator \(\text{Ind}_i^k\) in the final form, the normalization is implemented for each \(\text{Ind}_i^k\), and we get \(\tilde{\text{Ind}}_i^k\). Then the following combination of these normalized indicator is used as the final indicator in the element \(T_i\),

\[
\text{Ind}_i = \sum_{k=0}^{M} \tau_k \tilde{\text{Ind}}_i^k,
\]

where \(\tau_k\) is a positive parameter which is used to further balance the contribution of each indicator.

The criterion for refining a tetrahedron into 8 sub-tetrahedrons, or coarsening a patch of tetrahedrons to be a single but larger tetrahedron can be easily explained as the following. Suppose we have a mesh \(\{T_i, i = 0, 1, 2, \ldots\}\), and the indicators \(\{\text{Ind}_i\}, i = 0, 1, 2, \ldots\). We also need an appropriate tolerance \(\varepsilon_h\). Then for the tetrahedron whose indicator is larger than \(2^{Nz}\varepsilon_h\), where \(N\) is the dimension of the problem, and \(z\) is the convergence order of the numerical method (for the linear finite element method, \(z = 2\)), the refinement will be implemented. If the indicators of the tetrahedrons in a patch are all smaller than \(2^{-z}\varepsilon_h\), the coarsening will be implemented. The numerical results show that our strategy for the mesh adaptivity works very well.

So far, all the implementation details in the \(h\)-adaptive finite element solver for the Kohn–Sham equation have been discussed. The flowchart of the whole algorithm is given in Algorithm 2.

Algorithm 2. The \(h\)-adaptive finite element algorithm for the Kohn–Sham equation.

**Data:** \(\psi_i^{\text{initial}}, \text{MITER}, \text{TOL}\).

**Result:** \((\varepsilon_i, \psi_i), i = 1, 2, \ldots, N, E\).

**while** \(|E^{n+1} - E^n| > \text{TOL} \text{ or } \text{iter} < \text{MITER} \) **do**

- Implement Algorithm 1 on the current mesh;
- Get the indicators on the current mesh;
- Locally refine and coarsen the mesh;
- Update the wavefunctions from the old mesh to the new mesh;
- \(\text{iter}++\);

**end**

In the above algorithm, \(E\) stands for the total energy of the ground state, and \(\text{MITER}\) and \(\text{TOL}\) are the maximum number of the iteration and the tolerance, respectively.

4. Numerical solutions

The numerical simulations in this section are implemented with a C++ library AFEABIC (Adaptive finite element package for the AB-initio calculations), which is developed by the authors and will be soon released. The exchange–correlation potentials are generated by the library LibXC [22], and the pseudo-potentials are given by APE [21]. For the generation of the initial 3-D mesh, the library GMSH [35]. All simulations are implemented on a personal desktop with Intel Pentium Processor E6800(3.33 GHz, 2 M Cache), and 4 gigabytes ram.

4.1. Atoms

In this subsection, the ground state energies of several atoms are calculated with Algorithm 2. The computational domain is chosen as \([-15, 15]^3\).
First, the ground state energy of the helium atom is calculated. In this example, the all-electron calculation is implemented. Fig. 4 shows the results. From the figure (Fig. 4, top left), we can see that a radial mesh is generated successfully with our $h$-adaptive method. The mesh size changes dramatically. In the outer layer, the mesh size is around 20 atomic unit (Fig. 4, bottom left), while in the vicinity of the nucleus, it is around 0.03 atomic unit (Fig. 4, bottom right). With this

![Image of mesh generation](image)

Fig. 4. The ground state of the Helium atom. Top Left: the mesh in the whole domain, there are total 3851 mesh grids in the domain. Top Right: the isosurface of the electron density when $n = 0.05$. Bottom: the sliced mesh on the $X$–$Y$ plane from large domain (left) to the small domain (right).

![Image of total energy](image)

Fig. 5. The total energies of the atoms on the successively refined meshes with the $h$-adaptive method. Left: The lithium atom, Right: The Carbon atom. The all-electron calculations are implemented.
optimized mesh, the total energy of the ground state of the Helium atom is calculated as \(-2.839\) atomic unit with 32,607 mesh grids. This approximate energy agrees with the reference value \((-2.835\) atomic unit\([36]\)) very well. As a comparison, the approximate energy is \(-2.37\) atomic unit when the uniform mesh (137,313 mesh grids) is employed. In this case, the number of the mesh grids is around 4 times the number of the \(h\)-adaptive case. It means that a better approximation can be obtained by using less mesh grids with the help of our \(h\)-adaptive method.

Fig. 5 shows the convergence history of the total energy on the successively refined meshes with the \(h\)-adaptive method for the lithium atom (left one), and carbon atom (right one). It is obvious that our results for both atoms agree with the reference values \([36]\) (\(-7.34\) for lithium atom, and \(-37.43\) for carbon atom) very well.

Table 1 shows that our algorithm also works well when using the pseudo-potential. The norm-conserving Troullier–Martins pseudo-potential \([37]\) is implemented in the simulation, and the Kleinman–Bylander form \((5)\) is adopted. It can be seen from the table that our algorithm presents very good approximations, compared with those experimental values \([38]\).

### Table 1

<table>
<thead>
<tr>
<th>Atom</th>
<th>Lithium</th>
<th>Beryllium</th>
<th>Sodium</th>
<th>Magnesium</th>
<th>Aluminum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pseudo-energy</td>
<td>-5.2</td>
<td>-27.1</td>
<td>-5.0</td>
<td>-22.7</td>
<td>-52.9</td>
</tr>
<tr>
<td>Experiment ([38])</td>
<td>-5.3</td>
<td>-27.5</td>
<td>-5.1</td>
<td>-22.7</td>
<td>-53.3</td>
</tr>
</tbody>
</table>

Fig. 6. The results of the LiH using all-electron calculation. Top left: the electron density around two nuclei (left one is lithium nucleus, and right one is hydrogen nucleus); Top right: the sliced mesh on the X-Y plane around two nuclei; Bottom left: the convergence history of the total energy with the adaptive refinement; Bottom right: total energy vs. Li-H bond length.
4.2. Molecules

The first simulation of the molecule contributes to the lithium hydride (LiH). From valence bond theory, there is only one bond in the molecule. To calculate its ground state, the all-electron calculation is implemented. The initial bond length of Li-H is set to 3.01 atomic units according to the experimental value [39]. The numerical results are demonstrated in Fig. 6, and the following observations can be made from the results: (1), since the all-electron calculation is adopted, it can be imagined that the electron density around the lithium nucleus should be larger than that around the hydrogen nucleus, which is obviously seen from Fig. 6 (Top left); (2), with our $h$-adaptive technique, the regions with large variation of the electron density are successfully resolved by mesh grids, which means that the density of the mesh grids in the vicinities of the nuclei is much larger than other regions, see Fig. 6 (Top right). Furthermore, the density of mesh grids around the lithium nucleus is larger than that around the hydrogen nucleus, which shows that the variation of the electron density around the lithium nucleus is more intense; (3). Fig. 6 (Bottom left) shows the numerical convergence of our $h$-adaptive finite element method. With the $h$-adaptive mesh refinement, the total energy of LiH converges to around 8.0 hartree, which agrees with the reference value [39] very well; (4). It can be read from Fig. 6 (Bottom right) that the bond length of Li-H from our calculation is around 3.04 atomic units. Although it is slightly bigger than the experimental value (3.01 atomic units), the approximation is still very good.

As the second example, the ground state of the borane is simulated using our algorithm. For a single boron atom, its ground state configuration is $1s^2, 2s^2$, and $2p_x^1$. Based on the hybridization theory, a $2s$ electron is excited into a $2p$ orbital, and then the $2s$ orbital mixed with that two $2p$ orbitals to form three $sp^2$ hybrids. So there are three bonds in borane, and these three bonds are of the same length, and the angle between two bonds is equal to $120^\circ$. In the simulation, the bond length is chosen as 2.2491 atomic units according to the experimental result [39]. Since there is no contribution from $1s$ orbital of boron atom for forming the bond of borane, it is reasonable to use pseudo-potential for the boron atom in the Hamiltonian operator.

In the simulation, the total energy of the borane converges to around $-4.65$ hartree after four times adaptive refinement (17,616 mesh grids), which is comparable with the reference values in [39]. The numerical results are demonstrated in Fig. 7. It can be seen obviously that the electron density in the regions between the boron atom and each hydrogen atom is much larger than that in other regions. This clearly shows the location of three bonds which coincides with the experimental results. Fig. 7 (right one) demonstrates that the mesh grids successfully resolve those important regions with our $h$-adaptive methods. This example shows that our $h$-adaptive finite element solver also works very well when pseudo-potentials are adopted for simulating the molecules.

5. Concluding remarks

In this paper, we developed an $h$-adaptive finite element solver for solving the Kohn–Sham equation with the linear finite element space. The exchange–correlation potential and the pseudo-potential in the Hamiltonian operator are evaluated using the open source libraries LibXC [22] and APE [21], respectively. The algebraic multigrid method is adopted for solving the Poisson equation for the Hartree potential. The SCF iteration is used for the nonlinear eigenvalue problem, and the LOBPCG method is selected as the solver for the linear eigenvalue problem at each iteration. To improve the convergence of the SCF iteration, the simple mixing scheme is chosen. To improve the above finite element solver, an $h$-adaptive method is introduced to optimize the mesh quality. For consideration of the efficiency of the algorithm, the mesh structure is managed using the HGT. An indicator which is based on the gradient of the each wavefunction is given for the adaptivity.
Several numerical experiments including simulations of atoms and molecules have been conducted by using our solver. Both all-electron and pseudo-potential cases are implemented. The numerical results show that our $h$-adaptive finite element technique improves the mesh quality significantly according to the electronic structures. Compared with the uniform mesh case, a large amount of mesh grids are saved by using the $h$-adaptive method, and the accurate approximations can still be obtained at the same time. Furthermore, during the calculation of the relationship between the bond length and the total energy for the molecules, after the nuclei move, the distribution of the mesh grids can be improved accordingly with our $h$-adaptive method. That means the quality of the mesh grid is improved dynamically in our solver, which is more efficient than remeshing when a fixed mesh is no longer appropriate. More importantly, with the HGT, an efficient interpolation procedure can be designed for updating the wavefunctions from the old mesh to the new mesh. This makes the extension of our $h$-adaptive finite element solver to the time-dependent density functional theory (TDDFT) possible.

Another future direction is to study the multi-mesh case based on the following observations: (1). The selection of the tolerance in the adaptive procedure is not trivial, especially for the all-electron calculation, which is due to the different amplitudes of the wavefunctions. The method proposed in Section 3.2 partially resolves this issue. However, an improper tolerance still affects the numerical accuracy and the efficiency of the algorithm in the simulation. (2). When a single mesh is used in the simulation, each eigenpair will be solved using the same linear system. This is not an optimal choice. For example, to simulate the Carbon atom, the mesh is optimized using the $h$-adaptive method to resolve the $1s$ orbital. However, since the amplitude of $2p$ orbital is much smaller than the $1s$ orbital, see Fig. 8, the $2p$ orbital is actually over-resolved by using the same mesh. Therefore, a relative coarse mesh is already sufficient for the $2p$ orbital. In other words, the calculation of the $2p$ orbital can be implemented on a smaller linear system. It would be reasonable to assume that separate simulations of the wavefunctions may enhance the efficiency of current algorithm. This idea might also apply to the simulation of a large molecule. In that case, since the large variation of each orbital appears in very different region, a separate calculation would also make sense.

In order to resolve the above two issues, we plan to investigate the multi-mesh case, which is capable of distributing each orbital a different mesh and constructing the linear system and finite element space on its own mesh. Such a strategy would also permit the use of a different tolerances for different orbitals. It would also be flexible to control the size of the linear system for each orbital. The project is ongoing. The technique and numerical experiments on the $h$-adaptive finite element method for the multi-mesh case will be reported in a forthcoming paper.

Acknowledgment

The authors thank Prof. Chao Yang (Lawrence Berkeley National Laboratory) for his helpful comments, suggestions on this work. The research was supported in part by the NSF Focused Research Group grant DMS-0968360. The research of G. Bao was supported in part by the NSFC grants DMS-0908325, CCF-0830161, EAR-0724527, and DMS-0968360, the ONR grant N00014-12-1-0319, a Key Project of the Major Research Plan of NSFC (No. 91130004), and the On liberty from Zhejiang University.

References


