Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 223 (2007) 759–773
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

Finite element approach for density functional theory calculations on locally-refined meshes

J.-L. Fattebert *, R.D. Hornung, A.M. Wissink [1]

Center for Applied Scientific Computing (CASC), Lawrence Livermore National Laboratory, 7000 East Avenue, L-561, Livermore, CA 94551, United States

Received 29 March 2006; received in revised form 8 August 2006; accepted 3 October 2006
Available online 20 November 2006

Abstract

We present a quadratic finite element approach to discretize the Kohn–Sham equations on structured non-uniform meshes. A multigrid FAC preconditioner is proposed to iteratively solve the equations by an accelerated steepest descent scheme. The method was implemented using SAMRAI, a parallel software infrastructure for general AMR applications. Examples of applications to small nanoclusters calculations are presented.

PACS: 71.15.Ap; 71.15.Dx; 71.15.−m

MSC: 65F15; 65F50; 65N30; 65N55

Keywords: Finite element method; Multigrid; Kohn–Sham equations; Composite mesh

1. Introduction

Density functional theory (DFT) is a quantum model that has proved very successful in real applications, ranging from optical properties of nanostructures to phase diagrams of various materials. It introduces an independent particles description of the electronic structure of molecules or materials which is much simpler to treat than the original many-body Schroedinger equations [1,2]. Simulating realistic physical systems by DFT however is still computationally very demanding. More efficient, scalable numerical algorithms that

* Corresponding author.
E-mail address: fattebert1@llnl.gov (J.-L. Fattebert).
[1] Present address: NASA Ames Research Center, Moffet Field, CA, USA.

0021-9991/$ - see front matter
doi:10.1016/j.jcp.2006.10.013

reduce computer time and enable larger simulations are always in demand by chemists, physicists and biologists who are studying phenomenon at the molecular level.

The finite element (FE) method (see *e.g.* [3]), a popular solution technique for partial differential equations, has only recently begun to be used for solving the Kohn–Sham (KS) equations of density functional theory for realistic 3D applications [4–8]. Traditionally, pseudo-spectral approaches have been the most popular under the denomination plane waves (PW) method. The regular use of periodic boundary conditions with simple geometries explains this preference. However, as computer power increases and interest in studying larger and more diverse systems grows, discretzations using finite differences or finite elements, often referred to as real-space approaches, have recently attracted more interest [9]. The first motivation for real-space approaches is that they are easier to parallelize than pseudo-spectral approaches [10]. Another motivation for real-space discretizations is that algorithm complexity may be reduced from $O(N^3)$ to $O(N)$ by representing the electronic structure using a set of N non-orthogonal strictly localized orbitals [5,11–13]. In many cases, one can find a representation that spans a subspace very close to the invariant subspace associated with the occupied electronic states, usually described in term of eigenfunctions. In this paper, we focus on another motivation which is that we may refine the mesh locally to reduce the number of degrees of freedom needed to describe electronic wave functions in regions where they are very smooth. The use of a locally-refined structured mesh, when possible, leads to numerically more efficient algorithms. We hope that all aforementioned advantages of real-space methods can be realized leading to very efficient algorithms.

Various approaches for 3D DFT calculations using local mesh refinement and finite differences [14–16] and finite elements [17,6,18] have been explored. When using local mesh refinement, one faces the difficulty of building an efficient parallel implementation. A useful DFT code must be parallel to be competitive with highly optimized, parallel PW codes. Fortunately, one can rely on existing parallel infrastructure to facilitate the implementation [18]. We have developed an electronic structure code based on SAMRAI, an object-oriented, parallel software infrastructure for general AMR applications on structured grids [19] developed at Lawrence Livermore National Laboratory.

In the present work, we use the pseudopotential approximation which replaces singular atomic potentials by smoother regular potential functions that include core electron effects. Beside removing singularities, this approximation also simplifies the problem by removing degrees of freedom associated with the core electrons. These electrons are considered frozen since their effect on chemical binding can often be neglected. In this paper, we have chosen applications that require only local pseudopotentials; that is, atomic potentials that can be represented by simple radial functions.

The discussion in this paper is restricted to parallelepiped domains. This is general enough to treat most solid state applications where the computational domain has to coincide with a cell invariant under the crystal structure symmetry. For finite systems surrounded by vacuum, using a parallelepiped domain is also an appropriate approach. From a computational point of view, this restriction allows for the use of structured meshes which facilitates code implementation and improves numerical efficiency, allowing for instance matrix-free implementations.

In this paper, we propose a hierarchical finite element discretization for DFT calculations. For concreteness and to simplify the discussion, we present the quadratic finite element case which is also the special case we have implemented. But the approach can be generalized to higher order finite elements. An essential difference between our approach and the one proposed by Tsuchida and Tsukada [4] is the hierarchical formulation we use. This important feature allows to use simplified steepest descent directions vectors and to design a multigrid Poisson solver and preconditioner suited to our discretization. We also propose a discrete energy functional consistent with the weak discrete formulation of the KS equations, which ensures a minimum principle for the solution of the discretized problem. The weak formulation of the Kohn–Sham equations and their finite element discretization are introduced in Section 2. We then present a finite element approach for the full non-linear Density Functional Theory problem in Section 3. The solvers for the KS equations and the Poisson problem on structured adaptive meshes are presented in Section 4. Section 5 describes the implementation of our algorithm using the tools provided by the SAMRAI library. Finally, in Section 6 we illustrate our numerical approach with accuracy and convergence tests on some simple electronic structure calculations for beryllium clusters.

2. Kohn–Sham problem and its discretization

2.1. Kohn–Sham equations

We consider a 3D computational parallelepiped domain Ω. We are interested in solving the weak form of the Kohn–Sham equations for $2N$ electrons, that is finding N pairs $(\lambda^{(i)}, u^{(i)})$, $\lambda^{(i)} \in \Re, u^{(i)} \in \mathscr{V} - \{0\}$, such that[2]

$$\int_\Omega (\nabla u^{(i)} \nabla v + q(\{u^{(i)}\}_{i=1}^N)(\mathbf{x}) v(\mathbf{x}))\, \mathrm{d}\mathbf{x} = \lambda^{(i)} \int_\Omega u^{(i)} v\, \mathrm{d}\mathbf{x} \tag{1}$$

for all v in the admissible space $\mathscr{V}$. Functions in $\mathscr{V}$ should satisfy the essential boundary conditions, such as zero Dirichlet or periodic boundary conditions, be continuous, and have first derivatives with finite energy. We are usually interested in the N lowest eigenvalues $\lambda^{(i)}$ which can be interpreted as single particle energies for the electronic ground state. In this paper, we limit the discussion to non-metallic systems, *i.e.* we assume that those N eigenvalues are separated from the rest of the spectrum by a finite "band"-gap. The Kohn–Sham potential operator q is nonlinear. In this section, we ignore the difficulty introduced by this nonlinearity by assuming that q is a fixed scalar field depending on $\mathbf{x}$ only, and write $q(u)(\mathbf{x}) = q(\mathbf{x}) \cdot u(\mathbf{x})$. The nonlinearity will be treated later in Section 3.

Defining the $L^2(\Omega)$ scalar product

$$(u, v) = \int_\Omega u \cdot v\, \mathrm{d}\mathbf{x},$$

and the bilinear form

$$a(u, v) = \int_\Omega \nabla u \nabla v + q \cdot u \cdot v\, \mathrm{d}\mathbf{x}.$$

Eq. (1) can be written as

$$a(u^{(i)}, v) = \lambda^{(i)}(u^{(i)}, v), \quad \forall v \in \mathscr{V}. \tag{2}$$

We also define

$$b(u, v) = \int_\Omega \nabla u \nabla v\, \mathrm{d}\mathbf{x}.$$

Let $S_h \subset \mathscr{V}$ be a finite element space, $h > 0$ a discretization parameter. The finite element discretization of Eq. (2) requires finding the lowest eigenvalues $\lambda_h^{(i)} \in \Re$ and corresponding eigenfunction $u_h^{(i)} \in S_h$, such that

$$a(u_h^{(i)}, v_h) = \lambda_h^{(i)}(u_h^{(i)}, v_h)\ \forall v_h \in S_h. \tag{3}$$

In algebraic notation, Eq. (3) leads to a generalized matrix eigenvalue problem

$$K\mathbf{u}^{(i)} = \lambda_h^{(i)} M \mathbf{u}^{(i)}, \tag{4}$$

where K and M are the stiffness and mass matrices in the finite element basis. That is

$$(K)_{ij} = a(\phi_i^e, \phi_j^e), (M)_{ij} = (\phi_i^e, \phi_j^e),$$

where ϕ_i^e are the individual FE basis functions. M is symmetric positive definite, while K is simply symmetric. M and K are sparse matrices. The vector $\mathbf{u}^{(i)}$ is defined as the vector representation of $u_h^{(i)}$, *i.e.* a list of the expansion coefficients of $u_h^{(i)}$ in the finite element basis. We also define L to be the matrix representation of $b(\cdot,\cdot)$ in the finite element basis.

Note that matrices such as M, K, or L, operate on vectors made of the FE coefficients of functions in S_h. They generate vectors whose components should not be considered as coefficients of a function expansion in

[2] Here, we neglect the spin of the electrons and assume each electronic orbital is doubly occupied.

the FE basis, but dot products of a function with the FE basis functions. For example, the coefficient $(M\mathbf{v})_i = (\phi_i^e, v_h)$ is the scalar product of the FE function v_h and the FE basis function ϕ_i^e. To distinguish between these different types of vectors, we denote the vector space of scalar products by elementary FE basis functions as S_h^* and we use the superscript $*$ to indicate vectors in S_h^*. This distinction is important when one considers linear compositions of various vectors, in iterative algorithms for example.

We are interested in the sum of the N lowest eigenvalue $\lambda_h^{(i)}$ that satisfy Eq. (4). We also need the corresponding invariant subspace $\mathscr{U}^{(N)} = \text{span}\{\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(N)}\}$. We will use the matrix notation $U = (\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(N)})$ to denote a basis $\{\mathbf{u}^{(i)}\}_{i=1}^N$ of the subspace $\mathscr{U}^{(N)}$.

2.2. Finite element space for structured AMR

We consider functions that are continuous over the whole domain and can be represented in a polynomial basis within each grid cell. We use the 3D quadratic serendipity brick elements consisting of eight nodes (cell corners) and 12 edges values (located in the middle of each edge); *e.g.* see [20]. In this approach, the polynomial basis is expressed using 20 functions: $1, x, y, z, x^2, y^2, z^2, xy, xz, yz, x^2y, xy^2, x^2z, xz^2, y^2z, yz^2, xyz, x^2yz, xy^2z, xyz^2$. We associate degrees of freedom to the eight nodes and 12 edges of each brick element. From the polynomial basis, one can build 20 shape functions, each having the value 1 at one node or edge and the value 0 at all other nodes and edges. In our hierarchical approach, we expand the solution in a cell in a basis given by the eight trilinear shape functions, each of which is 1 at one node and 0 at all the other nodes, and completed by the 12 shape functions of the serendipity quadratic basis, each of which is 1 at one of the edges and 0 at all the nodes and other edges.

In structured adaptive mesh refinement (SAMR), the computational mesh is a hierarchy of levels of varying spatial mesh resolution. Each level is constructed from structured mesh components and corresponds to a uniform degree of mesh spacing. The levels are nested so that the coarsest level covers the entire computational domain Ω and each successively finer level covers a subdomain within the next coarser level. The cells on each mesh level are grouped into a collection of logically-rectangular regions called "patches". In this paper, we consider only fixed spatial mesh refinement, but mesh levels can also change in time as needed.

We use a refinement ratio of 2 in each coordinate direction between consecutive mesh levels. Thus, in 3D, a refined cell is divided into eight subcells of equal size ($2 \times 2 \times 2$). Instead of using special elements at coarse/fine interfaces, we allow hanging nodes and edges. Using such a refinement structure, coarse nodes and edges correspond to fine nodes at a fine–coarse interface (see Fig. 1). Fine edges at the boundary of a

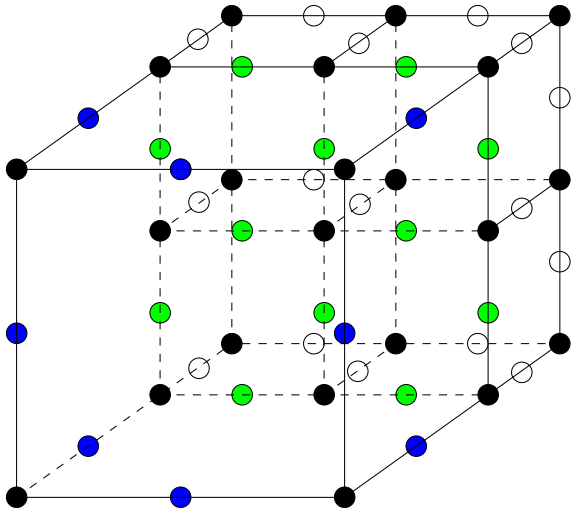

Fig. 1. Nodes and edges at coarse/fine interface in a quadratic FE method. Green edges values are defined by quadratic interpolation from three fine nodes values in edge direction.

finer level are "slave" edges; the value of a function at these edge points is defined by quadratic interpolation from three fine nodes values in the edge direction. This is necessary to ensure continuity at the fine–coarse interface. We also have "slave" nodes at the fine–coarse interface; these are fine nodes which do not correspond to coarse nodes or edges. No degree of freedom is associated to "slave" edges and nodes. While still somewhat non-standard, formulations based on such hanging nodes have been used by others, *e.g.* see Ref. [21].

One can view a finite element function in S_h as a linear combination of basis functions ϕ_i^e. Each basis function is continuous, has local support, takes the value 1 at exactly one node or edge, is a quadratic polynomial within each neighboring cell, and has the value 0 outside neighboring cells. No basis function is associated with a slave node or edge. Instead, to ensure continuity along the fine–coarse interface, basis functions are built as linear combinations of FE shape functions in neighboring cells, including FE shape functions associated to slave nodes and edges. Resulting basis functions have a support consisting of 3–8 cells, depending on whether they are centered on a node or an edge, and whether they are located in the interior of mesh level or at the interface between a coarse level and a fine level.

2.3. Mass and stiffness matrix assembly

Computing a matrix entry between two basis functions is done by computing a contribution from each cell to a given matrix element and summing the contributions over all relevant cells. In practice, cells contributions to the matrix representation of the Laplacian L and to the mass matrix M are computed analytically beforehand. Evaluating the entries of the stiffness matrix $K = L + Q$ requires a numerical integration for the term

$$q(\phi_i^e, \phi_j^e) = \int_\Omega \phi_i^e(\mathbf{x}) q(\mathbf{x}) \phi_j^e(\mathbf{x})\, d\mathbf{x} \approx (Q_h)_{ij} = q_h(\phi_i^e, \phi_j^e) = \sum_k w_k \phi_i^e(\mathbf{x}_k) q(\mathbf{x}_k) \phi_j^e(\mathbf{x}_k). \tag{5}$$

The coefficients w_k are the appropriate weights for the numerical quadrature formula used and $\mathbf{x}_k$ are quadrature points within the support of ϕ_i^e and ϕ_j^e. To note that this integration is not exact, we denote the numerical integration of the bilinear form $a(.,.)$ as $a_h(.,.)$.

We denote by $\tilde{S}_h$ the finite dimensional vector space of real functions defined by their values at the integration points. We define the operator $P^* : \tilde{S}_h \to S_h^*$ by

$$(f_h^*)_i := (P^* \tilde{f}_h)_i = \sum_{k=1}^{N_q} \phi_i^e(\mathbf{x}_k) \tilde{f}_h(\mathbf{x}_k) w_k. \tag{6}$$

This operator can be represented by an $N_e \times N_q$ matrix, where N_e is the size of the finite element basis and N_q the total number of integration points in Ω.

3. Nonlinear problem: electronic density and energy functional

As stated earlier, the operator q in the Kohn–Sham equations is actually nonlinear. It explicitly depends on the electronic density ρ. This density is a function of $u^{(i)}$, $i = 1, \ldots, N$ and is given by the general expression

$$\rho(\mathbf{x}) = 2 \sum_{i,j=1}^{N} (\overline{M}^{-1})_{ij} u_h^{(i)}(\mathbf{x}) u_h^{(j)}(\mathbf{x}) \tag{7}$$

for the case of N doubly occupied electronic orbitals. The entries of the matrix $\overline{M}$ are defined by

$$\overline{M}_{ij} = (u_h^{(i)}, u_h^{(j)}).$$

The electronic density defined by (7) is independent of the particular basis U chosen to represent the subspace $\mathcal{U}^{(N)}$. In the particular case of orthonormal functions $u_h^{(i)}$, $\overline{M}$ is the identity matrix.

In DFT, the potential operator q is the sum of three contributions:

$$q = V^{\text{ion}} + V^{\text{H}}[\rho] + V^{\text{xc}}[\rho]. \tag{8}$$

V^{ion} is a function of the atoms present in a simulation, and depends on their positions and species only. It is a linear operator representing the sum of radial functions (atomic local pseudopotentials) centered at the atomic positions. For the exchange and correlation potential V^{xc}, we use the so-called local density approximation (LDA). In this popular model, $V^{\text{xc}}[\rho](\mathbf{x}) = v^{\text{xc}}_{\text{LDA}}(\rho(\mathbf{x})) = \delta(\rho(\mathbf{x})\epsilon^{\text{xc}}_{\text{LDA}}(\rho(\mathbf{x})))/\delta\rho(\mathbf{x})$ for a given parametrized function $\epsilon^{\text{xc}}_{\text{LDA}}$ [22,23].

The Hartree potential V^{H} represents the Coulomb interaction between electrons, which is the electrostatic field generated by the electronic density ρ. It can be obtained by solving the Poisson equation

$$-\nabla^2 V^{\text{H}} = 4\pi\rho. \tag{9}$$

However, to avoid long range effects, we introduce a neutralizing charge ρ_{s} which cancels out ρ in the computational domain so that

$$\int_\Omega (\rho(\mathbf{x}) + \rho_{\text{s}}(\mathbf{x}))\,\mathrm{d}\mathbf{x} = 0. \tag{10}$$

In practice, we construct ρ_{s} as a sum of Gaussian charges ρ_{a} located at each atomic site $\mathbf{R}_{\text{a}}$ and which neutralizes each atomic pseudopotential individually,

$$\rho_{\text{a}}(\mathbf{r}) = -\frac{Z_{\text{a}}}{(\sqrt{\pi}r_c^a)^3}\exp\left(-\frac{|\mathbf{r}-\mathbf{R}_a|^2}{(r_c^a)^2}\right). \tag{11}$$

Here, Z_{a} is the valence charge of atom a, and r_c^a is a parameter chosen appropriately. One then solves the Poisson equation

$$-\nabla^2 V^{\text{C}} = 4\pi(\rho + \rho_{\text{s}}), \tag{12}$$

with zero Dirichlet or periodic boundary conditions to obtain

$$q = V^{\text{ion}} - V^{\text{s}} + V^{\text{C}}[\rho] + V^{\text{xc}}[\rho].$$

V^{s} is the Coulomb potential resulting from the charge distribution ρ_{s}. It is computed analytically by adding the solutions of the radial Poisson problem associated with each Gaussian charge:

$$v_{\text{s}}(\mathbf{r}) = \sum_{a=1}^{N_a}\frac{-Z_a}{|\mathbf{r}-\mathbf{R}_a|}\operatorname{erf}\left(\frac{|\mathbf{r}-\mathbf{R}_a|}{r_c^a}\right). \tag{13}$$

This procedure is standard in electronic structure calculations; *e.g.*, see [24].

The potential q is needed at the numerical integration points to evaluate the stiffness matrix coefficients. It is straightforward to compute V^{ion} at each integration point by simply evaluating the atomic potentials at these points. For V^{xc}, we need ρ at each integration point. This value is obtained by first evaluating each function $u_h^{(i)}$ at the integration points and then using Eq. (7).

To obtain the Coulomb potential V^{C} at the integration points, the process is more complicated. We first solve the Poisson problem (12) discretized using the same Finite Element approximation applied to the KS equations. We solve the linear system

$$L\mathbf{v}^{\text{C}} = \mathbf{f}^*, \tag{14}$$

and then we evaluate the FE solution

$$v_h^{\text{C}}(\mathbf{x}) = \sum_i (\mathbf{v}^{\text{C}})_i \phi_i^e(\mathbf{x}) \tag{15}$$

at the integration points. The loading vector $\mathbf{f}^*$ on the right-hand side of Eq. (14) is computed by a quadrature formula and its entries are defined by

$$(\mathbf{f}^*)_i = 4\pi(P^*(\rho+\rho_s))_i = 4\pi\sum_k \phi_i^e(\mathbf{x}_k)(\rho+\rho_s)(\mathbf{x}_k)w_k. \tag{16}$$

Here, the sum is over all the integration points in the support of ϕ_i^e.

We evaluate the KS energy of a FE solution according to the following definition:

$$E_h^{\mathrm{KS}} := \mathrm{Tr}(\overline{M}^{-1}\overline{L}) + \sum_k \epsilon^{\mathrm{xc}}(\rho(\mathbf{x}_k))\rho(\mathbf{x}_k)w_k + \sum_k (V^{\mathrm{ion}} - V^{\mathrm{s}})(\mathbf{x}_k)\rho(\mathbf{x}_k)w_k + \frac{1}{2}\sum_k (\rho+\rho_s)(\mathbf{x}_k)v_h^{\mathrm{C}}(\mathbf{x}_k)w_k - E_{\mathrm{self}} + E_{\mathrm{diff}}. \tag{17}$$

E_{self} and E_{diff} are quantities that depend only on the atomic positions and Gaussian neutralizing charges; they do not depend on the solution of the electronic structure problem [24]. The $N\times N$ matrix $\overline{L}$ is given by $\overline{L}_{ij} = b(u_h^{(i)}, u_h^{(j)})$. Since $\mathrm{Tr}(\overline{M}^{-1}\overline{L})$ and ρ do not depend on the representation U of the invariant subspace $\mathscr{U}^{(N)}$, the functional (17) is independent of the choice for U.

This expression is compatible with the discretized KS equations (3) in the sense that we have the following minimum principle:

Proposition 1. *The finite element approximation of the invariant subspace that minimizes the discretized energy functional E_h^{KS} (17) admits a basis $\{u_h^{(i)}\}_{i=1}^N$ which satisfies*

$$a_h(u_h^{(i)}, v_h) = \lambda_h^{(i)}(u_h^{(i)}, v_h), \quad \forall v_h \in S_h \tag{18}$$

for $i = 1, \ldots, N$, and q given by (8).

Proof. We can write E_h^{KS} as

$$E_h^{\mathrm{KS}} = E_h^{\mathrm{kin}} + E_h^{\mathrm{xc}} + E_h^{\mathrm{ion}} + E_h^{\mathrm{C}} - E_{\mathrm{self}} + E_{\mathrm{diff}},$$

where the various terms are defined by the obvious corresponding terms in Eq. (17). Since E_h^{KS} does not depend on the particular basis U of the invariant subspace $\mathscr{U}^{(N)}$, we can assume without loss of generality that $U^{\mathrm{T}}U = I$. In that case, $\overline{M} = I$ and

$$\rho(\mathbf{x}) = 2\sum_{i=1}^N (u_h^{(i)}(\mathbf{x}))^2.$$

We then examine the first-order variation of the various terms constituting E_h^{KS} in function of variations of $u_h^{(i)}$ subject to the orthonormality constraints

$$(u_h^{(i)}, u_h^{(j)}) = \delta_{ij}, \quad i,j = 1,\ldots,N. \tag{19}$$

For variations $\delta u_h^{(i)}$ of $u_h^{(i)}$, $i = 1,\ldots,N$, we have

$$\delta E_h^{\mathrm{kin}} = 2\sum_{i=1}^N b\left(u_h^{(i)}, \delta u_h^{(i)}\right). \tag{20}$$

We also have

$$\delta E_h^{\mathrm{ion}} = \sum_k (V^{\mathrm{ion}} - V^{\mathrm{s}})(\mathbf{x}_k)\delta\rho(\mathbf{x}_k)w_k = 2\sum_{i=1}^N \sum_k (V^{\mathrm{ion}} - V^{\mathrm{s}})(\mathbf{x}_k)u_h^{(i)}(\mathbf{x}_k)\delta u_h^{(i)}(\mathbf{x}_k)w_k. \tag{21}$$

Since by definition, $\delta(\rho(\mathbf{x})\epsilon^{\mathrm{xc}}(\mathbf{x})) = v^{\mathrm{xc}}(\mathbf{x})\delta\rho(\mathbf{x})$, we have

$$\delta E_h^{\mathrm{xc}} = \sum_k v^{\mathrm{xc}}(\rho(\mathbf{x}_k))\delta\rho(\mathbf{x}_k)w_k = 2\sum_{i=1}^N \sum_k v^{\mathrm{xc}}(\rho(\mathbf{x}_k))u_h^{(i)}(\mathbf{x}_k)\delta u_h^{(i)}(\mathbf{x}_k)w_k. \tag{22}$$

From (14)–(16), $v_h^{\mathrm{C}} = 4\pi L^{-1}P^*(\rho+\rho_{\mathrm{s}})$, and thus

$$\delta E_h^{\mathrm{C}} = 4\pi\sum_k (L^{-1}P^*(\rho+\rho_s))(\mathbf{x}_k)\delta\rho(\mathbf{x}_k)w_k = 8\pi\sum_{i=1}^N \sum_k (L^{-1}P^*(\rho+\rho_s))(\mathbf{x}_k)u_h^{(i)}\delta u_h^{(i)}(\mathbf{x}_k)w_k$$
$$= 2\sum_{i=1}^N \sum_k v_h^C(\mathbf{x}_k)u_h^{(i)}(\mathbf{x}_k)\delta u_h^{(i)}(\mathbf{x}_k)w_k. \tag{23}$$

E_{self} and E_{diff} do not depend on U and thus $\delta E_{\mathrm{self}} = 0$ and $\delta E_{\mathrm{diff}} = 0$.

Introducing the Lagrange multipliers $\lambda_h^{(i,j)}$ corresponding to the orthonormality constraints of Eq. (19), we obtain from Eqs. (20)–(23) and $\delta E_h^{\mathrm{KS}} = 0$, that the minimum of E_h^{KS} satisfies

$$a_h(u_h^{(i)}, \delta u_h^{(i)}) = \sum_{j=1}^{N} \lambda_h^{(i,j)}(u_h^{(j)}, \delta u_h^{(i)}) \ \forall \delta u_h^{(i)} \in S_h.$$

Proposition 1 follows directly from the fact that one can choose an orthonormal basis U such that $\lambda_h^{(i,j)} = 0$ for $i \neq j$. □

This proposition is very important not only to ensure that an iterative solver will be strictly converging from above toward the minimum energy, but also in the evaluation of the atomic forces. Indeed, the Hellman–Feynman theorem which states that forces can be determined with a single ground state calculation assumes that the discrete energy at the ground state is at a minimum with respect to any variation in the electronic wave functions [25].

4. Numerical solvers

4.1. Correction directions

Choosing to represent the invariant subspace we are looking for in a basis of general non-orthogonal functions, as opposed to a basis of eigenfunctions, allows more flexibility in the choice of iterative solvers. We use a *block accelerated preconditioned steepest descent* algorithm, independent of the basis U chosen. The basic ingredient for such an approach is the gradient of the functional (17), which is also the residual of Eq. (4). In the Ritz representation, and using a block matrix notation, this residual is given by

$$G = M^{-1}K\tilde{U} - \tilde{U}\tilde{\Lambda}, \tag{24}$$

where $\tilde{U}$ is the matrix whose columns are the Ritz vectors, and $\tilde{\Lambda}$ is a diagonal $N \times N$ matrix composed of the Ritz values. To avoid the inversion of M, we replace the residual (24) by an approximate residual

$$\tilde{G} = \tilde{M}^{-1}(K\tilde{U} - M\tilde{U}\tilde{\Lambda}) \tag{25}$$

where $\tilde{M}$ is a diagonal matrix approximating M. We use

$$(\tilde{M})_{ij} = \delta_{ij}(h_i)^3, \tag{26}$$

where h_i is the mesh spacing for the largest cell adjacent to node or edge i. The main purpose of this scaling matrix is to weight the coefficients according to the mesh refinement level. In our experience, this approximation works well for the quadratic hierarchical FE approach.

It is easy to see that for a general basis of non-orthogonal functions U, $\tilde{G}$ is given by

$$\tilde{G} = \tilde{M}^{-1}(KU - MU(\overline{M}^{-1}\overline{K}))$$

where $\overline{K}$ and $\overline{M}$ are $N \times N$ matrices given by

$$\overline{K}_{ij} = a_h(u_h^{(i)}, u_h^{(j)}), \quad \overline{M}_{ij} = (u_h^{(i)}, u_h^{(j)}).$$

Suppose we have a trial solution $U^{(k)}$. We can iteratively improve $U^{(k)}$ by simple corrections of the form

$$U^{(k+1)} = U^{(k)} - \eta T\tilde{G} \tag{27}$$

where T is a preconditioner and η a real positive coefficient. We propose an appropriate multigrid preconditioner in Section 4.4.

4.2. Anderson extrapolation scheme

As shown in [13], the convergence of a simple block preconditioned steepest descent algorithm with fixed shift can be improved significantly by using the extrapolation scheme of Anderson [26]. In a non-orthogonal basis U, we write this extrapolation scheme as

$$\overline{U}^{(\ell)} := U^{(\ell)} + \sum_{j=1}^{m} \theta_j^{(\ell)} (U^{(\ell-j)} - U^{(\ell)}), \tag{28}$$

where $U^{(\ell)}$ denote the trial solution at step ℓ. The coefficients $\theta_j^{(\ell)} \in \Re$ are defined as the solution of the linear system

$$\sum_{j=1}^{m} (R^{(\ell)} - R^{(\ell-i)}, R^{(\ell)} - R^{(\ell-j)}) \theta_j^{(\ell)} = (R^{(\ell)} - R^{(\ell-i)}, R^{(\ell)}), \quad i = 1, \ldots, m. \tag{29}$$

where $R^{(\ell)}$ is another iterative sequence associated to the same iterative process, for instance the residuals at steps $\ell, \ell - 1, \ldots$. The solution of Eq. (29) minimizes the norm of $\overline{R}^{(\ell)}$, defined as the extrapolation of $R^{(\ell)}$ according to the scheme (28). In practice, we use $(-T\tilde{G})$ as the sequence $R^{(\ell)}$ so that the solution of Eq. (29) minimizes the preconditioned approximate residual of the eigenvalue problem. Finally, the new trial solution is computed as

$$U^{(\ell+1)} = \overline{U}^{(\ell)} + \beta \overline{U}^{(\ell)}.$$

We usually choose a scalar value β between 0.5 and 1.

In the space of non-orthogonal functions representing a basis of an N-dimensional subspace $\mathscr{V}$, a natural scalar product would be

$$(V, W) = \mathrm{Tr}(\overline{M}^{-1} V^{\mathrm{T}} M W) \tag{30}$$

for V and W matrices representing N vectors of finite element coefficients, and $(\overline{M})_{ij} = (u_h^{(i)}, u_h^{(j)})$ at step ℓ. The scalar product in Eq. (30) is independent of any linear mixing within the basis we choose for $U^{(\ell)}$. It can however become computationally expensive for large problems. To reduce its cost, we drop the matrix $\overline{M}^{-1}$ in Eq. (30). The effect of this change is limited by orthonormalizing the trial solution at regular intervals, say every 20–30 iterations. Numerical tests show no major difference using this approximation.

Anderson's extrapolation scheme was designed to iteratively solve nonlinear equations [26]. In the case of an eigenvalue problem like the KS equations, the residual vanishes not only for the lowest eigenvalues we are interested in, but for any set of eigenvalues. In order to avoid the converging to undesired solutions, some care is required during the first few steps of the iteration when the trial solution is far from the ground state. In practice, we avoid problems by starting with a few – between 2 and 5 – iterations without extrapolation. Also, a "safety" interval is used for $\theta_j^{(\ell)}$ outside of which the extrapolation is turned off – for very large absolute values – or truncated to be inside the safety interval.

4.3. FAC Poisson solver

To solve for the electrostatic potential in DFT, we need an efficient solver for Eq. (14) discretized on an AMR grid. In this section, we present a multigrid fast adaptive composite (FAC) [27] Poisson solver appropriate for our quadratic FE discretization. An adaptation of this solver will be used in the next section as a preconditioner for the KS equations iterative solver. Our FAC solver is based on the underlying idea that one can precondition high-order finite elements with lower-order elements [28]. To do this, we decompose the quadratic FE space S_h into two complementary subspaces $S_h = V_h + W_h$, where W_h denotes the trilinear finite element space and V_h is the subspace of the functions in S_h with nodal interpolant $v^I = 0$. Let I_V and I_W be the natural injections from V_h and W_h, respectively, into S_h. In [28], a preconditioner B for the Laplacian operator L_h is proposed in the form

$$B := h^2 I_V I_V^{\mathrm{T}} + I_W L_W^{-1} I_W^{\mathrm{T}},$$

where L_W is the Laplacian operator in W_h and h is a typical mesh spacing for a quasi-uniform triangulation.

Based on the same idea of subspace decomposition, we have designed a multigrid FAC V-cycle to solve a Poisson problem discretized by hierarchical quadratic FE on an locally-refined mesh. The algorithm is as follows:

Algorithm 1. FAC Poisson solver for quadratic FE.

(1) Pre-smoothing: carry out ν_1 red–black Gauss–Seidel sweeps, starting with the edge degrees of freedom, followed by the node degrees of freedom.
(2) Coarsen residual r by dropping the edge degrees of freedom, leading to a trilinear FE residual equation $Lv = r$
(3) Solve iteratively trilinear FE residual equation on composite grid by FAC algorithm
   - V-cycles with Jacobi smoothing at each level
   - Solve trilinear FE problem on the finest uniform global mesh with standard multigrid solver
(4) Correct quadratic FE trial solution with solution v of trilinear FE residual equation.
(5) Post-smoothing: carry out ν_2 red–black Gauss–Seidel sweeps, starting with the edge degrees of freedom, followed by the node degrees of freedom.

The FAC algorithm attempts to iteratively solve the Poisson problem on a locally-refined mesh using a series of approximate solves (*i.e.*, smoothing steps) on the uniform mesh levels. When a mesh does not cover the whole computational domain, boundary conditions are provided by interpolating the current solution from the next coarser level. To treat the trilinear FE problem on the coarsest mesh level, we employ a linear solver from the *hypre* library [29].

4.4. Preconditioning strategy

The utility of a preconditioner to accelerate the iterative solution of the Kohn–Sham equations has long been recognized in the plane waves community [30]. Typically, an equation for the error δU on the trial invariant subspace representation $\tilde{U}$ can be written down as

$$K\delta U - M\delta U\tilde{\Lambda} = -K\tilde{U} + M\tilde{U}\tilde{\Lambda}, \tag{31}$$

where $\tilde{\Lambda}$ denotes the diagonal matrix of Ritz values associated with the trial solution $\tilde{U}$ – Ritz vectors – of the eigenvalue problem (4). In the domain of highly oscillatory functions, the Laplacian becomes the dominant part of the operator $K - \lambda M$ and the left-hand side of Eq. (31) can be approximated by $L\delta U$. This is valid for the high frequency component of δU, precisely the components that limit the length of the step in a simple steepest descent with a fixed shift algorithm. Thus, in Fourier space, one can design a diagonal preconditioner to rescale the weights of various frequency components and damp the high frequency components [30]. In real-space, a similar preconditioner can be designed using the multigrid method (*e.g.*, see [31]). For electronic structure calculations, this has been described in [24] for a finite differences scheme on a uniform mesh. Here, we apply a similar idea for a quadratic FE discretization on a locally-refined mesh.

We use the following preconditioner:

$$T = \hat{L}^{-1}\left(\frac{1}{\alpha}L - \tilde{M}\right) - \frac{1}{\alpha}I. \tag{32}$$

The operator $\hat{L}^{-1}$ is defined by its action on a vector $\mathbf{f}^* \in S_h^*$. The computation of $\mathbf{u} = \hat{L}^{-1}\mathbf{f}^*$ is accomplished by applying one $V(2, 2)$ multigrid cycle for the linear system $L\mathbf{u} = \mathbf{f}^*$ using the FAC scheme described in Section 4.3, but visiting only the levels with mesh spacing $h \leqslant H$. The coarse level problem ($h = H$) is only approximately solved by four Jacobi smoothing steps. The parameter α is an approximation of the largest eigenvalue for the KS operator on the level with mesh spacing H. The parameter H is problem dependent and is chosen heuristically by numerical experiments. To understand the form of this preconditioner, we examine its effect in the low and high frequency domain of the FE space. For typical elliptic equations like the Poisson problem, smoothing sweeps in a multigrid cycle reduce the error in the frequency range associated to each grid level while leaving lower frequency error components almost unchanged. Thus, for the high frequency components, $\hat{L}^{-1}L \sim I$ and $T \sim -\hat{L}^{-1}\tilde{M}$, effectively damping the high frequency components of $\mathbf{f}^*$. In the low frequency domain (wavelength $\gg H$), $\hat{L}^{-1} \sim 0$ and T becomes a simple steepest descent damping factor α^{-1} for a mesh spacing H.

5. Implementation

The algorithms described in this paper are implemented using the tools provided by SAMRAI, an object oriented parallel software infrastructure for general AMR applications on structured grids [19]. SAMRAI is an object-oriented C++ software library developed in the Center for Applied Scientific Computing at Lawrence Livermore National Laboratory. SAMRAI simplifies the implementation of SAMR applications by providing general tools to build dynamic locally-refined mesh hierarchies and manipulate data, such as arrays of node, edge, and cell quantities, on those hierarchies. Each mesh level is decomposed into patches that are distributed among processors using load balancing algorithms also provided by the library. Parallel data management and communication operations, such as exchanging data between patches on different mesh levels, is provided by the library. In our FE formulation, we must properly account for nodes and edges around patch boundaries that belong to more than one patch, including patches on different levels. This occurs, for example, when computing a vector in S_h^*. SAMRAI provides tools to sum contributions from all the patches and get a common node or edge value.

In our KS solver implementation, the stiffness and mass matrices are never stored. Instead the matrix-vector application is defined using the non-zero matrix elements. In our SAMR approach, all the cells are identical at each level and matrix elements for the Laplacian of the Mass matrix depend on the refinement level only. For the potential operator, represented by the matrix Q_h, we use Eq. (5) with 27 Gauss quadrature points per cell ($O(h^6)$).

Typically, we treat slave nodes and edges as additional unknowns and enforce continuity by additional equations. From a practical point of view, operations on individual patches are carried out uniformly over all the cells within a patch. Then in a postprocessing step, values at the boundaries are corrected to take into account the continuity constraints. It means that when computing elements of vectors in S_h^*, contributions from a coarser or finer level (*e.g.*, at coarse–fine mesh boundaries) or from neighboring patches at the same refinement level need to be summed up. In particular, values attributed to slave nodes or edges FE basis functions are used to compute contributions associated to coarse–fine interface basis functions.

6. Numerical results

6.1. Poisson problem

In this section, we apply the FAC algorithm presented in Section 4.3 on a test Poisson problem. This problem is central in electronic structure calculations to compute the electrostatic interactions (Eq. (14)). In our algorithm, this is also relevant for the efficiency of the preconditioner. We evaluate the FAC algorithm convergence rate for a quadratic FE discretization of

$$\begin{cases} -\triangle u = f, & \text{in } \Omega = (-2.5, 2.5) \times (-2.5, 2.5) \times (-2.5, 2.5) \\ u = 0, & \text{on } \partial\Omega. \end{cases} \tag{33}$$

The right-hand side f is defined by the radial function

$$g(r) = \frac{1}{r_c^3 \pi^{3/2}} \left(8\mathrm{e}^{-4r^2/r_c^2} - \mathrm{e}^{-r^2/r_c^2} \right)$$

with the origin chosen at (0, 0, 0). The integral of the function g over $\mathscr{R}^3$ is 0. In an infinite domain ($\mathscr{R}^3$), this problem would admit the exact solution

$$u(r) = \frac{1}{4\pi r} (\mathrm{erf}(2r/r_c) - \mathrm{erf}(r/r_c))$$

which quickly decays to zero. In Table 1, we present convergence results for the FAC algorithm. The uniform (coarse) mesh problem is approximately solved by *hypre* asking for a reduction of the residual by a factor 0.1. The results demonstrate a mesh-independent convergence rate, as well as very similar convergence rates for locally refined and uniform meshes.

Table 1
Number of FAC-$V(2, 2)$-cycles needed to reduce the initial error by a factor 10^{-8} in H^1 norm for various uniform meshes and number of refinement levels

Uniform global mesh	No. refinement levels		
	0	1	2
$16 \times 16 \times 16$	12	12	13
$32 \times 32 \times 32$	12	12	12
$64 \times 64 \times 64$	12	11	12

The mesh is locally refined in 25% of the total volume for the first level and 5% for the second level.

6.2. Electronic structure calculations

As test applications for DFT calculations, we have chosen Beryllium clusters made of 4 and 17 atoms. Clusters calculations benefit from local mesh refinement since they usually require computational domains much larger than the cluster itself to simulate the surrounding vacuum. Fig. 2 illustrates a Be_4 cluster calculation on a hierarchy made of 2 grid levels. In the present work, we use the pseudopotential of Beryllium parameterized by Goedecker et al. [32].

We verified our numerical algorithm by computing the total energy on a domain with periodic boundary conditions using various mesh spacings and observing convergence towards the energy computed by an independent plane waves code. The convergence rate for the energy is $O(h^4)$ in the mesh spacing variable h, which is in agreement with the theoretical convergence rate for the eigenvalues [33]. This is shown in Fig. 3. We also observe improved efficiency resulting from local mesh refinement. The results obtained with locally-refined meshes provide the same accuracy as the results obtained on uniform meshes corresponding to refined regions meshes with a six-fold reduction in number of degrees of freedom.

We tested our multigrid preconditioner on this same application, using various Finite Elements meshes. For all the calculations presented in Fig. 4, the coarsest grid in the multigrid preconditioner was set to $16 \times 16 \times 16$ cells. We show results using discretizations on three different uniform meshes – $16 \times 16 \times 16$, $32 \times 32 \times 32$, and

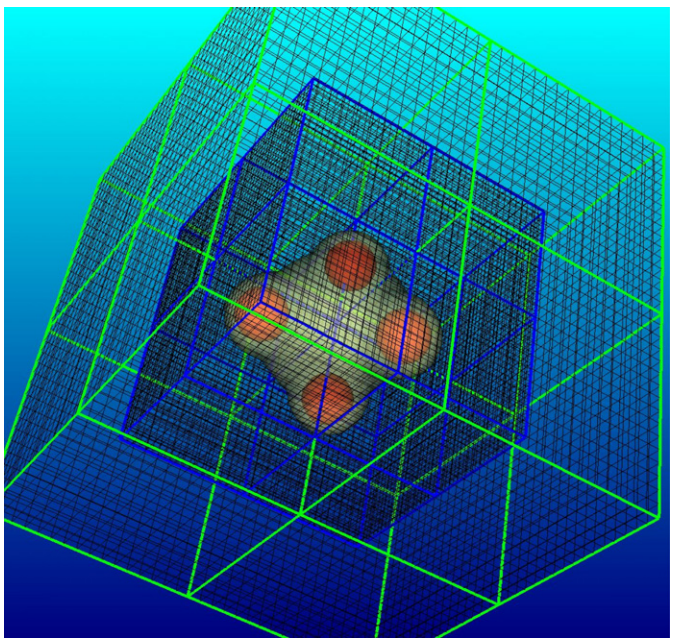

Fig. 2. Electronic structure calculation of Be_4 cluster on composite mesh. The two structured mesh levels and their decomposition in eight patches are shown. An isosurface of the electronic density is plotted, surrounding the four Be atoms represented by spheres.

$64\times\ 64\times 64$ – as well as with the first two meshes refined locally in 25% of the volume. The results were obtained using the block Anderson extrapolation scheme described in Section 4.2, with $m = 2$, and multigrid preconditioning. In the iterative process, the non-linear potential is updated at each step, *i.e.* after every update of the wave functions, to ensure that the true gradient is always used. The numerical results show a convergence rate nearly independent of the discretization mesh and the use of local refinement. This demonstrate the efficiency of the multigrid preconditioner (see Fig. 5).

We measure the strong parallel scaling on a larger problem: a Be cluster made of 17 atoms. The solution to this problem involves 34 functions that we represent on a mesh of $64\times 64\times 64$ cells, refined by a factor 2 in 1/8 of the volume – thus the fine level mesh is also made of $64\times 64\times 64$ cells. For this problem, each level is divided into a number of patches corresponding to the number of processors. The size of the patches varies

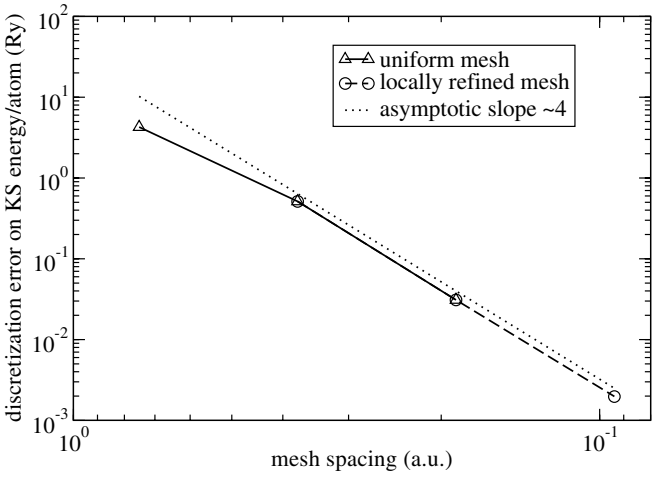

Fig. 3. Discretization error as a function of mesh spacing for a Be_4 cluster calculation with periodic boundary conditions. The reference result is an independent fully converged Plane Waves calculation.

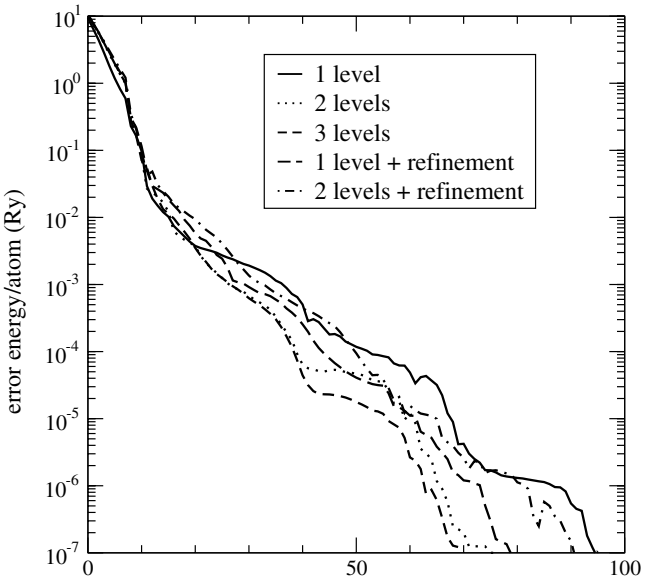

Fig. 4. Convergence of KS energy during iterative minimization. Test case: Be_4 cluster, $N = 8$.

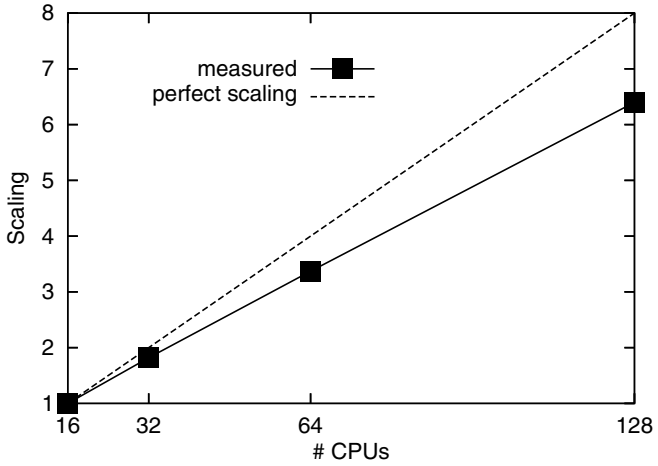

Fig. 5. Parallel scaling for fixed size problem (Be_{17} cluster). The coarse level is made of $64 \times 64 \times 64$ cells. The mesh is refined in 1/8 of the volume.

from $16 \times 32 \times 32$ cells on 16 CPUs to $8 \times 16 \times 16$ cells on 128 CPUs. Going from 8 processors to 128, we measure a parallel efficiency of about 80% which is quite satisfactory for this fixed size problem.[3]

7. Concluding remarks

In this paper, we have presented a finite element method for density functional theory calculations on structured locally-refined meshes. We have also described an efficient multilevel solver for this problem. The algorithm complexity essentially scales like $O(M \times N^2)$, where M is the size of the finite element basis used to discretize the problem, and N is the number of electronic wave functions to compute (see computation of $\tilde{G}$, $\overline{M}$, $\overline{K}$ for instance).

As mentioned in the introduction, one interesting aspect of any real-space discretization is the possibility of representing the electronic structure in terms of spatially localized orbitals to achieve an $O(N)$ computational complexity. Representing localized orbitals in a finite element basis with adaptive mesh refinement such as presented above instead of strictly localized functions – such as proposed in Ref. [12], *e.g.* – is an idea that we are currently studying.

Finally, to put the method presented above into perspective, we should mention some comparison with the previous work of one of us (J.-L.F.) using finite differences. For total energy calculations on uniform meshes, the quadratic finite element method appears to achieve a very similar accuracy and convergence rate as a standard 4th-order finite difference calculation – measured per degree of freedom. For the same number of degrees of freedom, our present (not fully optimized) implementation of the finite element approach is somewhat slower than our finite difference code, but of the same order of magnitude (factor $\sim$2). Obviously, using local mesh refinement reduces the number of degrees of freedom and easily compensates for that difference in many cases.

Acknowledgments

We thank B. Lee, J. Pask and F. Gygi for useful discussions. The numerical simulations were carried out on the Lawrence Livermore National Laboratory Linux cluster MCR. This work was performed under the auspices of the US Department of Energy by University of California Lawrence Livermore National Laboratory Under Contract No. W-7405-Eng-48.

[3] The calculations were carried out on the Lawrence Livermore National Laboratory 1024 nodes Linux cluster MCR.

References

[1] R.M. Dreizler, E.K.U. Gross, Density Functional Theory, Springer, Berlin, 1990.
[2] E. Cances, M. Defranceschi, W. Kutzelnigg, C. LeBris, Y. Maday, Computational chemistry: a primer, in: P. Ciarlet, C. Le Bris (Eds.), Handbook of Numerical Analysis, vol. X, Elsevier Science, 2003, pp. 3–270.
[3] S.C. Brenner, L.R. Scott, The Mathematical Theory of Finite Element Methods, Springer-Verlag, New York, 2002.
[4] E. Tsuchida, M. Tsukada, Electronic-structure calculations based on the finite-element method, Phys. Rev. B 52 (8) (1995) 5573–5578.
[5] E. Tsuchida, M. Tsukada, Large-scale electronic-structure calculations based on the adaptive finite-element method, J. Phys. Soc. Jpn. 67 (11) (1998) 3844–3858.
[6] E. Tsuchida, M. Tsukada, Adaptive finite-element for electronic-structure calculations, Phys. Rev. B 524 (11) (1996) 7602.
[7] J.E. Pask, P.A. Sterne, Finite elements methods in ab initio electronic structure calculations, Model. Simulat. Mater. Sci. Eng. 13 (2005) 71–96.
[8] P. Havu, V. Havu, M.J. Puska, M.H. Hakala, A.S. Foster, R.M. Nieminen, Finite-element implementation for electron transport in nanostructures, J. Chem. Phys. 124 (2006) 054707.
[9] T. Beck, Real-space mesh techniques in density-functional theory, Rev. Mod. Phys. 72 (4) (2000) 1041–1080.
[10] E.L. Briggs, D.J. Sullivan, J. Bernholc, Real-space multigrid-based approach to large-scale electronic structure calculations, Phys. Rev. B 54 (20) (1996) 14362–14375.
[11] E. Hernandez, M.J. Gillan, Self-consistent first-principles technique with linear scaling, Phys. Rev. B 51 (15) (1995) 10157–10160.
[12] J.-L. Fattebert, J. Bernholc, Towards grid-based O(N) density-functional theory methods: optimized non-orthogonal orbitals and multigrid acceleration, Phys. Rev. B 62 (3) (2000) 1713–1722.
[13] J.-L. Fattebert, F. Gygi, Linear scaling first-principles molecular dynamics with controlled accuracy, Comput. Phys. Commun. 162 (2004) 24–36.
[14] F. Gygi, G. Galli, Real-space adaptive-coordinate electronic-structure calculations, Phys. Rev. B 52 (4) (1995) 2229–2232.
[15] N.A. Modine, G. Zumbach, E. Kaxiras, Adaptive-coordinate real-space electronic-structure calculations for atoms, molecules, and solids, Phys. Rev. B 55 (16) (1997) 1337–1340.
[16] J.-L. Fattebert, Finite difference schemes and block Rayleigh Quotient Iteration for electronic structure calculations on composite grids, J. Comput. Phys. 149 (1999) 75–94.
[17] E. Tsuchida, M. Tsukada, Real space approach to electronic-structure calculations, Sol. State Com. 94 (1) (1995) 5–8.
[18] S. Kohn, J. Weare, M. Ong, S. Baden, Software abstractions and computational issues in parallel structured adaptative mesh methods for electronic structure calculations, in: Proceedings of the of Workshop on Structured Adaptative Mesh Refinement Grid Methods, Minneapolis, 1997.
[19] R. Hornung, S. Kohn, Managing application complexity in the SAMRAI object-oriented framework, Concurr. Comput.: Pract. Ex. 14 (2002) 347–368.
[20] G. Dhatt, G. Touzot, The Finite Element Method Displayed, J. Wiley & Sons, New York, 1984.
[21] R. Becker, M. Braack, Multigrid techniques for finite elements on locally refined meshes, Numer. Linear Algebra Appl. 7 (2000) 363–379.
[22] D.M. Ceperley, B.J. Alder, Ground state of the electron gas by a stochastic method, Phys. Rev. Lett. 45 (1980) 566–569.
[23] J.P. Perdew, A. Zunger, Self-interaction correction to density-functional approximations for many electrons systems, Phys. Rev. B 23 (1981) 5048–5079.
[24] J.-L. Fattebert, M. Buongiorno Nardelli, Finite difference methods for ab initio electronic structure and quantum transport calculations of nanostructures, in: P. Ciarlet, C. Le Bris (Eds.), Handbook of Numerical Analysis, vol. X, Elsevier Science, 2003, p. 571.
[25] R. Feynman, Forces in molecules, Phys. Rev. 56 (1939) 340–343.
[26] D.G. Anderson, Iterative procedures for nonlinear integral equations, J. Ass. Comput. Mach. 12 (4) (1965) 547–560.
[27] S. McCormick, J. Thomas, The fast adaptive composite grid (FAC) method for elliptic equations, Math. Comp. 46 (174) (1986) 439–456.
[28] S. Brenner, Preconditioning complicated finite elements by simple finite elements, SIAM J. Sci. Comput. 17 (5) (1996) 1269–1274.
[29] R. Falgout, U. Yang, hypre: A library of high performance preconditioners, in: P. Sloot, C. Tan, J. Dongarra, A. Hoekstra (Eds.), Computational Science – ICCS 2002 Part III, Lecture Notes in Computer Science, vol. 2331, Springer-Verlag, 2002, pp. 632–641.
[30] M.P. Teter, M.C. Payne, D.C. Allan, Solution of Schrödinger's equation for large systems, Phys. Rev. B 40 (18) (1989) 12255–12263.
[31] J.H. Bramble, J.E. Pasciak, J. Xu, Parallel multilevel preconditioners, Math. Comp. 55 (191) (1990) 1–22.
[32] S. Goedecker, M. Teter, J. Hutter, Separable dual-space Gaussian pseudopotentials, Phys. Rev. B 54 (3) (1996) 1703–1710.
[33] G. Strang, G.J. Fix, An Analysis of the Finite Element Method, Prentice-Hall, Englewood Cliffs, NJ, 1973.