Journal of Computational Physics 228 (2009) 8367–8379

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Efficient $O(N)$ integration for all-electron electronic structure calculation using numeric basis functions

V. Havu [a,b,*], V. Blum [b], P. Havu [b], M. Scheffler [b]

[a] Department of Applied Physics, Helsinki University of Technology – TKK, Finland
[b] Fritz Haber Institute of the Max Planck Society, Berlin, Germany

ARTICLE INFO

Article history:
Received 24 November 2008
Received in revised form 13 July 2009
Accepted 12 August 2009
Available online 21 August 2009

PACS:
02.60.Jh
71.15.Ap
71.15.Dx

Keywords:
Electronic structure theory
Density functional theory
Atom-centered basis functions
Numerical integration grid
Spatial partitioning

ABSTRACT

We consider the problem of developing $O(N)$ scaling grid-based operations needed in many central operations when performing electronic structure calculations with numeric atom-centered orbitals as basis functions. We outline the overall formulation of localized algorithms, and specifically the creation of localized grid batches. The choice of the grid partitioning scheme plays an important role in the performance and memory consumption of the grid-based operations. Three different top-down partitioning methods are investigated, and compared with formally more rigorous yet much more expensive bottom-up algorithms. We show that a conceptually simple top-down grid partitioning scheme achieves essentially the same efficiency as the more rigorous bottom-up approaches.

1. Introduction

Computational electronic structure theory (EST) (e.g., density functional theory [1], Hartree–Fock, or many-body perturbation theories such as MP2 and *GW*) is playing an increasingly prominent role in science and technology. Traditionally, a large variety of discretization methods via basis sets has been available for the Kohn–Sham equations [2], on which many practical implementations are based. In particular, a successful basis choice employed in a variety of all-electron implementations [3–8] is numeric atom-centered orbitals (NAOs). These offer an efficient prescription that can be used for accurate full-potential, all-electron calculations of periodic and non-periodic systems on equal footing.

The present paper examines the real-space, three-dimensional integration grid infrastructure needed to optimally handle NAO-based all-electron EST. While all algorithms are described and tested with NAOs in mind, we note that the same algorithms and basic observations will also apply to many other all-electron basis set prescriptions for EST. Indeed, similar grid-based operations are needed in several widely used and actively developed codes (e.g., DMol [5], TURBOMOLE [9], ADF [10], Gaussian [11], and many others). $O(N)$ integration formalisms have also been given in the wider $O(N)$ community, e.g., in the context of the pseudopotential-based Siesta [12], ONETEP [13], OpenMX [14], or Conquest [15] codes.

* Corresponding author. Address: Helsinki University of Technology, Department of Applied Physics, P.O. Box 1100, FI-02015 TKK, Finland.
E-mail address: Ville.Havu@tkk.fi (V. Havu).

0021-9991/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2009.08.008

The use of NAOs entails inevitably a real-space grid that is required to perform the basic operations of EST: integration of the Hamilton matrix, update of the electron density, and solution of the electrostatic potential (we do not specifically address the latter in this paper). In practice, these grid-based operations are the main cost associated with NAOs for all but the largest systems ($\gtrsim$ 500–1000 atoms). The reason is that the number of NAOs needed to achieve an accurate solution is relatively low ($\lesssim$50 basis functions/atom for meV/atom total-energy accuracy [5,8]) compared to the number of basis functions required in other methods. Consequently, even an $O(N^3)$ solution of the discretized Kohn–Sham eigenvalue problem does not dominate in all but the largest systems—and even there, the development of $O(N)$ methods that circumvent the Kohn–Sham eigenvalue solution is an active research area, and appropriate implementations (e.g., Refs. [12,14–16]) are now available for many system classes.

By using explicit confining potentials in their construction [6,7,17–23], NAOs give a natural platform to make the grid-based operations scale $O(N)$ by properly localizing all basis functions. Specifically, the good and eventually nearly linear-scaling performance of NAOs in the integrations and in the update of the electron density results from spatial localization of the basis functions combined with *a careful division of the grid into spatially localized regions* for the grid-based operations [11,15,24,25]. In this paper we (1) reiterate that there is a scheme that via basis and grid localization results in nearly linear-scaling grid operations for NAOs, (2) study the problem of partitioning the grid into localized batches in detail using three top-down grid partitioning methods, and (3) show that a formally more rigorous class of bottom-up grid partitioning methods is too expensive for our purposes. Finally, we compare the computational efficiency of these methods for different prototypical applications (polyalanine chains, Cu clusters, Au surfaces and fcc Al bulk).

Before moving on to the detailed description and analysis of our approach, we show what properly implemented [8] basis and grid localization schemes should achieve in terms of scaling, in the case of NAOs. Fig. 1 shows the computational scaling for a series of real molecules, specifically a series of fully extended conformations of polyalanine peptide molecules. In Fig. 1(a), a single alanine aminoacid is shown, identified by its CH_3 side group. Fig. 1(b) shows how several alanine units are linked together into a peptide chain. The conformation shown here is known as "fully extended." Because it is essentially linear, the distance between successive basis function centers in this system class grows rapidly with length. We emphasize that we here choose this system class specifically because it is well suited to demonstrate the correct scalability limit of our implementation *in principle*. Finally, Fig. 1(c) shows the scaling as the polyalanine chain length increases from a single

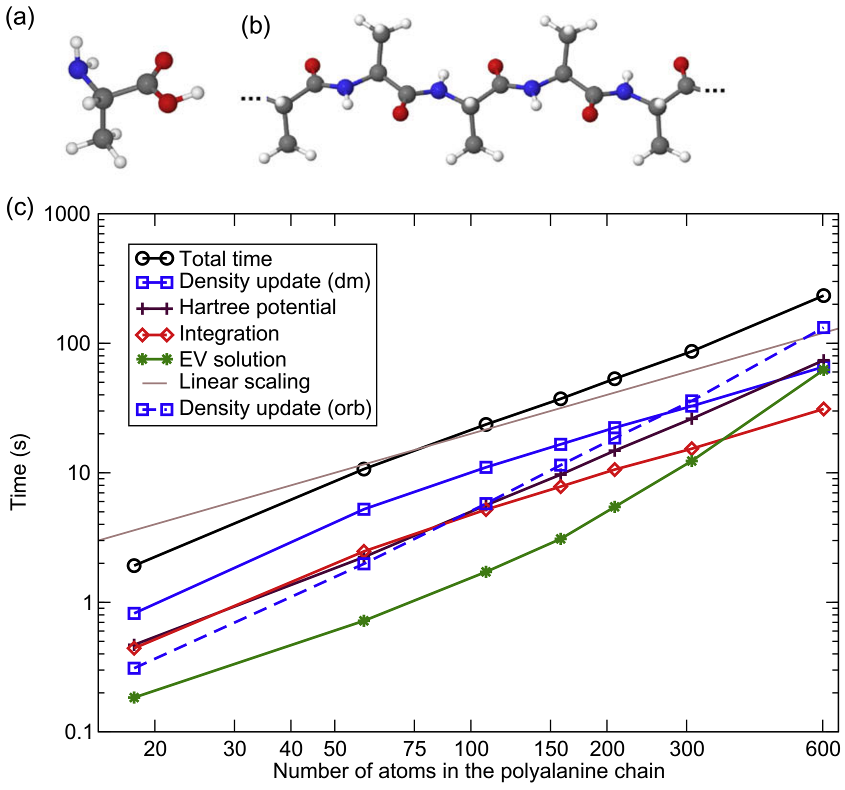

Fig. 1. Scalability for fully extended polyalanine molecules with respect to system size. (a) Single alanine aminoacid. (b) Segment of a fully extended polyalanine chain (five CH_3 residues). (c) Timings in seconds for one self-consistency iteration of finite, fully extended polyalanine chains of increasing length. The scaling exponents of the operations for the four largest systems are: Total: $O(N^{1.5})$, electron density update using Eq. (6): $O(N^{1.0})$ and using Eq. (5): $O(N^{1.8})$ solution of the Hartree potential with an atom-wise multipole decomposition: $O(N^{1.5})$ integration of the Hamilton matrix: $O(N^{1.0})$, ScaLAPACK based solution of the eigenvalues and eigenvectors $O(N^{2.2})$.

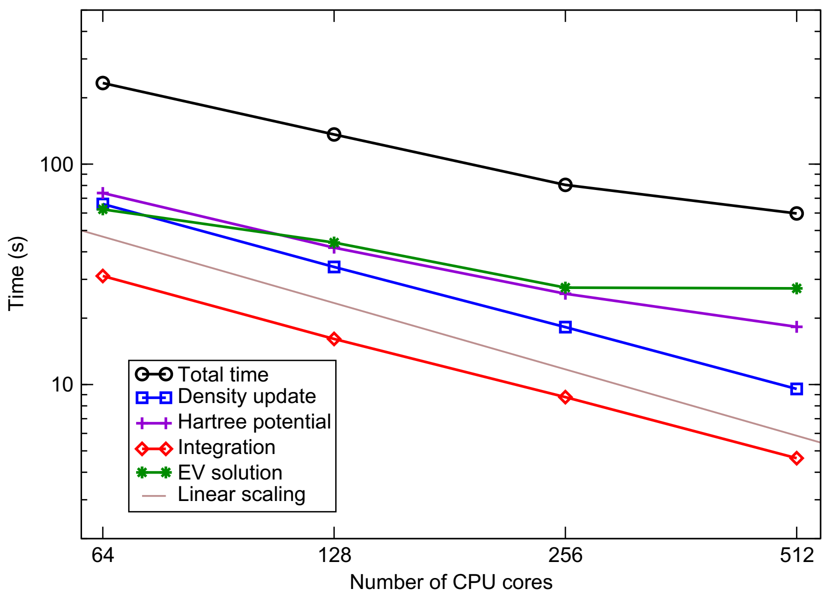

Fig. 2. Scalability for fully extended polyalanine of sixty residues (603 atoms) with respect to number of CPU cores. Timings are in seconds for one self-consistency iteration.

Table 1
Basis and grid settings for all elements as used in the calculations. The basis sizes given are for ionic and hydrogen-like functions *in addition* to the minimal basis of occupied atomic orbitals. The notation $\langle n\rangle\langle l\rangle$ for the basis (e.g., 3s) means that for each atom there are $\langle n\rangle$ radial functions that correspond to an angular momentum denoted by $\langle l\rangle$, in addition to the minimal basis of atomic valence orbitals. If $\langle n\rangle = 1$, it is omitted from the notation.

	H	C	N	O	Cu	Au	Al
Basis: minimal +	3s2p	2s2p2df	2s2p2df	2s2p2df	spdfg	spdf	spd
r_{cutoff} (Å)	5.5	5.5	5.5	5.5	6.0	6.0	6.0
Min. angular grid	110	110	110	110	50	50	110
Max. angular grid	590	590	590	590	590	590	302
n_{radial}	49	69	71	73	107	147	41

residue of 18 atoms (the terminations are included in this count) up to a molecule of 60 residues with 603 atoms. Fig. 2 shows the parallel scalability for our approach in the case of the 603 atom fully extended polyalanine chain. We emphasize that all computational settings reflect tightly converged production settings as detailed in Table 1. It can be seen that the grid-based operations (update of the electron density using a density matrix and integration of the Hamilton matrix) are nearly linear scaling and the overall scaling reaches the value $O(N^{1.5})$ for the four largest systems. Since our focus is on those methodological steps that involves localized basis functions in real-space, the solution of the Hartree potential and the eigenvalue problem are shown for illustration only, but have not yet been adapted for the same rigorous scaling requirements as the (without localization) dominant basis-dependent steps. All our results are computed using the FHI-aims computer code, a recently developed NAO-based implementation for electronic structure calculations [8]. The timings were obtained on an IBM p575 Power5+ system using 16 CPU cores, except for Figs. 1(c) and 2 that were calculated on a Cray XT/4, and Fig. 7 that was computed on IBM Power6 575 system with 32 CPU cores.

2. Localization of the basis functions

The general form of a NAO basis function is given by

$$\varphi_{i,lm} = \frac{u_i(r)}{r} Y_{lm}(\theta, \phi), \tag{1}$$

where Y_{lm} is a spherical harmonic and $u_i(r)$ is a real-valued function representing the radial shape of the atom-centered basis function. In our actual implementation, we use as angular momentum functions Y_{lm} the real and imaginary parts of the complex spherical harmonics, leading to only real-valued functions $\varphi_{i,lm}$ (the full index set i,lm is condensed to i below). The radial function, $u_i(r)$, is usually taken to be a solution to a Schrödinger equation of atomic, ionic or hydrogenic type, and is localized using a smooth cutoff potential so that $u_i(r)$ becomes zero beyond some cutoff radius, r_{cutoff} (see Ref. [8] for details).

To illustrate the effect of this localization, let us consider the two operations we are focusing on: (1) the integration of the Hamilton matrix and (2) the update of the electron density.

(1) The Hamilton matrix, h, has the entries $h_{ij} = \int_{\mathbb{R}^3} \varphi_i \hat{h} \varphi_j \, dr^3$ where φ_i and φ_j are the basis functions. If implemented without localization, the integration leads to $O(N^3)$ scaling of the computation, since for each pair of basis functions we need to run over all the grid. However, this can be avoided by enforcing a localization of the basis functions using a cutoff potential. As the spatial extent of the considered system grows, the number of non-zero basis functions that need to be consider for each grid point levels off to a constant value, leaving only the number of grid points as a growth factor for the complexity of the operation.

(2) The electron density is defined by

$$n(\mathbf{r}) = \sum_{k=1}^{n_{states}} f_k |\psi_k(\mathbf{r})|^2, \tag{2}$$

where f_k's are the occupation numbers, and the sum runs over all Kohn–Sham eigenstates, which are given by

$$\psi_k(\mathbf{r}) = \sum_{i=1}^{n_{basis}} c_i^k \varphi_i(\mathbf{r}) \tag{3}$$

so that

$$\begin{aligned} n(\mathbf{r}) &= \sum_{k=1}^{n_{occ}} f_k \sum_{i=1}^{n_{basis}} c_i^k \varphi_i(\mathbf{r}) \sum_{j=1}^{n_{basis}} \left(c_j^k\right)^* \varphi_j(\mathbf{r}) & (4)\\ &= \sum_{k=1}^{n_{occ}} f_k \sum_{i=1}^{n_{basis}} \sum_{j=1}^{n_{basis}} c_i^k \varphi_i(\mathbf{r}) \left(c_j^k\right)^* \varphi_j(\mathbf{r}) & (5)\\ &= \sum_{i=1}^{n_{basis}} \sum_{j=1}^{n_{basis}} D_{ij} \varphi_i(\mathbf{r}) \varphi_j(\mathbf{r}). & (6) \end{aligned}$$

Here $D_{ij} = \sum_{k=1}^{n_{occ}} f_k c_i^k (c_j^k)^*$ is the density matrix. Again, if implemented naively with no regard to localization, Eq. (4) leads to an $O(N^3)$ operation count [$O(N^3)$ for the setup of each occupied Kohn–Sham orbital at each grid point, followed by a scalar product of Kohn–Sham orbitals at each grid point—the latter is an $O(N^2)$ operation]. Taking into account localization, the number of non-zero basis functions at each grid point will eventually level off as for the Hamiltonian as the system size grows, but n_{occ} will not level off, so that the complexity will be reduced to an $O(N^2)$ algorithm. After summing up the density matrix, the actual *grid-based* density update Eq. (6) will be a linear-scaling operation apart from the cost of setting up the density matrix, which is an $O(N^2)$ operation but has only a small prefactor compared to the grid-based operation in the case of NAOs. For very large systems, the density matrix D_{ij} becomes sparse, and is usually stored in a sparse storage format. Although we do not presently pursue such an approach, we note that the sparsity of D_{ij} can also be used directly to bypass the Kohn–Sham orbitals c_j^k, leading to $O(N)$ scaling by imposing localization conditions (see Refs. [15,26] and references therein).

Note that the density matrix based density update Eq. (6) will be superior to the orbital based Eq. (4) only when the number of occupied orbitals becomes larger than the number of basis functions that are locally non-zero. Due to localization there will be such a turning point as the system size grows. This is also visible in Fig. 1, which shows the cross-over point (dashed and solid blue[1] lines and square symbols) around 300 atoms.

3. Grid operations

To illustrate the role that is played by the grids in NAO-based calculations, we return to the example of computing the Hamilton matrix. In practice, the integration grid used for evaluating an approximation to h_{ij} is composed of overlapping atom-centered grids, where each radial shell of points is a Lebedev grid [27–29]. The radial positions of the shells on the axis ranging from zero to infinity are taken to be $r_s = -\log\left(1 - \left(\frac{s}{n_{radial}+1}\right)^2\right)$, $s = 1, \ldots, n_{radial}$ [30]. For other options and extensive tests on the grids in conjunction with Gaussian basis functions we refer to Ref. [9].

The overlapping grids are then partitioned using a *partitioning of unity* method [5,31], i.e., each atom-centered grid (centered at a site α) is associated with a weight function $p_\alpha(\mathbf{r})$ ($\alpha = 1, \ldots, N_{grids}$) such that $\sum_\alpha p_\alpha(\mathbf{r}) = 1$ for every point $\mathbf{r} \in \mathbb{R}^3$. On this grid the exact value of h_{ij} is approximated using a quadrature over the discrete integration points $\{\mathbf{r}\}$, i.e., we set $h_{ij} = \sum_\mathbf{r} w(\mathbf{r}) f(\mathbf{r})$ where $w(\mathbf{r})$ is the combined weight of radial, angular and partition weights at $\mathbf{r}$, $f(\mathbf{r}) = (\varphi_i \hat{h} \varphi_j)(\mathbf{r})$ is the integrand and the summation runs over all grid points.

[1] For interpretation of color in Figs. 1-7, the reader is referred to the web version of this article.

In practice the quadrature is not done one grid point at a time but larger batches of points, B_v, are used [10]. This allows the use of matrix–matrix products for computing the matrix elements h_{ij}. The full algorithm (closely related to the ones in [10,11,25,32]) is presented in detail in Algorithm 1. Three observations are imminent at this point. First, for Step 1a to yield as small number of non-zero basis functions as possible it is important to have grid batches that are as localized as possible. Second, Step 1a scales technically $O(N^2)$ (e.g., compute the square of the distance of each atom from each grid point) but has an extremely small prefactor. In fact, the indices of the non-zero basis functions for each batch can be computed at the initialization stage for the current atomic configuration and stored thereafter if desired, requiring only one $O(N^2)$ step for each atomic configuration. Third, the loop over the batches is an embarrassingly parallel operation. In fact, the only communication operation is a global reduction that takes place when summing up the results in Step 2.

Algorithm 1. Integration of the Hamilton matrix with given grid batches B_v

```
(1) For each batch of points $B_v$:
      (a) Find the non-zero basis functions in the batch. Denote the number of these by $\mathrm{nnz}(B_v)$.
      (b) For each $\mathbf{r} \in B_v$ and each non-zero basis function $\varphi_i,\ i = 1, \ldots, \mathrm{nnz}(B_v)$ evaluate the matrices $K_{i,\mathbf{r}} = w(\mathbf{r})\varphi_i(\mathbf{r})$
          and $L_{j,\mathbf{r}} = (\hat{h}\varphi_j)(\mathbf{r})$.
      (c) Compute the part of the submatrix of the Hamilton matrix that corresponds to $B_v,\ h_{ij}[B_v]$, with a matrix–
          matrix product over the points in $B_v$, i.e., $h_{ij}[B_v] = \sum_{\mathbf{r}} K_{i,\mathbf{r}} L_{j,\mathbf{r}}$.
(2) After the loop over $B_v$'s is finished sum up the results: $h_{ij} = \sum_v h_{ij}[B_v]$.
```

4. Grid partitions

4.1. General form of the problem

Let us start with the formal definition of the general problem of finding the grid batches that are as localized as possible for the grid based operations as given in Problem 1:

Problem 1. Given a (finite and non-empty) set of points $P \subset \mathbb{R}^3$, find the batches $\{B_v\},\ v = 1, \ldots, N_{batches}$ such that

(1) $\cup_v B_v = P$.
(2) $B_v \cap B_\mu = \emptyset$ for $v \neq \mu$.
(3) $c_0 \leqslant \#B_v \leqslant c_1$ for some $c_\tau, \#P \geqslant c_\tau \geqslant 1$. Here $\#S$ denotes the number of points in the set S.
(4) The quantity
  (a) $\mathrm{avg}_v\{\mathrm{nnz}(B_v)\}$ or
  (b) $\max_v\{\mathrm{nnz}(B_v)\}$
  is minimized.

The target 4a aims for optimal performance whereas the target 4b is geared towards minimal workspace memory consumption.

It should be noted that the general form of Problem 1 is not solvable in practice due to high complexity, and therefore it is necessary to employ heuristic methods instead of solving the optimization problem directly. However, in the special case when $c_0 = 1$, the optimal solution can be found and it is given by batches containing only single points. We use this case as a reference for our algorithms and denote $B_v^{ref} = \{\mathbf{r} | \mathbf{r} = \mathbf{r}(v)\}$, the set containing the vth integration point.

We note that the problem setting is closely related to partitioning methods needed in adaptive multilevel methods for solving parallel differential equations (see, e.g., Ref. [33] and references therein) except that in our case the grid is not evolving and the global localization of the partitions is not an issue. Another relation can be found to the integration methods discussed in Ref. [34]. However, our aim is not to partition the space into regions where different cubatures can be applied but focus on the localization of the integration batches for a given grid and cubature. Finally, the problem of finding good grid partitions is a generic one for many codes similar to ours. Nonetheless, while the *use* of grid partitions is mentioned in several works (e.g., Refs. [11,24,25,35], in the slightly different context of load balancing in Ref. [36]), to our knowledge only one work (Ref. [11]) discusses the choice of their *shape* to minimize the computational work in a comparative way (in that case, a hybrid of the radial shell and octree methods below).

4.2. Three top-down methods to partition the grid

In this section we present three top-down methods to partition the atom centered real-space grids. The methods are top-down in the sense that they generate the batches without using information on the structure of the grid within the generated batch. The resulting algorithms are very efficient and the batching can be done with minimal cost. On the other hand, the local distribution of the points is not fully accounted for and, e.g., local variations in the density of points are not considered. We will return to this question in Section 5.3 for our discussion about alternative bottom-up methods for generating the grid batches.

4.2.1. Radial shells and their partitions

The first method is based on the geometric properties of the overlapping grids and is the most straightforward of the methods we consider. In this case, the batches are simply taken to be the Lebedev grids making up the radial shells. Furthermore, they can be refined by considering halves, quadrants and octants of the shells, leading to

$$B_v = \{\text{angular grid/subset of the grid for one radial shell}\}$$

for $v = 1, \dots, N_{batches}$.

4.2.2. Octree partitions of the grid

The second method does away with the relation between the grids and the atoms and considers the grid only as a set of points in $\mathbb{R}^3$. Then the grid is recursively partitioned into eight sub-grids by splitting the set of points with planes parallel to the coordinate axes. The details are given in Algorithm 2 that uses a depth-first method to build the octree defining the grid batches B_v. In Step 4 several conditions can of course be employed to accept S_μ as a new batch. We have used two conditions that must be satisfied simultaneously: (1) the value representing a weighted size of the batch,

$$\#S_\mu \times \frac{\max_{\mathbf{r}\in S_\mu}\{\text{nnz}(\{\mathbf{r}\})\}}{\max_{\mathbf{r}\in P}\{\text{nnz}(\{\mathbf{r}\})\}},$$

must be less than a given bound, C_S, and (2) the absolute size of the batch, $\#S_\mu$, must be less than a given (different) bound C_H. We note that the octree method has a close relation to spatial division methods used for a long time in computational geometry, e.g for constructing a mesh for finite-element calculations [37,38].

Algorithm 2. Octree method for grid partitioning

```
(1) Initialize the active set $S$ to contain all the points $S = P$
(2) Compute the center of mass for $S$
(3) Split $S$ into eight subsets $S_1, \dots, S_8$ using cut-planes parallel to coordinate axes and going through the center of mass.
(4) For each subset $S_\mu$ check if $S_\mu$ is acceptable:
      (a) If yes, make $S_\mu$ into a batch $B_v = S_\mu$.
      (b) If no, go to 2 with $S_\mu$ as the new active set $S$.
```

4.2.3. Grid adapted cut-plane method

The obvious drawback (from the algorithmic point of view) of the octree method is that the coordinate axes are given a special role as the planes determining the three planes to cut the set S. Also tying the local origin to the center of mass of S does not necessarily result in even-sized subsets S_μ. Both shortcomings are relatively easy to overcome by (1) using only a single plane to cut S but adapting the orientation of the plane to S and (2) adjusting the location of the plane so that the resulting partitions are even-sized. The details are given in Algorithm 3. In Step 6 we use the same criteria as in the case of the octree method. We note that the adapted cut-plane method is a variation of a method presented in the lecture notes by Kahan [39] and closely related to "Principal Direction Divisive Partitioning" algorithm used in data mining [40].

Algorithm 3. Grid adapted cut-plane method for grid partitioning

```
(1) Initialize the active set $S$ to contain all the points $S = P$
(2) Compute the center of mass for $S$
(3) Find the direction of the cut-plane by computing the normal $\mathbf{n}$ of a plane $\Pi$ through the center of mass such that
    $\sum_{\mathbf{r}\in S}\|\mathbf{r} - \Pi\|^2$ is maximized.
(4) Compute the position of the cut-plane that divides $S$ into two even-sized sets. This is done by sorting the set of real
    numbers $\{\mathbf{r}\cdot\mathbf{n}\}_{\mathbf{r}\in S}$ and dividing the sorted set.
(5) Split $S$ into two subsets $S_1, S_2$ using the cut-plane.
(6) For both subsets $S_\mu, \mu = 1, 2$, check if $S_\mu$ is acceptable:
      (a) If yes, make $S_\mu$ into a batch $B_v = S_\mu$.
      (b) If no, go to 2 with $S_\mu$ as the new active set $S$.
```

The complexity of Algorithm 3 is easy to compute. At each level k of the partition tree the cost is linear in the number of points in the grid, $O(N)$, so that the total cost is $O(k_{max}N)$. Since the algorithm keeps the binary tree balanced we have that $k_{max} \sim \log_2 N$ and the complexity of the full algorithm is $O(N\log_2 N)$. Similar reasoning gives that the octree method, Algorithm

2, has a complexity $O(N\log_8 N)$ for optimal inputs. However, since our grid is not uniform and thus the octree is not balanced the actual runtimes are slightly worse. In practice, grid partitioning accounts only for a negligible part of the cost of the EST calculation.

5. Results

5.1. Top-down methods: timings and batch size

To test the different grid partitioning methods, we consider four different physical systems: fully extended polyalanine chains, cluster cutouts of fcc Cu, periodic Au(100)-"hex" surfaces [41–44], and bulk fcc Al. The basis sizes and grid parameters used in our calculations are shown in Table 1 and they correspond to the values given in Ref. [8].

We show the changes in timings for one self-consistency cycle, the weighted average number of non-zero basis functions over batches, $\langle \mathrm{nnz} \rangle$:

$$\langle \mathrm{nnz} \rangle = \frac{\sum_v \# B_v \cdot \mathrm{nnz}(B_v)}{\sum_v \# B_v}$$

as well as the maximal number of non-zero basis functions over all the batches: $\mathrm{nnz}_{max} = \max_v\{\mathrm{nnz}(B_v)\}$. In addition, we show the average size of the batches, B_v and their standard deviation, σ. With the octree and adapted methods, we have used the values $C_S = 200$ for the desired weighted size of a batch, and $C_H = 400$ for the maximal size of a batch as the parameters controlling the termination of the algorithm.

The results for the polyalanine chains are displayed in Fig. 3. It is clearly visible that the adapted cut-plane method yields a partitioning that (1) gives the smallest average and maximal number of non-zero basis functions, (2) allows for the largest average batch size and consequently (3) gives the best performance. The actual savings for a single self-consistency cycle for the largest molecule are 14% and 24% when compared to the radial shells and octree methods, respectively.

The results for the octree method are notably bad for the longest polyalanine chains. This is a result of the "long and thin" nature of the molecules. Since the octree method recursively splits each batch into eight sub-batches the shape of the original molecule is inherited by the batches. Consequently, some batches extend over a large portion of the molecule and lead to a growing number of non-zero basis functions in the batch. This phenomenon is illustrated in Fig. 4 where the worst batch

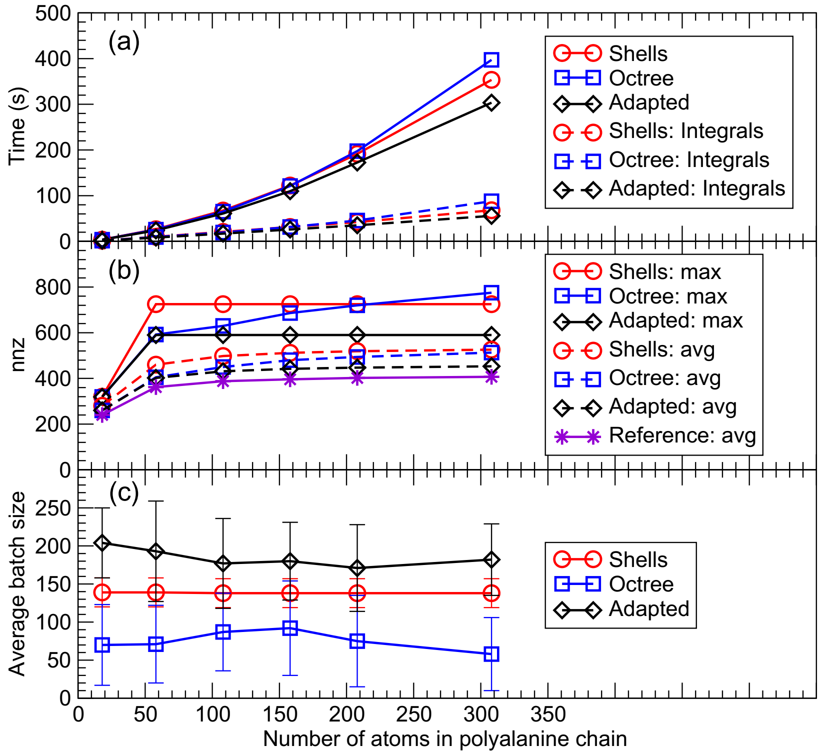

Fig. 3. Results of different grid batching algorithms for fully extended polyalanine molecules: (a) timings in seconds for one self-consistency cycle of different grid partitioning methods, full lines = entire cycle, dashed lines = integration of the Hamilton matrix, (b) non-zero basis functions per batch, nnz, and (c) average size of the batches and their standard deviation.

in this respect is shown in red and is clearly far from optimal shape. The problem could be solved by running a post-analysis on the batches produced by Algorithm 2 but since the adapted cut-plane method solves the problem in a natural and efficient way, there is no point in pursuing the matter further.

For cluster cutouts of fcc Cu, we obtain results with a similar trend except that in this case the radial shells method is clearly the worst choice whereas the octree method is closer to the adapted method as shown in Fig. 5. Actual savings are 33% and 11% for the largest cluster. The difference in favor of the adapted method over the octree method is explained in the two lower panels of Fig. 5. In the middle panel it can be seen that the maximal and average number of non-zero basis functions is almost equal for both methods but the adapted method obtains this result with considerably larger batches as shown in the lowest panel of Fig. 5. It follows that the adapted method is able to perform the matrix multiplication in Step 1c of Algorithm 1 with a larger matrix size, giving better overall performance. In addition, when working with packed matrix storage for the global Hamiltonian h_{ij}, fewer batches mean that less time is consumed for sorting the locally non-zero matrices $h_{ij}[B_v]$ into the global structure (which must be done once per batch). It should also be noted from Fig. 5 that the number of non-zero basis functions has not yet saturated, contrary to the case of the fully extended polyalanines. Consequently, the method has not yet reached the (near) linear scaling regime for these systems.

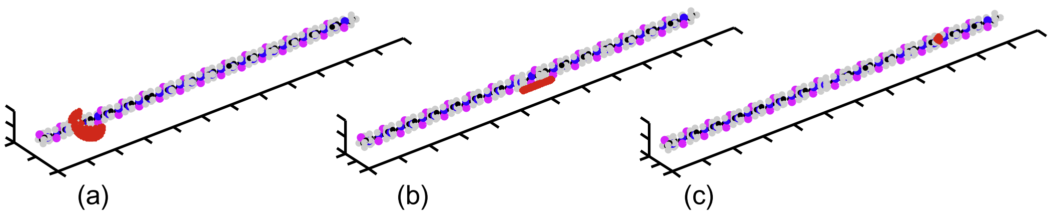

Fig. 4. Polyalanine chain of 30 residues (308 atoms) and the worst batch produced by each of the batching methods (red): (a) shells, (b) octree, and (c) adapted.

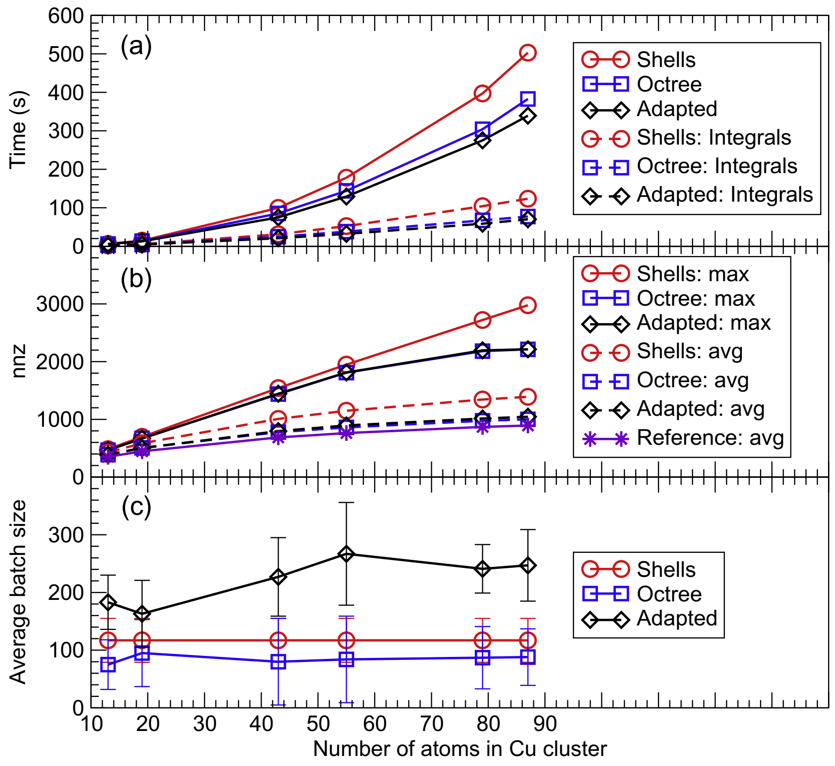

Fig. 5. Results of different grid batching algorithms for the cluster fcc Cu cutouts: (a) timings in seconds for one self-consistency iteration of different grid partitioning methods, full lines = entire cycle, dashed lines = integration of the Hamilton matrix, (b) maximal and average number of non-zero basis functions per batch, nnz, and (c) average size of the integration batches and their standard deviation (bottom).

Our third example is a periodic system: a $(5 \times m)$ approximant surface slab to the Au(100)-"hex" surface reconstruction [41–44] with three layers and an increasing number m of lateral rows. In periodic systems, we need to map the grid from the periodic images to the "zeroth" supercell. This effectively breaks the original shell structure and leads to a complete failure of the radial shells method. This is clearly visible in the top panel of Fig. 6 where the integration alone with the radial shells method takes more time as the entire cycle using the octree or the adapted methods. The adapted method is again the winner here saving 72% and 10% in calculation time for the largest surface when compared to radial shells and octree methods, respectively.

The failure mode of the radial shells method becomes evident when we inspect the two lower panels of Fig. 6. The extremely high number of the non-zero basis functions clearly shows that the radial shell based partitioning is not the correct approach for periodic systems. On the other hand, the two other methods show a clear saturation of the number of non-zero basis functions, indicating good localization of the batches already from the start. Since the integration grid is distributed rather uniformly over the supercell, the differences between octree and adapted methods are relatively small. However, again the adapted method is able to produce the same localization effect using considerably larger batches.

The final example is a system that is periodic in all three dimensions: bulk fcc Al with an increasing number of atoms in the supercell. The results in Fig. 7 are comparable to the Au(100) surface slab. Again the radial shells method is a failure and the adapted method performs best. In this case the savings in the calculation time for the largest supercell are 43% and 8% when compared to radial shells and octree methods, respectively.

For this example, we comment additionally on two more performance related questions, first the memory requirements for the integrations and the density update, and second the setup time of the (nominally) $O(N^2)$ update of the density matrix, D_{ij}. For a three-dimensional bulk system like fcc Al, both points should be particularly critical because the density of active basis functions (overlapping from other unit cells) in the integration volume is highest compared to other structure types with more vacuum (surfaces, molecules,...). On the memory side, one limiting array is the temporary storage of the factors $K_{i,\mathbf{r}} = w(\mathbf{r})\varphi_i(\mathbf{r})$ and $L_{j,\mathbf{r}} = (\hat{h}\varphi_j)(\mathbf{r})$ (see Algorithm 1) required to store the basis functions and the Hamiltonian applied to them on each grid point of the current batch. For the fcc Al unit cells of Fig. 7, the maximum storage size is manageable and approximately constant as a function of unit cell size, and does not exceed 350,000 entries even for 500 atoms when using the adapted method. Likewise, the construction of the density matrix D_{ij} for the 500 atom cell remains rather manageable, taking approx. 2 s out of a total of 60 s used for the density update as given in Eq. (6).

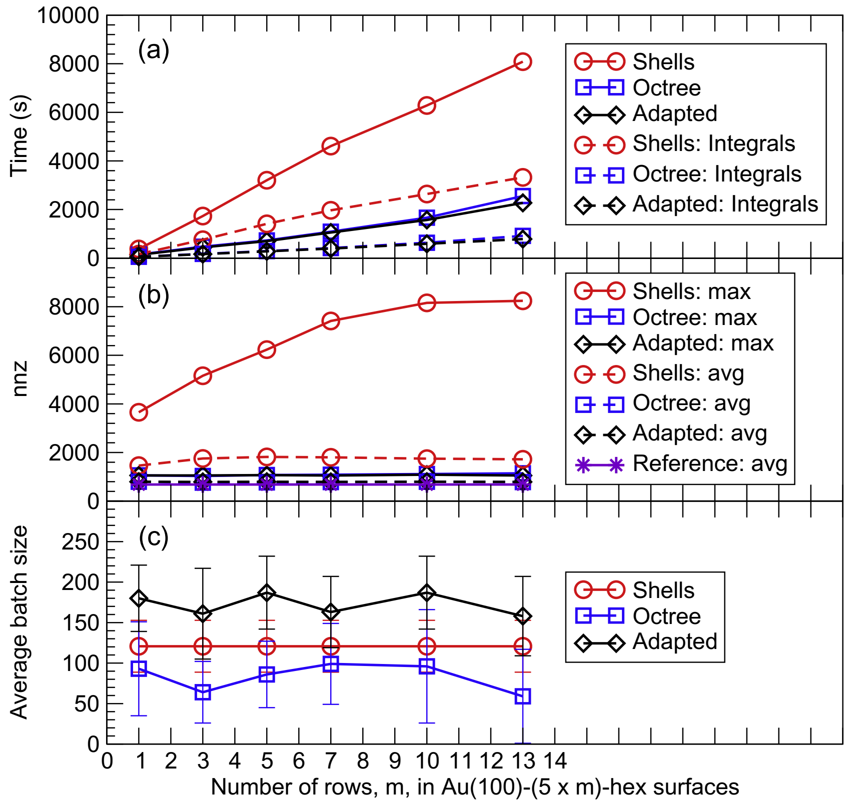

Fig. 6. Results of different grid batching algorithms for the Au(100)-$(5 \times m)$-hex surfaces: (a) timings in seconds for one self-consistency iteration of different grid partitioning methods, full lines = entire cycle, dashed lines = integration of the Hamilton matrix, (b) maximal and average number of non-zero basis functions per batch, nnz, and (c) average size of the batches and their standard deviation.

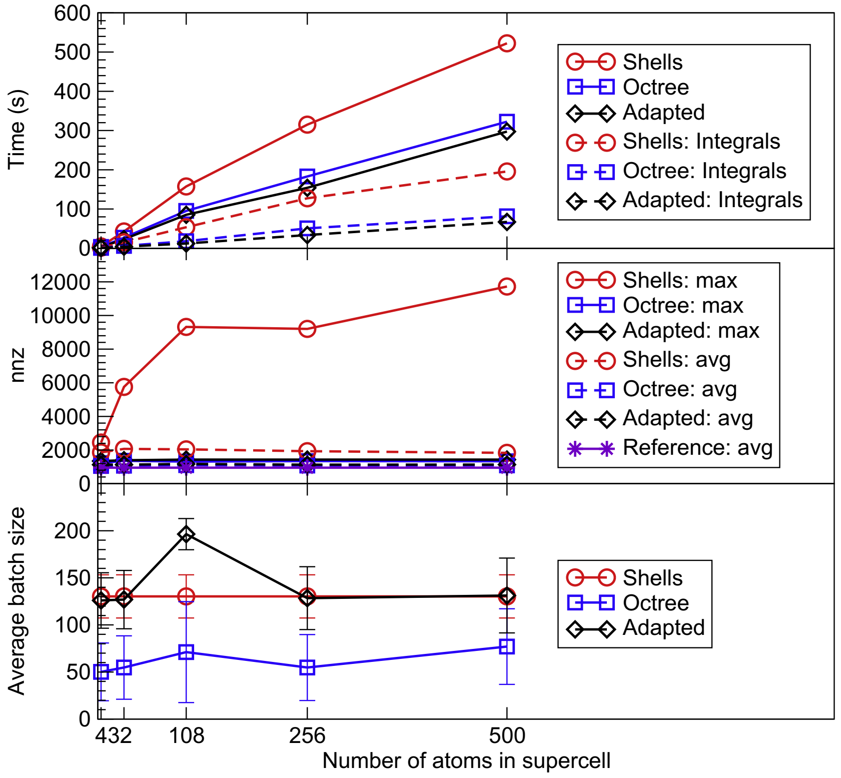

Fig. 7. Results of different grid batching algorithms for the Al-fcc bulk: (a) timings in seconds for one self-consistency cycle of different grid partitioning methods, full lines = entire cycle, dashed lines = integration of the Hamilton matrix, (b) non-zero basis functions per batch, nnz, and (c) average size of the batches and their standard deviation.

The general trend in all cases is thus similar: The adapted cut-plane method performs best in all respects and the octree method and the radial shells with subdivisions are worse. A remarkable failure of the method based on radial shells can be seen in the case of the periodic systems, where the maximal number of non-zero basis functions skyrockets. This is due to the fact that, in periodic systems, the grid operations are all performed on the supercell and thus the 'natural' ordering of the radial shells is destroyed.

5.2. On the optimality of the grid partitions

Due to the complexity of the general problem of creating the grid batches we cannot determine accurately how close our heuristic methods are to the optimal solution. However, we can get some idea on the quality of our results by comparing to the case where the batches are taken to be only single points, i.e., when $c_0 = c_1 = 1$. In this case we can compare the actual number of the evaluations of the basis functions to the number of evaluations required in the case of single-point batches. To this end, recall that $B_v^{ref} = \{\mathbf{r}(\mu)\}$ and define the effectivity of the batches, e, as

$$e = \frac{\sum_v \mathrm{nnz}(B_v) \times \#B_v}{\sum_v \mathrm{nnz}(B_v^{ref})}. \tag{7}$$

The effectivities for all three different batching schemes for all four test cases are reported in Tables 2–5. As expected from the results above, the adapted method provides best overall effectivity and the results for the radial shells method are worst by a large factor. When comparing the values for e it should be taken into account that the average batch size is largest for the adapted method and consequently effectivity close to one is even harder to achieve.

The last two rows of Table 4 show the improvement in e in the case of the reconstructed Au-surface when the parameters controlling the batch sizes are halved and divided by four, i.e., when $C_S = 100$, $C_H = 200$ and $C_S = 50$, $C_H = 100$, respectively. This aids in improving the effectivity e even further, albeit at the price of a larger overall number of batches (increasing the overhead from per-batch operations such as sorting matrix elements into the global h_{ij}), and reducing the size of the performed matrix products.

Table 2
Effectivity *e* of different batching schemes for the fully extended polyalanines.

Number of atoms	18	58	108	158	208	308
Radial shells	1.17	1.27	1.29	1.29	1.29	1.29
Octree	1.08	1.12	1.16	1.21	1.23	1.26
Adapted	1.08	1.11	1.11	1.11	1.11	1.12

Table 3
Effectivity *e* of different batching schemes for the fcc Cu-clusters.

Number of atoms	13	19	43	55	79	87
Radial shells	1.22	1.31	1.47	1.51	1.55	1.56
Octree	1.10	1.13	1.13	1.13	1.12	1.12
Adapted	1.11	1.12	1.16	1.17	1.17	1.17

Table 4
Effectivity *e* of different batching schemes for the Au(1 0 0)-($m \times 5$)-surfaces.

Number of Au rows	1	3	5	7	10	13
Radial shells	2.14	2.57	2.67	2.64	2.56	2.52
Octree	1.17	1.11	1.13	1.15	1.17	1.15
Adapted ($C_S = 200,\ C_H = 400$)	1.17	1.16	1.17	1.16	1.17	1.16
Adapted ($C_S = 100,\ C_H = 200$)	1.12	1.12	1.13	1.12	1.13	1.12
Adapted ($C_S = 50,\ C_H = 100$)	1.09	1.09	1.09	1.09	1.09	1.09

Table 5
Effectivity *e* of different batching schemes for the fcc Al bulk.

Number of Al atoms in unit cell	4	32	108	256	500
Radial shells	1.96	2.18	2.15	2.03	1.93
Octree	1.11	1.13	1.16	1.13	1.15
Adapted	1.19	1.19	1.24	1.19	1.19

5.3. How about bottom-up methods?

All the methods presented above are so called top-down methods, i.e., they start from a given set of points and recursively divide it into smaller chunks until a desired batch size is reached. The inherent drawback of top-down methods is that the local features of the distribution of the grid points in three-dimensional space are not accounted for in detail. This observation gives rise to another set of approaches, so called bottom-up methods, where the local environment of each of the grid points is analyzed before creating the grid batches.

We have implemented and tested two bottom-up methods: First, a method where a Delaunay mesh with the grid points as nodes is created and the mesh is partitioned with a multilevel graph partitioning method. This is all realized using external established tools [45,46]. Second, a method where the batches are built by grouping nearby points together and then recursively merging the groups until a desired batch size is reached. We denote the number of merged items per level a grouping factor, *gf*.

The bottom-up methods are able to produce a set of batches that is similar to the ones produced the grid adapted method as can be seen from Tables 6 and 7. However, they require more resources. The first method, a combined Delaunay mesh and graph partitioning approach, uses a large amount of memory to store the mesh. The second method, the grouping algorithm,

Table 6
Performance of the bottom-up methods for a single polyalanine residue (18 atoms).

	Graph part.	Groups, $gf = 2$	Groups, $gf = 4$	Groups, $gf = 8$
$\langle$nnz$\rangle$	257.97	265.37	266.01	273.36
Effectivity	1.08	1.11	1.11	1.14
Average batch size	200.00	255.90	255.56	508.81
Std. dev. of batch size	4.93	3.37	8.65	36.25

Table 7
Performance of the bottom-up methods for a Au(1 0 0)-(1 × 5) surface.

	Graph part.	Groups, $gf = 2$	Groups, $gf = 4$	Groups, $gf = 8$
⟨nnz⟩	796.73	834.79	845.39	902.90
Effectivity	1.17	1.22	1.24	1.32
Average batch size	199.98	255.99	255.89	509.54
Std. dev. of batch size	4.65	0.61	3.91	32.13

needs a lot of searches to find the nearest neighbors of the groups at each level. These searches take a lot of computation time to complete. Even for small test systems the grid partitioning using the grouping algorithm becomes by far the most time consuming part of the electronic structure calculation, rendering the approach useless in practice.

On the other hand, the actual graph partitioning method is fast and it can accept also other graphs than Delaunay meshes as input. The bottom-up methods can thus be developed further by building a graph by connecting nearby points and then splitting the graph. In this case, it is important to include the local distribution of the grid points by using a graph whose nodes can have a varying index.

6. Conclusions

The results and theoretical considerations above show how grid partitioning combined with localization of the basis functions leads to linearly scaling grid-based operations, i.e., the integration and the electron density update, in EST calculations using NAOs as the basis set. The effect of the grid partitions is most pronounced for periodic systems, but also the performance for non-periodic cases is notably improved when the grid is properly divided into batches.

The fact that localization entails the performance of NAOs is not a surprise. However, the complexity of the actual problem of finding an optimal grid partitions is too high to be tackled in full, and heuristic methods must be employed. It is somewhat more surprising that the methods implemented and tested here exhibit such a big difference in their performance. The best method we have obtained, the adapted cut-plane method, is rather close to the theoretical optimum for our test systems, indicating a good level of heuristic approach. The octree method suffers from the tendency to generate batches with very few points leading to inefficiency and has the drawback of unnecessarily replicating the geometry of the system. Finally, periodic systems present a more complicated environment due to the complex mapping of the periodic images of the grid-points. This problem manifests itself most strikingly in the failure of the radial shell method.

In this work we have focused mainly on the top-down methods. The other approach, bottom-up methods, suffers from the fact that it is problematic to generate a graph over the grid that accurately describes the local environment of the grid points. In addition, however, the much simpler top-down adaptive grid method used above performs at nearly the same or better effectivity than the formally more rigorous bottom-up methods attempted here. Apparently, this fast, well-tuned top-down approach captures all practically needed aspects of the grid partitioning problem, leaving no incentive to pursue any more complicated schemes.

Acknowledgment

Ville Havu acknowledges the Alexander von Humboldt (AvH) foundation for funding, and the use of the "Louhi" supercomputer of the Finnish IT center for sciences, CSC.

References

[1] P. Hohenberg, W. Kohn, Inhomogeneous electron gas, Phys. Rev. 136 (1964) B864.
[2] W. Kohn, L. Sham, Self-consistent equations including exchange and correlation effects, Phys. Rev. 140 (1965) A1133.
[3] F. Averill, D. Ellis, An efficient numerical multicenter basis set for molecular orbital calculations: application to $FeCl_4$, J. Chem. Phys. 59 (1973) 6412.
[4] B. Delley, D. Ellis, Efficient and accurate expansion methods for molecules in local density models, J. Chem. Phys. 76 (1982) 1949.
[5] B. Delley, An all-electron numerical method for solving the local density functional for polyatomic molecules, J. Chem. Phys. 92 (1990) 508.
[6] K. Koepernik, H. Eschrig, Full-potential nonorthogonal local-orbital minimum-basis band-structure scheme, Phys. Rev. B 59 (1999) 1743.
[7] A. Horsfield, Efficient ab initio tight binding, Phys. Rev. B 56 (1997) 6594.
[8] V. Blum, R. Gehrke, F. Hanke, P. Havu, V. Havu, X. Ren, K. Reuter, M. Scheffler, Ab initio molecular simulations with numeric atom-centered orbitals, Comput. Phys. Commun. (2009). <http://dx.doi.org/10.1016/j.cpc.2009.06.022>.
[9] O. Treutler, R. Ahlrichs, Efficient molecular numerical integration schemes, J. Chem. Phys. 102 (1995) 346.
[10] C. Fonseca Guerra, J. Snijders, G. te Velde, E. Baerends, Towards an order-*n* dft method, Theor. Chem. Acc. 99 (1998) 391–403.
[11] R. Stratmann, G. Scuseria, M. Frisch, Achieving linear scaling in exchange-correlation density functional quadratures, Chem. Phys. Lett. 257 (1996) 213–223.
[12] J.M. Soler, E. Artacho, J.D. Gale, A. Garcia, J. Junquera, P. Ordejon, D. Sanchez-Portal, The siesta method for ab initio order-*n* materials simulation, J. Phys.: Condens. Matter 14 (2002) 2745–2779.
[13] A. Mostofi, C.-K. Skylaris, P. Haynes, M. Payne, Total-energy calculations on a real space grid with localized functions and a plane-wave basis, Comput. Phys. Commun. 147 (2002) 788–802.
[14] T. Ozaki, H. Kino, Efficient projector expansion for the ab initio LCAO method, Phys. Rev. B 72 (2005) 045121.
[15] D. Bowler, T. Miyazaki, M. Gillan, Recent progress in linear scaling *ab initio* electronic structure techniques, J. Phys.: Condens. Matter 14 (2002) 2781.

[16] C.-K. Skylaris, P.D. Haynes, A.A. Mostofi, M.C. Payne, Introducing ONETEP: Linear-scaling density functional simulations on parallel computers, J. Chem. Phys. 122 (2005) 084119.
[17] H. Eschrig, I. Bergert, An optimized LCAO version for band structure calculations application to copper, Phys. Stat. Sol. (b) 90 (1978) 621.
[18] H. Eschrig, Optimized LCAO Method and The Electronic Structure of Extended Systems, Akademie Verlag and Springer, Berlin, 1988.
[19] O. Sankey, D. Niklewski, Ab initio multicenter tight-binding model for molecular-dynamics simulations and other applications in covalent systems, Phys. Rev. B 40 (1989) 3979.
[20] D. Porezag, T. Frauenheim, T. Köhler, G. Seifert, R. Kaschner, Construction of tight-binding-like potentials on the basis of density-functional theory: application to carbon, Phys. Rev. B 51 (1995) 12947.
[21] S. Kenny, A. Horsfield, H. Fujitani, Transferable atomic-type orbital basis sets for solids, Phys. Rev. B 62 (2000) 4899.
[22] J. Junquera, O. Paz, D. Sanchez-Portal, E. Artacho, Numerical atomic orbitals for linear-scaling calculations, Phys. Rev. B 64 (2001) 235111.
[23] T. Ozaki, Variationally optimized atomic orbitals for large-scale electronic structures, Phys. Rev. B 67 (2003) 155108.
[24] Y. Li, M. Wrinn, J. Newsam, M. Sears, Parallel implementation of a mesh-based density functional electronic structure code, J. Comput. Chem. 16 (1995) 226–234.
[25] J. Baker, M. Shirel, Ab initio quantum chemistry on pc-based parallel supercomputers, Parallel Comput. 26 (2000) 1011–1024.
[26] S. Goedecker, Linear scaling electronic structure methods, Rev. Mod. Phys. 71 (1999) 1085–1123.
[27] A.H. Stroud, Approximate Calculation of Multiple Integrals, Prentice-Hall, Englewood Cliffs, NJ, 1971.
[28] B. Delley, High order integration schemes on the unit sphere, J. Comput. Chem. 17 (1995) 1152.
[29] V. Lebedev, D. Laikov, A quadrature formula for the sphere of the 131st algebraic order of accuracy, Doklady Math. 59 (1999) 477–481.
[30] J. Baker, J. Andzelm, A. Scheiner, B. Delley, The effect of grid quality and weight derivatives in density functional calculations, J. Chem. Phys. 101 (1994) 8894.
[31] A. Becke, A multicenter numerical integration scheme for polyatomic molecules, J. Chem. Phys. 88 (1988) 2547–2553.
[32] J.M. Perez-Jorda, W. Yang, An algorithm for 3d numerical integration that scales linearly with the size of the molecule, Chem. Phys. Lett. 241 (1995) 469–476.
[33] W.F. Mitchell, A refinement-tree based partitioning method for dynamic load balancing with adaptively refined grids, J. Parallel Distributed Comput. 67 (2007) 417–429.
[34] G. te Velde, E. Baerends, Numerical integration for polyatomic structures, J. Comput. Phys. 99 (1992) 84–98.
[35] M. Watson, P. Salek, P. Macak, T. Helgaker, Linear-scaling formation of Kohn-Sham Hamiltonian: Application to the calculation of excitation energies and polarizabilities of large molecular systems, J. Chem. Phys. 121 (2004) 2915.
[36] C. Gan, M. Challacombe, Linear scaling computation of the Fock matrix. VI. Data parallel computation of the exchange-correlation matrix, J. Chem. Phys. 118 (2003) 9128.
[37] M.S. Shephard, M.K. Georges, Automatic three-dimensional mesh generation by the finite octree technique, Int. J. Numer. Methods Eng. 32 (1991) 709–749.
[38] S.A. Vavasis, The QMG package, URL: <http://www.cs.cornell.edu/home/vavasis/qmg-home.html>.
[39] W. Kahan, Separating Clouds by a Plane, Lecture Notes, CS Division, UC Berkeley. URL: <http://www.cs.berkeley.edu/~wkahan/MathH110/Separate.pdf>.
[40] D. Boley, Principal direction divisive partitioning, Data Mining Knowl. Discovery 2 (1998) 325–344.
[41] D. Fedak, N. Gjøstein, On the anomalous surface structures of gold, Surf. Sci. 8 (1967) 77–97.
[42] M. van Hove, R. Koestner, P. Stair, J. Biberian, L. Kesmodel, I. Bartos, G. Somorjai, The surface reconstructions of the (100) crystal faces of iridium, platinum and gold: I. Experimental observations and possible structural models, Surf. Sci. 103 (1981) 189.
[43] G. Binnig, H. Rohrer, C. Gerber, E. Stoll, Real-space observation of the reconstruction of Au(100), Surf. Sci. 144 (1984) 321.
[44] D. Gibbs, B. Ocko, D. Zehner, S. Mochrie, Structure and phases of the Au(001) surface: In-plane structure, Phys. Rev. B 42 (1990) 7330.
[45] C. Barber, D. Dobkin, H. Huhdanpaa, The quickhull algorithm for convex hulls, ACM Trans. Math. Software 22 (1996) 469–483. <http://www.qhull.org>.
[46] G. Karypis, V. Kumar, METIS A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices, Version 4.0. URL: <http://glaros.dtc.umn.edu/gkhome/views/metis>.