JCTC Journal of Chemical Theory and Computation

Article

pubs.acs.org/JCTC

Construction of the Fock Matrix on a Grid-Based Molecular Orbital Basis Using GPGPUs

Sergio A. Losilla,[†,∥] Mark A. Watson,[‡] Alán Aspuru-Guzik,[§] and Dage Sundholm*[,†]

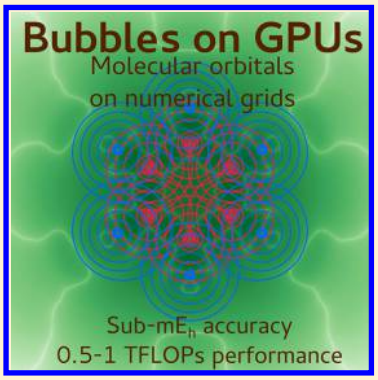
[†]Department of Chemistry, University of Helsinki, P.O. Box 55, FIN-00014 Helsinki, Finland

[‡]Schrödinger, Inc, 120 West 45th Street, New York, New York 10036, United States

[§]Department of Chemistry and Chemical Biology, Harvard University, Cambridge, Massachusetts 02138, United States

Supporting Information

ABSTRACT: We present a GPGPU implementation of the construction of the Fock matrix in the molecular orbital basis using the fully numerical, grid-based *bubbles* representation. For a test set of molecules containing up to 90 electrons, the total Hartree−Fock energies obtained from reference GTO-based calculations are reproduced within 10^{-4} E_h to 10^{-8} E_h for most of the molecules studied. Despite the very large number of arithmetic operations involved, the high performance obtained made the calculations possible on a single Nvidia Tesla K40 GPGPU card.



1. INTRODUCTION

Nowadays, most molecular electronic structure calculations where all electrons are explicitly treated are performed with the molecular orbitals (MOs) represented in terms of atomic orbitals. Atomic orbitals are atom-centered basis functions that can be expressed in spherical coordinates as

$$\chi_i(\mathbf{r}) = R_{A_i l_i m_i}(r_{A_i})\, Y_{l_i m_i}(\theta_{A_i}, \varphi_{A_i}) \tag{1}$$

where r_A, θ_A, and φ_A are the spherical coordinates relative to the position of the Ath nucleus, $\mathbf{R}_A$. The angular functions $Y_{lm}(\theta,\varphi)$ are spherical harmonics. The maximum angular momentum number l included in a given expansion is denoted by L. Different families of radial basis functions $R_{Alm}(r_A)$ can be found in the literature.[1−4]

The main reason for the success of atomic-orbital based approaches is the compactness of the resulting basis set. For example, a valid, although rough, model for the ground state of H_2 consists of just two s ($L = 0$) Slater-type orbitals. For Hartree−Fock and density functional theory methods, the error in the energy with respect to the complete basis set decreases like e^{-L}, whereas a much slower decay like $(L + 1)^{-3}$ is observed for correlated wave function methods.[5−7] Enlarging the basis set with a larger L and more radial functions provides more accurate results. However, even though increasing L yields systematically smaller contributions, the basis-set convergence is neither smooth nor fast. Adding radial functions is not straightforward because, in practice, it requires optimizations of the basis set that might bias the outcome of some calculations, including Rydberg states or distorted molecules in strong laser fields. Despite the existence of procedures to reduce this problem,[8] an atomic-orbital basis set cannot be arbitrarily large because of the onset of numerical instabilities caused by linear dependencies.

A suitable approach to obtain arbitrarily high accuracy is to use local basis sets that consist of functions with compact support, such as finite-element functions, arranged on a real space grid. The representation of the orbitals can be systematically improved in the regions of space where it is needed without encountering problems with linear dependencies. The most precise electronic structure calculations have been performed using some variety of local basis functions.[9−17] However, the molecular orbitals and the electronic interaction potentials are very steep in the vicinity of the nuclei due to the singularity of the nuclear potentials. Unless special coordinates can be exploited, such as for atoms and diatomic molecules, a huge number of local basis functions must be used to describe these sharp features. For this reason, grid-based approaches have not been possible for arbitrary molecular geometries until recently.

Beyond the high precision that they provide, grid-based approaches have become attractive in the recent years because of their potential for massive parallelization. The type of operations that they involve is well-suited for parallelization on SIMD (single instruction multiple data) architectures such as general-purpose graphics processing units (GPGPU), whereas global basis-set methods usually rely on the diagonalization of very large matrices and integral algorithms that cannot easily be parallelized, especially on GPGPUs. Although remarkable progress has been made in this direction,[18−20] the performance

Received: December 15, 2014
Published: April 15, 2015

DOI: 10.1021/ct501128u
J. Chem. Theory Comput. 2015, 11, 2053−2062

suffers when the angular momentum quantum number (L) of the basis increases. For a basis set including d functions, a speedup of only a factor of 10 was obtained.[21] On the other hand, very efficient implementations of real-space electronic structure methods are becoming increasingly common on GPGPUs.[22−26]

Recently, we proposed a numerical scheme to represent the scalar functions encountered in electronic structure calculations, such as orbitals, electron densities, and potentials.[27] The functions are decomposed into atom-centered parts (the *bubbles*), which contain the steep contributions, and a remainder represented on a Cartesian grid (the *cube*). The bubbles are written as products of angular parts and radial parts. The radial parts are expanded in one-dimensional finite-element functions, and the angular parts are spherical harmonics with small angular momentum number (e.g., up to $L = 2$).

We showed that it is possible to partition molecular charge densities such that the remainder is smooth and can be accurately represented on a three-dimensional grid of tractable size (ca. 10−20 grid points per a_0).[27] We also described algorithms to perform the two crucial operations required for constructing the Fock matrix, namely, computing products of scalar functions and computing the action of linear operators that can be written in a Cartesian-separable form. The missing piece was the Laplacian operator, which is typically an obstacle for grid-based implementations. For that purpose, we have augmented the bubbles representation to support the explicit representation of nuclear singularities in a natural and efficient manner.

In this work, we demonstrate the accuracy and efficiency of the bubbles representation by using it to construct and diagonalize the Fock matrix in the MO basis. The numerical representation of functions is described in Section 2. The necessary algorithms and their implementation are described in Section 3. The implementation and benchmarks on GPGPUs, including test calculations on molecular systems, are discussed in Section 4. We summarize the obtained results, including a future outlook, in Section 5.

2. NUMERICAL REPRESENTATION OF MOLECULAR FUNCTIONS

2.1. The Bubbles Representation. For a system consisting of N_{at} atoms, a function $f(\mathbf{r})$ is partitioned as

$$f(\mathbf{r}) = \sum_{A=1}^{N_{at}} f^A(r_A,\theta_A,\varphi_A) + f^\Delta(\mathbf{r}) \tag{2}$$

where the atom-centered functions $f^A(r_A,\theta_A,\varphi_A)$ are the *bubbles* and the remainder $f^\Delta(\mathbf{r})$ is the *cube*. The bubbles consist of radial and angular parts

$$f^A(r_A,\theta_A,\varphi_A) = \sum_{l=0}^{L}\sum_{m=-l}^{l} f^{Alm}(r_A)\, Y_{lm}(\theta_A,\varphi_A) \tag{3}$$

L is small, typically up to 2, corresponding to functions of s, p, and d symmetry. The angular part of the bubbles are real spherical harmonics, $Y_{lm}(\theta,\varphi)$, which are stored in their Cartesian representation

$$Y_{lm}(\theta,\varphi) = \sum_{uvw} C_{uvw}^{lm}\left(\frac{x}{r}\right)^u\left(\frac{y}{r}\right)^v\left(\frac{z}{r}\right)^w \tag{4}$$

Expressions to compute the C_{uvw}^{lm} coefficients can be found in the literature.[28] For brevity, the spherical harmonics centered at point $\mathbf{R}_A$ are denoted $Y_{lm}^A \equiv Y_{lm}(\theta_A,\varphi_A)$.

The radial functions $f^{Alm}(r_A)$ have the form

$$f^{Alm}(r_A) = \sum_i r_A^{k_f} f_i^{Alm}\, \chi_i^A(r_A) \tag{5}$$

where f_i^{Alm} are the expansion coefficients of the polynomial basis functions $\chi_i^A(r_A)$ (described in detail in the next section) and k_f is an integer, in most cases 0. The cutoff radius of the radial grid $r_{max} = 20a_0$ and the number of cells is $M_A = 200Z_A^{1/4}$, with Z_A being the nuclear charge of the Ath atom. These parameters provide a sufficiently precise representation of the radial functions for the studied molecules. The details on how the nonequidistant radial grid is generated have been discussed elsewhere.[27] The $r_A^{k_f}$ factor permits the explicit representation of some classes of singularities for $k_f < 0$. This is crucial for the evaluation of the Laplacian of functions with cusps at the nuclear position, as will be shown later in Section 3.2.2.

The basis used to represent the cube $f^\Delta(\mathbf{r})$ is a three-dimensional tensor product of one-dimensional polynomial bases

$$f^\Delta(\mathbf{r}) = \sum_{ijk} f_{ijk}^\Delta\, \chi_i^x(x)\, \chi_j^y(y)\, \chi_k^z(z) \tag{6}$$

The one-dimensional bases of the cube are similar to those of the radial functions of the bubbles, constructed as discussed in the next section. For the cube, the grids are chosen to be equidistant, with the grid ranges chosen such that the boundaries are at least $8a_0$ apart from any atomic center.

In the notation for a function designated by f in eqs 2−6, the different pieces are denoted by superscripts (f^A, f^Δ, f^{Alm}), and the expansion coefficients, by subscripts ($f_{ijk}^\Delta, f_i^{Alm}$).

Similar partitions in terms of Gaussian functions and finite elements can be found in the literature.[29−31] This representation has been particularly successful for representing the electrostatic potentials in the calculation of the Coulomb matrix for GTO basis sets. When used to represent orbitals, it is indeed more compact than employing fully numerical radial functions. However, the global nature of the GTO functions imposes similar limitations as pure GTO basis sets: for example, to obtain arbitrary accuracy, a large number of GTO functions with very high angular momentum would be needed. Otherwise, a very large number of finite elements close to the nuclei would be required.

2.2. The One-Dimensional Polynomial Basis. The construction of the one-dimensional polynomial basis is illustrated in Figure 1. The calculation domain $[x_{min},x_{max}]$ is subdivided into M intervals, or *cells*. The boundaries between cells are denoted by s_i, with $s_1 = x_{min}$ and $s_{M+1} = x_{max}$. Each cell is further subdivided into P subintervals of equal length. The length of the subintervals, or *step*, of the jth cell is denoted by h_j. The grid points $\{x_i\}$ are located at the limiting points of these subintervals. The last grid point in one cell is also the first grid point of the next one. The total number of points is $N = MP + 1$.

In each cell, a Pth order Lagrange interpolating polynomial basis is constructed, with l_{ij} denoting the jth function of the ith cell

$$l_{i,j}(x) = \begin{cases} \displaystyle\prod_{\substack{0\le k\le P\\ k\neq j}} \frac{(x-s_i)/h_i - k}{j-k} & \text{if } s_i \le x < s_i + Ph_i \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

DOI: 10.1021/ct501128u
J. Chem. Theory Comput. 2015, 11, 2053−2062

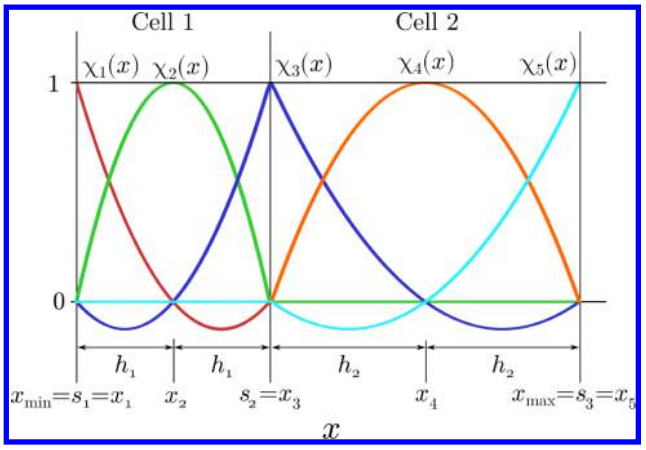


Figure 1. One-dimensional polynomial basis with $M = 2$ and $P = 2$, illustrating the grid points (x_1, x_2, etc.), the cell steps (h_1 and h_2), and the cell boundaries (s_1, s_2, and s_3).

with $1 \le i \le M$ and $0 \le j \le P$. The functions are reorganized using a single index

$$\chi_{(i-1)P+j+1}(x) = l_{ij}(x) \tag{8}$$

The basis functions centered at junction points span two adjacent cells

$$\chi_{iP}(x) = l_{i,P}(x) + l_{i+1,0}(x) \tag{9}$$

For the example in Figure 1, the basis set is

$$\begin{aligned} \{\chi_1(x) &= l_{10}(x), \\ \chi_2(x) &= l_{11}(x), \\ \chi_3(x) &= l_{12}(x) + l_{20}(x), \\ \chi_4(x) &= l_{21}(x), \\ \chi_5(x) &= l_{22}(x)\} \end{aligned} \tag{10}$$

This ensures that expansions in the one-dimensional basis are continuous within the grid range. Note that continuity in the derivatives is not enforced in any way. In practice, this is not a problem provided a sufficiently tight grid is used.

Since $\chi_i(x_j) = \delta_{ij}$, the expansion coefficient for $\chi_i(x)$ is the value of the function at the corresponding grid point x_i, i.e.

$$f_i = f(x_i) \tag{11}$$

Throughout this work, Lagrangian interpolating polynomials of 6th order ($P = 6$) are used. This choice offers good accuracy while keeping the number of points in each cubic cell manageable.

2.3. Memory Considerations. The memory requirements for such a representation are largely dominated by the size of the cube. For a typical grid step of $0.1a_0$, the number of cube grid points per dimension is 100−200, yielding a total number of cube grid point coefficients ($N_{cub} = N_x N_y N_z$) of 10^6 to 10^7, corresponding to 10−100 MB when using 64-bit floating-point numbers. The storage requirements of the bubbles are very small compared to the total memory used, amounting to $N_{bub} = \sum_A (L+1)^2 (M_A P + 1)$ coefficients. For $L = 2$, the number of coefficients per atom ranges between 5000 and 15 000, corresponding to about 100 kB per atomic center.

As the volume of the cube grows approximately linearly with the number of atoms, so does the total number of coefficients

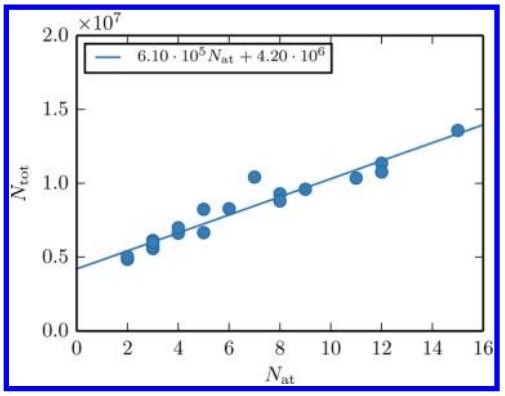


Figure 2. Number of coefficients N_{tot} required to store the electron density as a function of the number of atoms for the molecules discussed in Section 4.3 when equidistant grids with a step of $0.1a_0$ are used.

N_{tot}. This is illustrated in Figure 2 for a grid step of $0.1a_0$, which provides mE_h accuracy for the molecular test systems discussed in Section 4.3. An empty box ($N_{at} = 0$) of $16 \times 16 \times 16 a_0^3$ requires 4.20×10^6 coefficients (ca. 40 MB for 64 bit floats), and each atom costs 6.10×10^5 coefficients more (approximately 5 MB per atom). Halving the grid step increases the number of coefficients approximately by a factor of 8.

3. CONSTRUCTION OF THE FOCK MATRIX

The Hartree−Fock (HF) method is one of the fundamental pillars of electronic structure theory. Despite being considered obsolete for most practical applications, it is the basis of the very accurate correlated wave function methods. Moreover, its formulation is intimately linked with density functional theory, the workhorse of modern quantum chemistry.

In the restricted Hartree−Fock (RHF) method, the electronic wave function of a closed shell molecule is represented as a Slater determinant formed by N_{occ} doubly occupied molecular orbitals $\{\phi_i(\mathbf{r})\}$, which are orthonormal. The total electronic density is given by

$$\rho(\mathbf{r}) = 2 \sum_{i \in occ} |\phi_i(\mathbf{r})|^2 \tag{12}$$

The electronic energy of the system is given by

$$E_{RHF} = \sum_{i \in occ} \int_{\mathbb{R}^3} \phi_i(\mathbf{r}) [2\hat{h} + 2\hat{J} - \hat{K}] \phi_i(\mathbf{r}) d^3r \tag{13}$$

where the one-electron, Coulomb, and exchange operators are given by

$$\hat{h}\phi_i(\mathbf{r}) = \left[-\frac{1}{2}\nabla^2 - \sum_A \frac{Z_A}{|\mathbf{r} - \mathbf{R}_A|} \right] \phi_i(\mathbf{r}) \tag{14}$$

$$\hat{J}\phi_i(\mathbf{r}) = \sum_{k \in occ} \left[\int_{\mathbb{R}^3} \frac{\phi_k(\mathbf{r}')\, \phi_k(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} d^3r' \right] \phi_i(\mathbf{r}) \tag{15}$$

and

$$\hat{K}\phi_i(\mathbf{r}) = \sum_{k \in occ} \left[\int_{\mathbb{R}^3} \frac{\phi_k(\mathbf{r}')\, \phi_i(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} d^3r' \right] \phi_k(\mathbf{r}) \tag{16}$$

The optimal set of molecular orbitals are obtained by minimizing E_{RHF}. A linear combination of molecular orbitals that

DOI: 10.1021/ct501128u
J. Chem. Theory Comput. 2015, 11, 2053−2062

minimizes E_{RHF} is obtained by diagonalizing the matrix of the Fock operator, $\hat{F}$

$$\hat{F}\phi_i(\mathbf{r}) = [\hat{h} + \hat{J} - \hat{K}]\phi_i(\mathbf{r}) \tag{17}$$

Figure 3 shows the simple algorithm used for computing the Fock matrix elements. The two main bottlenecks of the

- *Compute kinetic energy matrix*

for all j molecular orbitals **do**
  $|ket\rangle = -\frac{1}{2}\nabla^2\phi_j$
  for all $i \leq j$ molecular orbitals **do**
    $T_{ij} = \langle \phi_i | ket \rangle$

- *Compute nuclear attraction and Coulomb matrices*

$\rho = 2\sum_{i\in\mathrm{occ}} |\phi_i|^2$
for all j molecular orbitals **do**
  for all $i \leq j$ molecular orbitals **do**
    $V(\mathbf{r}) = \int_{\mathbb{R}^3} \phi_i(\mathbf{r}')\phi_j(\mathbf{r}')|\mathbf{r}-\mathbf{r}'|^{-1}\mathrm{d}^3r'$
    $U_{ij} = -\sum_A Z_A V(\mathbf{R}_A)$
    $J_{ij} = \int_{\mathbb{R}^3} \rho(\mathbf{r})V(\mathbf{r})\mathrm{d}^3r'$

- *Compute exchange matrix*

for all j molecular orbitals **do**
  for all k occupied molecular orbitals **do**
    $|ket\rangle \leftarrow \left[\int_{\mathbb{R}^3} \phi_k(\mathbf{r}')\phi_j(\mathbf{r}')|\mathbf{r}-\mathbf{r}'|^{-1}\mathrm{d}^3r'\right]\phi_k(\mathbf{r})$
  for all $i \leq j$ molecular orbitals **do**
    $K_{ij} = \langle \phi_i | ket \rangle$

$\mathbf{F} = \mathbf{T} + \mathbf{U} + \mathbf{J} - \mathbf{K}$

Figure 3. Algorithm for the numerical construction of the Fock matrix in the MO basis.

operations appearing in Figure 3 are multiplications and applications of the operators in the numerical representation.

In this work, we discuss the numerical construction of the Fock matrix for fixed molecular orbitals taken from a converged basis-set calculation, whereas an iterative solution of the Fock equations in the numerical basis is the ultimate future goal.

Having the Fock matrix at our disposal, it becomes possible to implement a traditional self-consistent iterative scheme to solve the HF equations in the MO basis. However, even assuming that the errors introduced by the numerical grid are negligible, the error with respect to the exact HF energy would be limited by the choice of initial molecular orbitals, in the same way as ordinary atomic-orbital based methods are limited by the basis set. Instead, the grid-based representation allows us to explicitly optimize each molecular orbital directly. A convenient way to realize this is by convoluting with the Helmholtz kernel.[16,17,32,33] In this way, the exact HF energy can be approached with arbitrary accuracy.

3.1. Multiplication of Scalar Functions. Some examples of multiplications needed to construct the Fock matrix are $|\phi_i|^2$, $\rho(\mathbf{r})V(\mathbf{r})$, and in dot products, e.g., $\langle\phi_i|ket\rangle$. Although functions represented in the bubbles basis can be accurately multiplied in a pointwise manner, a more desirable approach is to obtain the resulting function directly in the bubbles representation.

Let us consider consider the product $f(\mathbf{r})g(\mathbf{r}) = h(\mathbf{r})$, which can be expanded as

$$\left[\sum_A f^A(r_A,\theta_A,\varphi_A) + f^\Delta(\mathbf{r})\right]\left[\sum_A g^A(r_A,\theta_A,\varphi_A) + g^\Delta(\mathbf{r})\right] = \sum_A h^A(r_A,\theta_A,\varphi_A) + h^\Delta(\mathbf{r}) \tag{18}$$

We previously proposed a scheme to directly compute the radial parts of the resulting function such that its cube $h^\Delta(\mathbf{r})$ is sufficiently smooth.[27] The radial functions are obtained as

$$h^{Alm}(r_A) = \sum_{l_1m_1}\sum_{l_2m_2} \langle Y_{lm}|Y_{l_1m_1}Y_{l_2m_2}\rangle [f^{Al_1m_1}(r_A)\tilde{g}^{Al_2m_2}(r_A)\mu_f(r_A) + \tilde{f}^{Al_1m_1}(r_A)g^{Al_2m_2}(r_A)\mu_g(r_A)] \tag{19}$$

The radial functions $\tilde{f}^{Alm}(r_A)$ in eq 19 are obtained by expanding $f - f^A = \sum_{B\neq A} f^B + f^\Delta$ around $\mathbf{R}_A$ in a truncated Taylor series of order T, which is projected onto the spherical harmonics basis. The same procedure is used to compute $\tilde{g}^{Alm}(r_A)$.

The $\mu_f(r)$ and $\mu_g(r)$ functions in eq 19 enforce a correct long-range behavior and depend on the properties of $f(\mathbf{r})$ and $g(\mathbf{r})$. When $f(\mathbf{r})$ has no cusps (e.g., it is the result of a convolution with the Poisson kernel), $\mu_f(r) = 0$. Otherwise, when it has cusps (e.g., it is a molecular orbital), $\mu_f(r) = \omega(r)$, with $\omega(r)$ being the flat function

$$\omega(r) = \frac{1}{2}\mathrm{erfc}\left(r - \frac{2}{r}\right) \tag{20}$$

shown in Figure 4. The same applies to $g(\mathbf{r})$ and $\mu_g(r)$.

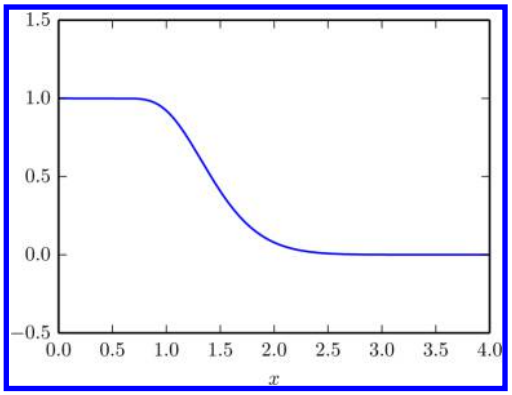


Figure 4. Flat function $\omega(r)$ given by eq 20

The L value of the resulting function is the sum of the L values of the multiplicands (or the order of their Taylor series, whichever is larger), although it can be truncated to a smaller number. In such case, all the discarded higher l-number functions are represented in the cube.

Obtaining the radial parts as in eq 19 is computationally simple. The coefficients of the product function in the cube are then computed point-by-point

$$h^\Delta_{ijk} = \left[f^\Delta_{ijk} + \sum_{A=1}^{N_{\mathrm{at}}} f^A(x_i,y_j,z_k)\right]\left[g^\Delta_{ijk} + \sum_{A=1}^{N_{\mathrm{at}}} g^A(x_i,y_j,z_k)\right] - \sum_{A=1}^{N_{\mathrm{at}}} h^A(x_i,y_j,z_k) \tag{21}$$

This is the main computational bottleneck because terms like $\sum_{A=1}^{N_{\mathrm{at}}} f^A(x_i,y_j,z_k)$ require the evaluation of the bubbles of $f(\mathbf{r})$, $g(\mathbf{r})$, and $h(\mathbf{r})$ at every grid point of the cube. We denote this operation *bubbles injection*.

The algorithm is outlined in Figure 5. First, we precompute the coefficients of the interpolating polynomials. That is, for each cell i, determine the coefficients a_{ij} that satisfy

$$\sum_{j=0}^{P} a_{ij}\left(\frac{x - s_i}{h_i}\right)^j = \sum_{k=(i-1)P+1}^{iP+1} f_i\chi_k(x) \qquad s_i \leq x < s_i + Ph_i \tag{22}$$

DOI: 10.1021/ct501128u
J. Chem. Theory Comput. 2015, 11, 2053−2062

for all A bubbles **do**
- *Precompute interpolating polynomials*

for all $l = 0, \ldots, L$ **do**
for all $m = -l, \ldots, l$ **do**
for all $i = 1, \ldots, M_A$ **do**
Compute a_{ij}^{Alm} such that
$\sum_j a_{ij}^{Alm}((x - s_i)/h_i)^j = \sum_j f_i^{Alm} b_{ij}(x)$

- *Interpolate the bubbles*

for all $\mathbf{r}_{\alpha\beta\gamma} = (x_\alpha, y_\beta, z_\gamma)$ grid points **do**
- *Compute relative coordinates, distance, unit vector*

$\mathbf{r}_A \leftarrow \mathbf{r}_{\alpha\beta\gamma} - \mathbf{R}_A$
$r_A \leftarrow |\mathbf{r}_A|$
$\hat{\mathbf{r}}_A \leftarrow \mathbf{r}_A / r_A$
- *Find cell*

Find i such that $s_i \le r_A \le s_i + Ph_i$
- *Compute local coordinate*

$q \leftarrow (r_A - s_i)/h_i$
for all $0 \le l \le L$, $-l \le m \le l$ **do**
- *Evaluate interpolating polynomial*

$f \leftarrow \sum_{0 \le j \le P} a_{ij}^{Alm} q^j$
- *Evaluate spherical harmonic*

for all (u, v, w) such that $C_{uvw}^{lm} \neq 0$ **do**
$Y \leftarrow Y + C_{uvw}^{lm} \hat{x}_A^u \hat{y}_A^v \hat{z}_A^w$
- *Accumulate result*

$f_{\alpha\beta\gamma} \leftarrow f_{\alpha\beta\gamma} + fY$

Figure 5. Algorithm for the bubbles injection for a function $f(\mathbf{r})$ that is computing $f_{\alpha\beta\gamma} = \sum_{A=1}^{N_{at}} f^A(x_\alpha y_\beta z_\gamma)$ for every grid point $\mathbf{r}_{\alpha\beta\gamma}$.

The computational cost for this is negligible. Then, for every grid point, the bubble is interpolated. This consists of a series of steps, namely, computing the relative coordinates of the grid point respect to the bubble center, finding the cell in the radial grid, transforming to cell coordinates, and, for every l and m, interpolating the radial function and evaluating the spherical harmonic.

Since the number of cube grid points is large and the operation must be performed for each one of the $f(\mathbf{r})$, $g(\mathbf{r})$, and $h(\mathbf{r})$ functions, multiplications are costly. However, all loops can be run in any order. In particular, the loop over cube grid points requires no communication whatsoever, making the operation embarrassingly parallel. Moreover, the operations for different grid points are exactly identical, making them ideal for execution on SIMD architectures. This also holds for finding the cell in the radial grid, as it can be done unconditionally by performing a binary search with $\log_2(M_A + 1)$ comparisons. The total cost for the injection is, in practice, proportional to the number of interpolations, $N_{cub}N_{at}(L + 1)^2$, which, in principle, scales quadratically with N_{at}. However, taking into account that the bubbles have a finite radial extent, the actual scaling is asymptotically linear $O(N_{at})$. More accurate radial grids of the bubbles can be used when needed because the computational cost increases very little with the number of radial grid points. Note that simply increasing the number of radial grid points will have no impact in the overall accuracy because the limiting factor is the quality of the cube grid.

3.2. Linear Transformations. The two types of linear transformations appearing in Figure 3 are calculations of electrostatic potentials (e.g., $\int_{\mathbb{R}^3} \phi_i(\mathbf{r}')\, \phi_j(\mathbf{r}') |\mathbf{r} - \mathbf{r}'|^{-1} \mathrm{d}^3 r'$) and Laplacians ($-(1/2)\nabla^2 \phi_j$). The effect of linear operators can be computed independently for the bubbles and cube parts of the functions. For an input function $f(\mathbf{r})$ and an output function $g(\mathbf{r})$, we have

$$g(\mathbf{r}) = \hat{O} f(\mathbf{r}) = \sum_{A=1}^{N_{at}} \hat{O} f^A(r_A, \theta_A, \varphi_A) + \hat{O} f^\Delta(\mathbf{r}) \tag{23}$$

The effect of the operator on the bubbles can be computed efficiently and accurately when the operator can be expressed in spherical coordinates in a manageable form. For the cube, the linear operators are expressed in a Cartesian-separated form of rank R

$$\hat{O} \approx \sum_{p=1}^{R} \omega_p \hat{O}^{x,p} \hat{O}^{y,p} \hat{O}^{z,p} + C\hat{I} \tag{24}$$

where R is the *rank* of the operator, $\hat{I}$ is the identity operator, and C and $\{\omega_p | 1 \le p \le R\}$ are constants. The coefficients of g^Δ can then be obtained as

$$g_{ijk}^\Delta \approx \sum_{p=1}^{R} \omega_p \sum_{k'=1}^{N_z} O_{kk'}^{z,p} \sum_{j'=1}^{N_y} O_{jj'}^{y,p} \sum_{i'=1}^{N_x} O_{ii'}^{x,p} f_{i'j'k'}^\Delta + C f_{ijk}^\Delta \tag{25}$$

The elements of the $\mathbf{O}^p$ matrices are

$$O_{ii'}^{\xi,p} = \hat{O}^{\xi,p} \chi_{i'}(\xi)|_{\xi = \xi_i} \tag{26}$$

The i', j', and k' indices refer to elements of the input grid, and the i, j, and k indices, to elements of the output grid. The input and output grids may be different. The rank R, the explicit form of the matrix elements of the operator matrices, the ω_p values, and C coefficient depend on the operator in question.

The expression in 25 can be written as a series of matrix multiplications. Following the notation of Kolda et al.,[34] we denote the three-index tensor containing the elements f_{ijk}^Δ as $\mathbf{F}$ and the tensor of the cube of the output function as $\mathbf{G}$. $\mathbf{A}$ and $\mathbf{B}$ are, respectively, intermediate two- and three-dimensional tensors. Two-index slices are expressed as, e.g., $\mathbf{F}_{:i:}$, which is a matrix whose elements are $(\mathbf{F}_{:i:})_{jk} = (\mathbf{F})_{jik}$. The operator matrices are stored as three-dimensional tensors $\mathbf{O}^x$, $\mathbf{O}^y$, and $\mathbf{O}^z$, with $(\mathbf{O}^\xi)_{i,i',p} = O_{ii'}^{\xi,p}$. The algorithm is outlined in Figure 6.

for all $1 \le p \le R$ **do**
for all k' slices along the z axis **do**
$\mathbf{A} \leftarrow \mathbf{O}_{::p}^x \mathbf{F}_{::k'}$
$\mathbf{B}_{::k'} \leftarrow \mathbf{A}(\mathbf{O}_{::p}^y)^{\mathbf{T}}$
for all j slices along the y axis **do**
$\mathbf{G}_{:j:} \leftarrow \mathbf{G}_{:j:} + \omega_p \mathbf{B}_{:j:}(\mathbf{O}_{::p}^z)^{\mathbf{T}}$
$\mathbf{G} \leftarrow \mathbf{G} + C\mathbf{F}$

Figure 6. Algorithm for linear transformations.

The operation consists of a series of matrix multiplications. The matrix multiplications are carried out in the order x, y, and then z because the elements of the cube and the operator matrices are stored in column-major order. The total number of floating-point operations is given by $2RN_xN_yN_z(N_x + N_y + N_z + 2)$, or for a cubic grid with $N = N_x = N_y = N_z$, it is approximately $N(6RN^3 + 2) \sim 6RN^4$. Because of the linear increase of N_{cub} with the number of atoms N_{at}, the computational cost is expected to grow as $O(RN_{at}^{4/3})$.

3.2.1. Convolution with the Poisson Kernel. Electrostatic potentials are computed by direct integration

$$V(\mathbf{r}) = \int_{\mathbb{R}^3} \frac{\rho(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} \mathrm{d}^3 r' \tag{27}$$

DOI: 10.1021/ct501128u
J. Chem. Theory Comput. 2015, 11, 2053−2062

For the bubbles, the resulting radial functions are given by[35]

$$V^{Alm}(r_A) = \frac{4\pi}{2l+1}\left[r_A^{-(l+1)} \int_0^{r_A} \rho^{Alm}(r'_A)\, r'^{l+2}_A \, \mathrm{d}r'_A + r_A^l \int_{r_A}^{\infty} \rho^{Alm}(r'_A)\, r'^{1-l}_A \, \mathrm{d}r'_A \right] \tag{28}$$

The integrations in 28 are computed very accurately in the polynomial basis, with a negligible cost.

For the cube, the Coulomb potential is approximated using a numerical quadrature[27,36] of the well-known integral identity[37,38]

$$\frac{1}{r_{12}} = \frac{2}{\sqrt{\pi}} \int_0^{\infty} \mathrm{e}^{-t^2 r_{12}^2}\, \mathrm{d}t \approx \sum_{p=1}^{R} \omega_p\, \mathrm{e}^{-t_p^2 r_{12}^2} + \frac{\pi}{t_f^2}\delta(\mathbf{r}_1 - \mathbf{r}_2) \tag{29}$$

The details on how to obtain the points t_p and weights ω_p can be found in our previous work.[27,36] A key advantage of this quadrature approach is that the accuracy can be systematically improved such that it can provide machine precision for two- and three-electron integrals over GTOs.[39,40] The quadrature parameters ($t_l = 2$, $t_f = 500$, $N_{\mathrm{lin}} = 20$, $N_{\mathrm{log}} = 16$), for a total operator rank of $R = 36$, are chosen such that the grid step determines the accuracy of the calculation.

3.2.2. Laplacian. The effect of the Laplacian on the bubbles can be computed by means of the following expression

$$\nabla^2 f^{Alm}(r)\, Y_{lm}(\theta, \varphi) = r^{-2}\left[\frac{\mathrm{d}}{\mathrm{d}r}\left(r^2 \frac{\mathrm{d}}{\mathrm{d}r} f^{Alm}(r) \right) - l(l+1) f^{Alm}(r) \right] Y_{lm}(\theta, \varphi) \tag{30}$$

The r^{-2} factor is stored implicitly by setting $k_g = k_f - 2$ (see eq 5). In this way, the problematic singularity can be treated accurately without additional costs.

For the cube, the Laplacian can be exactly represented as an operator of rank $R = 3$, with the coefficients appearing in eq 25 set to $\omega_1 = \omega_2 = \omega_3 = 1$, $C = 0$, and the operator matrix elements given by

$$O_{ii'}^{x,1} = \partial_x^2 \chi_{i'}^x(x_i);\quad O_{jj'}^{y,1} = \chi_{j'}^y(y_j);\quad O_{kk'}^{z,1} = \chi_{k'}^z(z_k) \tag{31}$$

$$O_{ii'}^{x,2} = \chi_{i'}^x(x_i);\quad O_{jj'}^{y,2} = \partial_y^2 \chi_{j'}^y(y_j);\quad O_{kk'}^{z,2} = \chi_{k'}^z(z_k) \tag{32}$$

$$O_{ii'}^{x,3} = \chi_{i'}^x(x_i);\quad O_{jj'}^{y,3} = \chi_{j'}^y(y_j);\quad O_{kk'}^{z,3} = \partial_z^2 \chi_{k'}^z(z_k) \tag{33}$$

Some loss of accuracy is expected, which occurs every time numerical differentiation is performed. However, the applications below show that the error can be kept within acceptable limits.

4. GPGPU IMPLEMENTATION AND BENCHMARKING

In this section, we describe the GPGPU implementation of the operations described in Sections 3.1 and 3.2, and we assess their performance using some model systems. We also present timings for the same molecular systems that we studied in our previous work.[27] All timings include the transfer of data between the main memory and the GPGPU.

The calculations have been performed on CSC's Taito supercluster, on Nvidia Tesla K40 graphics cards, using CUDA 7.0. The cards have a theoretical peak performance of 1.43 TFLOPs for double precision floating-point operations and 12 GB of memory. The computing nodes where they are installed are equipped with Intel Xeon E5-2620-v2 CPUs (2.67 GHz) and 32 GB of memory.

The performance is reported as number operations per second, where *operation* is chosen such that it can be easily calculated based solely on the input parameters. The presented algorithms have been designed to be run exclusively on many-core architectures, and the serial performance is meaningless. For this reason, we do not report any speedup factors. When speedup factors are given, they provide only a rough estimate of the improved computational efficiency. As many-core parallel platforms become more important in the future, comparisons with the performance of single CPU architectures should be abandoned in favor of more objective measures.

4.1. Multiplications. As discussed in Section 3.1, the injection operation is embarrassingly parallel over the grid points of the cube. Its implementation as a CUDA C kernel is straightforward, involving no changes in the algorithm, besides striving to reduce the number of registers used. Initially, the required data structures (spherical harmonics, cube grid) are transferred to the GPGPU, and the cube is allocated. Then, for each bubble, the radial grids and expansion coefficients are copied to the GPGPU, and the injection is carried out for that bubble. Finally, the resulting cube is copied back to main memory.

There are many possibilities for implementing the parallelization and to map the operations to CUDA threads and blocks. The algorithm in Figure 5 suggests that each thread computes the contributions from all (l,m) pairs of one given bubble at a given grid point $\mathbf{r}_{\alpha\beta\gamma}$ to avoid redundant calculations of, e.g., relative coordinates. There are thus two possibilities for distributing the workload, namely, that each block takes care of a different bubble with the outer loop running over clusters of points or vice versa. In the first approach, with one bubble per block and all blocks targeting the same cluster of points, shared memory can be used for gathering the accumulated results. However, the second approach, with one cluster of points per block and all blocks interpolating the same bubble, offers superior overall performance. The reason is that this mapping scheme saturates the GPGPU more efficiently because the number of point blocks amounts to tens of thousands for a typical grid with 200^3 points that is subdivided in clusters of 8^3 points, whereas the number of atoms is much smaller. The main disadvantage of the second approach is that the results have to be accumulated directly into the global memory. The optimal block size was found to be 128 threads, such that each block evaluates clusters of $8 \times 4 \times 4$ points. The seemingly small block size is due to the large number of registers needed by each thread to store all local variables. Using larger blocks severely decreases the amount of blocks that can be executed simultaneously, leading to a reduced efficiency. Clusters of points are chosen to be as cubic as possible to exploit caching because nearby points will very likely interpolate from nearby radial cells.

The performance of the injection was tested for different N_{at}, L, and $N = N_x = N_y = N_z$ values with $Z = 1$. As a measurement of the performance, we use the number of interpolations per second, computed as $N^3 N_{\mathrm{at}}(L+1)^2$ divided by the total computing time, including data transfers. Note that an actual two-function product requires three injections.

Figure 7 shows performance as a function of the number of cube grid points per dimension N. The performance is saturated for grids of a size of about $150 \times 150 \times 150$. Saturation with respect to L and N_{at} is slower. For $L = 2$ and $N_{\mathrm{at}} \leq 5$, the performance is about 1.5−3 billion interpolations per second.

DOI: 10.1021/ct501128u
J. Chem. Theory Comput. 2015, 11, 2053−2062

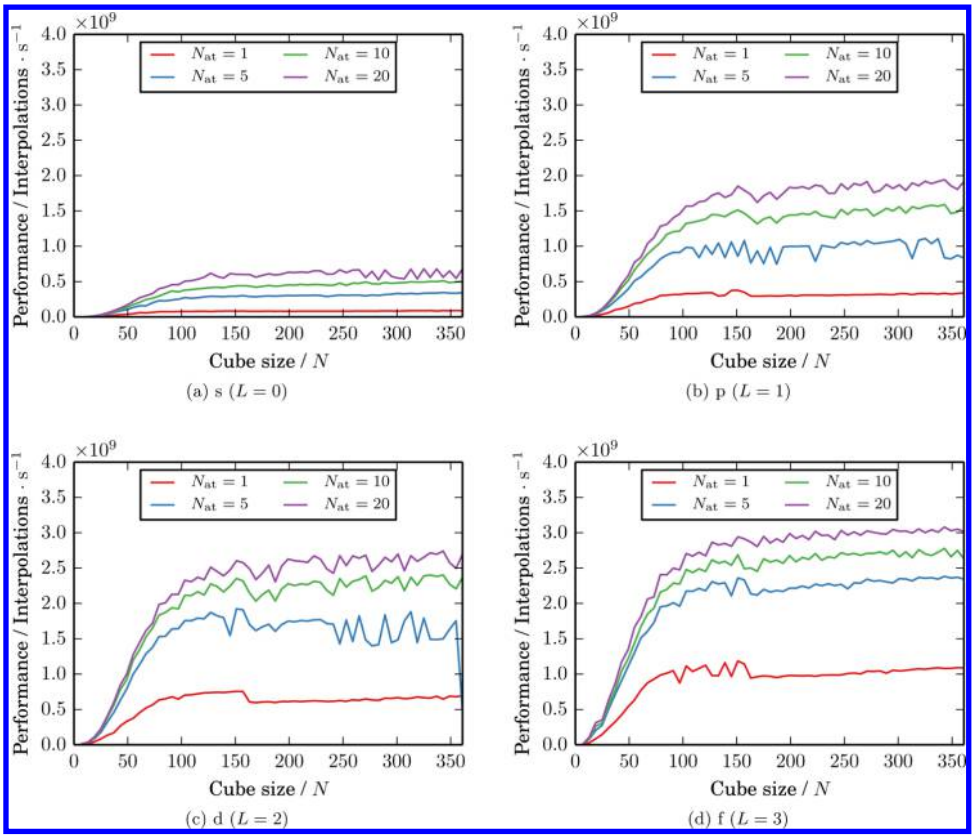


Figure 7. Performance for the bubbles injection in number of interpolations per second, for different values of L and number of bubbles N_{at} as a function of the number of points per dimension N.

For example, for a cube grid of 200 × 200 × 200 and $L = 2$, the approximate computation time is 25−50 ms.

4.2. Linear Transformations. We based our GPGPU implementation of the linear transformations on the cuBLAS library (version 7.0), which is Nvidia's GPGPU port of the widespread BLAS library.[41] Matrix algebra operations are typically repetitive and therefore amenable for implementation on GPGPUs. GPGPU-accelerated matrix multiplications are indeed employed in quantum chemical contexts.[42−45] In our implementation, we first transfer the input tensors ($\mathbf{F}$, $\mathbf{O}^x$, $\mathbf{O}^y$, and $\mathbf{O}^z$) and allocate the output tensor $\mathbf{V}$ and the intermediate tensors $\mathbf{G}$ and $\mathbf{H}$. The total memory requirements are $3N^3 + 3N^2R + N^2$ floating-point numbers. Then, the R series of multiplications are carried out and $C\mathbf{F}$ is added, as described in Figure 6. Finally, the output tensor is downloaded to the main memory. We tested the cuBLAS threaded version of the matrix multiplication, parallelizing over R. However, this did not significantly affect performance. This is possibly due to the need to allocate larger intermediate tensors, which, in turn, limits the amount of available memory and hence the number of concurrent threads. No other parameters controlling the execution were modified.

We benchmarked the linear transformation for cubic grids of different N up to 361 ($M = 60$) and R ranging from 1 to 60. The performance in GFLOPs is shown in Figure 8 for some selected values of R. Performance increases both with R and N as the calculation becomes heavier and the ratio of computing time to data transfer increases. For $N > 200$, the performance behaves more irregularly, dropping for some seemingly unsuitable matrix sizes. Nevertheless, for $R \geq 20$, it remains above 500 GFLOPs. The "saw-teeth" with periods of 64 are much more pronounced in the K40 cards than what we observed in models based on older CUDA architectures, such as the C1040 or M2070 cards.

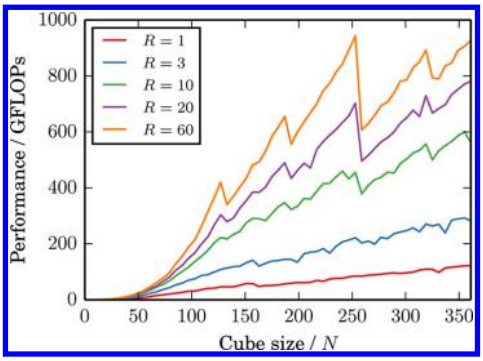


Figure 8. Performance in GFLOPs of the linear transformations for selected values of the operator rank R as a function of the number of points per dimension $N = 6M + 1$, for $M \leq 60$.

Generally, one would not expect such high performance on GPGPUs for matrices of the size discussed here. For instance, in the work of Olivares-Amaya et al.,[43] it is clear that, in order to fully use the power of the GPGPU, matrices of several thousand elements per dimension need to be considered. The reason for the excellent performance obtained here is that $3RN$ matrix multiplications are computed sequentially with the GPGPU−CPU transfers occurring only at the beginning and at the end.

4.3. Test Calculations on Molecular Systems. The accuracy of the scheme has been assessed on a set of 19 closed-shell molecules, including up to 15 atoms and 90 electrons.[27] The molecular structures were optimized with Turbomole[46] at the density functional theory (DFT) level using the BP86 functional and def-SV(P) basis sets.[47−50] The orbitals and the total energies were obtained in single-point HF calculations using the same basis set. The N_{occ} occupied MOs were then reconstructed in the

DOI: 10.1021/ct501128u
J. Chem. Theory Comput. 2015, 11, 2053−2062

bubbles representation. L was set to 2, sufficient to accommodate all atomic orbitals in the bubbles, such that the cubes are zero. The Fock matrix was then built in the MO basis, using only the occupied MOs. To assess the accuracy, the total RHF energy was computed numerically. For each molecule, the calculations were performed using four different cube grid steps: 0.4 a_0, 0.2 a_0, 0.1 a_0, and 0.05 a_0. The detailed results of the calculations can be found as Table S1 in the Supporting Information.

The errors in the integrated number of electrons and the total RHF energies are shown in Figures 9 and 10, respectively.

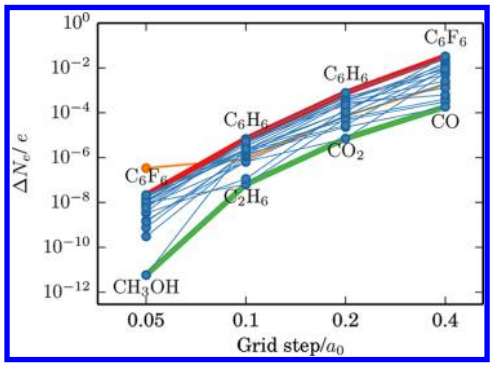


Figure 9. Errors in the integrated number of electrons as a function of the grid step. Individual trends for each molecule are shown in blue. The curve for ZnH_2 is shown separately in orange. The molecules that incurred into the maximum and minimum errors are highlighted in red and green, respectively.

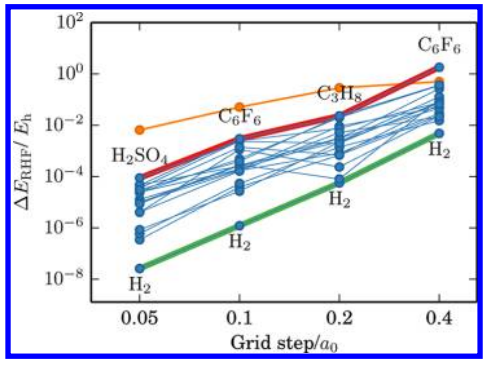


Figure 10. Errors in the RHF energy as a function of the grid step. Individual trends for each molecule are shown in blue. The curve for ZnH_2 is shown separately in orange. The molecules that incurred into the maximum and minimum errors are highlighted in red and green, respectively.

The error was at most 3 mE_h for the 0.1a_0 grid and 90μE_h for the 0.05a_0 grid, except for ZnH_2, which is the only molecule that contains a fourth-row element. For ZnH_2, the errors also decrease systematically, although they were about 2 orders of magnitude larger than those for the rest of the molecules. In Figure 9, one sees that the electron density of ZnH_2 is reproduced as accurately as for the other systems, except for the finest grid. In our previous work,[27] no particular problems were observed for the self-interaction energy of the electron density of ZnH_2. Therefore, the errors seem to arise from the calculation of the kinetic energy. We will examine this issue in more detail in the future.

For the application of linear operators, the total number of floating-point operations, taking into account the number of occupied orbitals N_{occ} and ranks R of the operators, is given in Table 1. The grids with 0.1a_0 and 0.05 a_0 steps are sufficient to

Table 1. Operation Count for the Injections

operator	count
∇^2 (R = 3)	N_{occ}
Coulomb (R = 36)	$N_{occ}(3N_{occ} + 1)/2$
total FLOP count	$6N_{occ}(18N_{occ} + 7)N_xN_yN_z(N_x + N_y + N_z)$

saturate the GPGPUs. The observed performance ranges between 0.5 and 0.9 TFLOPs. For a cube of 200 × 200 × 200 and R = 36, this translates to a total computation time of 2 to 3.4 s per Coulomb operator application.

For the injection, the total number of floating-point operations is difficult to estimate, and, in any case, it does not translate easily between different implementations. Hence, the number of interpolations is a more adequate measurement. The total number of injections and total number of interpolations are given in Table 2. The increase in L results from the multiplication of

Table 2. Operation Count for the Cube Linear Transformations

L	no. of injections
2	$N_{occ}(11N_{occ} + 9)/2$
4	$N_{occ}(4N_{occ} + 3)$
6	$N_{occ}(3N_{occ} + 1)/2$
8	$N_{occ}(N_{occ} + 1)$
total interpolation count	$N_{occ}(304N_{occ} + 221)N_{at}N_xN_yN_z$

functions, e.g., the product of two functions with L = 2 results in a function with L = 4. Again, for this operation, the grids with 0.1a_0 and 0.05a_0 steps provided sufficient workload to saturate the GPGPUs. The performance ranged from 1.5 to 2.3 billion interpolations per second. For L = 2 and a cube of 200 × 200 × 200, the computation time is estimated as 30−50 μs per atomic center.

The total computational cost of one construction of the Fock matrix is approximately proportional to $N_{occ}^2N_{tot}$, which asymptotically corresponds to $O(N_{at}^3)$, as expected for the current naive implementation. In Figure 11, it is shown how the

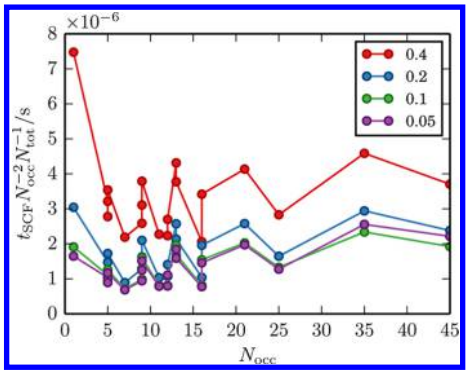


Figure 11. Total time for the construction of the Fock matrix divided by $N_{occ}^2N_{tot}$, as measurement of the overall performance for all test molecules and grid steps.

total time divided by $N_{occ}^2N_{tot}$ remains approximately constant at 1−3 μs (except for the 0.4 a_0 grid, which performs worse by a factor of ca. 2). Most of the time is taken up by injections (40−80%) and linear transformations (10−20%), as expected.

Constructing the Fock matrix with the 0.1a_0 step grid takes a few minutes for smaller systems, whereas larger molecules take

DOI: 10.1021/ct501128u
J. Chem. Theory Comput. 2015, 11, 2053−2062

several hours. For instance, for CO_2 it takes less than 10 min, whereas the calculation on C_6F_6 took about 12 h.

5. CONCLUSIONS AND OUTLOOK

In this work, we have outlined and benchmarked a GPGPU-based algorithm for the construction of the all-electron Fock matrix in the MO basis using the numerical *bubbles* representation. We have shown that numerical electronic structure methods are suitable for the emerging parallel computer architectures. Although the number of arithmetic operations is very large, the operations can be parallelized very efficiently, and computing the Fock matrix becomes feasible on a single GPGPU card. The GPGPU implementation was straightforward, either by means of available libraries (cuBLAS) or by porting our code to CUDA.

The representation has been shown to be accurate enough to compute the Fock matrix to mE_h or better precision. This enables implementing self-consistent iterative methods to find the best linear combination of input molecular orbitals. The aim is to combine those well-known methods with techniques to directly optimize each molecular orbital. Such approaches have proven to be successful using multiwavelets and other similar grid-based representations.

When larger grids are employed for tackling large molecules or for improving the accuracy by using denser grids, the obvious bottleneck will be memory. The largest grids considered here consist of 500^3 grid points. Such calculations are approaching the memory limit of the K40 card, since almost 1 GB of memory per function is required. To overcome the memory bottleneck, we are exploring parallelization via domain partitioning. In such an approach, the computational domain is subdivided into a number of subdomains, which are assigned to different nodes equipped with a GPGPU card. The calculations suggest that each subdomain should consist of about $200 \times 200 \times 200$ grid points. Expensive internode communication can be minimized by using a modified version of the fast multipole method,[52,53] enabling scalability to much larger grids than presented in this work. Previous experience in parallelization of grid-based methods also indicates that the computational time can be reduced by orders of magnitude by employing a large number of GPGPUs.[16,51]

In summary, the work presented here is a stepping stone in the development of a fully numerical electronic structure methodology that is aimed for GPGPU clusters. In the future, grid-based approaches will most likely be used for production calculations when GPGPU and other similar architectures become more common and algorithms such as those presented in this work are mature for that.

■ ASSOCIATED CONTENT

Supporting Information

Numerical values obtained in the test calculations. This material is available free of charge via the Internet at http://pubs.acs.org.

■ AUTHOR INFORMATION

Corresponding Author

*E-mail: sundholm@chem.helsinki.fi.

Present Address

[‖](S.A.L.) Department of Chemistry, University of Aarhus, DK-8000 Aarhus C, Denmark.

Funding

This research has been supported by the Academy of Finland through project nos. 137460 and 275845 and its Computational Science Research Programme (LASTU/258258).

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

CSC, the Finnish IT Center for Science, is acknowledged for computer time, and the Magnus Ehrnrooth Foundation, for travel money. The authors would also like to thank Javier Gómez and Stig-Rune Jensen for helpful discussions and Elias Toivanen, Dou Du, and Jonas Jusélius for support with the development and maintenance of the code. A.A.-G. acknowledges support from the National Science Foundation under award no. OIA-1125087 as well as support from the Corning Foundation.

■ REFERENCES

(1) Slater, J. C. Atomic Shielding Constants. *Phys. Rev.* **1930**, *36*, 57−64.

(2) Boys, S. F. Electronic wave functions I. A general method of calculation for the stationary states of any molecular system. *Proc. R. Soc. London* **1950**, *200*, 542−554.

(3) Blum, V.; Gehrke, R.; Hanke, F.; Havu, P.; Havu, V.; Ren, X.; Reuter, K.; Scheffler, M. Ab initio molecular simulations with numeric atom-centered orbitals. *Comput. Phys. Commun.* **2009**, *180*, 2175−2196.

(4) Delley, B. An all-electron numerical method for solving the local density functional for polyatomic molecules. *J. Chem. Phys.* **1990**, *92*, 508−517.

(5) Klopper, W.; Kutzelnigg, W. Gaussian basis sets and the nuclear cusp problem. *J. Mol. Struct.: THEOCHEM* **1986**, *135*, 339−356.

(6) Kutzelnigg, W.; Morgan, J. D. Rates of convergence of the partial-wave expansions of atomic correlation energies. *J. Chem. Phys.* **1992**, *96*, 4484−4508.

(7) Carroll, D. P.; Silverstone, H. J.; Metzger, R. M. Piecewise polynomial configuration interaction natural orbital study of 1 s^2 helium. *J. Chem. Phys.* **1979**, *71*, 4142−4163.

(8) Manninen, P.; Vaara, J. Systematic Gaussian basis-set limit using completeness-optimized primitive sets. A case for magnetic properties. *J. Comput. Chem.* **2006**, *27*, 434−445.

(9) Sundholm, D.; Olsen, J. Large MCHF calculations on the hyperfine structure of B(^{2}P) and the nuclear quadrupole moments of ^{10}B and ^{11}B. *J. Chem. Phys.* **1991**, *94*, 5051−5055.

(10) Froese Fischer, C. Multi-configuration Hartree−Fock program with improved stability. *Comput. Phys. Commun.* **1972**, *4*, 107−116.

(11) McCullough, E. A., Jr. Seminumerical SCF calculations on small diatomic molecules. *Chem. Phys. Lett.* **1974**, *24*, 55−58.

(12) McCullough, E. A., Jr. Partial-wave self-consistent-field method for diatomic molecules—computational formalism and results for small molecules. *J. Chem. Phys.* **1975**, *62*, 3991−3999.

(13) Laaksonen, L.; Pyykkö, P.; Sundholm, D. Two-dimensional fully numerical solutions of molecular Schrödinger equations. I. One-electron molecules. *Int. J. Quantum Chem.* **1983**, *23*, 309−317.

(14) Laaksonen, L.; Pyykkö, P.; Sundholm, D. Two-dimensional fully numerical solutions of molecular Schrödinger equations. II. Solution of the Poisson equation and results for singlet states of H_2 and HeH^+. *Int. J. Quantum Chem.* **1983**, *23*, 319−323.

(15) Kobus, J.; Laaksonen, L.; Sundholm, D. A numerical Hartree−Fock program for diatomic molecules. *Comput. Phys. Commun.* **1996**, *98*, 346−358.

(16) Yanai, T.; Fann, G. I.; Gan, Z.; Harrison, R. J.; Beylkin, G. Multiresolution quantum chemistry in multiwavelet bases: analytic derivatives for Hartree−Fock and density functional theory. *J. Chem. Phys.* **2004**, *121*, 2866−2876.

(17) Bischoff, F. A.; Valeev, E. F. Low-order tensor approximations for electronic wave functions: Hartree−Fock method with guaranteed precision. *J. Chem. Phys.* **2011**, *134*, 104104.

(18) Ufimtsev, I. S.; Martinez, T. J. Quantum chemistry on graphical processing units. 1. Strategies for two-electron integral evaluation. *J. Chem. Theory Comput.* **2008**, *4*, 222−231.

(19) Kussmann, J.; Ochsenfeld, C. Pre-selective screening for matrix elements in linear-scaling exact exchange calculations. *J. Chem. Phys.* **2013**, *138*, 134114.

DOI: 10.1021/ct501128u
J. Chem. Theory Comput. 2015, 11, 2053−2062

(20) Titov, A. V.; Ufimtsev, I. S.; Luehr, N.; Martinez, T. J. Generating efficient quantum chemistry codes for novel architectures. *J. Chem. Theory Comput.* **2013**, *9*, 213−221.
(21) Wilkinson, K. A.; Sherwood, P.; Guest, M. F.; Naidoo, K. J. Acceleration of the GAMESS-UK electronic structure package on graphical processing units. *J. Comput. Chem.* **2011**, *32*, 2313−2318.
(22) Genovese, L.; Ospici, M.; Deutsch, T.; Méhaut, J.-F.; Neelov, A.; Goedecker, S. Density functional theory calculation on many-cores hybrid central processing unit-graphic processing unit architectures. *J. Chem. Phys.* **2009**, *131*, 034103.
(23) Tomono, H.; Aoki, M.; Iitaka, T.; Tsumuraya, K. GPU based acceleration of first principles calculation. *J. Phys.: Conf. Ser.* **2010**, *215*, 012121.
(24) Andrade, X.; Alberdi-Rodriguez, J.; Strubbe, D. A.; Oliveira, M. J. T.; Nogueira, F.; Castro, A.; Muguerza, J.; Arruabarrena, A.; Louie, S. G.; Aspuru-Guzik, A.; Rubio, A.; Marques, M. A. L. Time-dependent density-functional theory in massively parallel computer architectures: the octopus project. *J. Phys.: Condens. Matter* **2012**, *24*, 233202.
(25) Yan, J.; Li, L.; O'Grady, C. Graphics processing unit acceleration of the random phase approximation in the projector augmented wave method. *Comput. Phys. Commun.* **2013**, *184*, 2728−2733.
(26) Andrade, X.; Aspuru-Guzik, A. Real-space density functional theory on graphical processing units: computational approach and comparison to Gaussian basis set methods. *J. Chem. Theory Comput.* **2013**, *9*, 4360−4373.
(27) Losilla, S. A.; Sundholm, D. A divide and conquer real-space approach for all-electron molecular electrostatic potentials and interaction energies. *J. Chem. Phys.* **2012**, *136*, 214104.
(28) Helgaker, T.; Jørgersen, P.; Olsen, J. *Molecular Electronic-Structure Theory*; John Wiley & Sons: Chichester, 2000.
(29) Yamakawa, S.; Hyodo, S.-A. Gaussian finite-element mixed-basis method for electronic structure calculations. *Phys. Rev. B* **2005**, *71*, 035113.
(30) Watson, M. A.; Hirao, K. A linear-scaling spectral-element method for computing electrostatic potentials. *J. Chem. Phys.* **2008**, *129*, 184107.
(31) Watson, M. A.; Kurashige, Y.; Nakajima, T.; Hirao, K. Linear-scaling multipole-accelerated Gaussian and finite-element Coulomb method. *J. Chem. Phys.* **2008**, *128*, 054105.
(32) Harrison, R. J.; Fann, G. I.; Yanai, T.; Gan, Z.; Beylkin, G. Multiresolution quantum chemistry: basic theory and initial applications. *J. Chem. Phys.* **2004**, *121*, 11587−98.
(33) Frediani, L.; Fossgaard, E.; Flå, T.; Ruud, K. Fully adaptive algorithms for multivariate integral equations using the non-standard form and multiwavelets with applications to the Poisson and bound-state Helmholtz kernels in three dimensions. *Mol. Phys.* **2013**, *111*, 1143−1160.
(34) Kolda, T. G.; Bader, B. W. Tensor decompositions and applications. *SIAM Rev.* **2009**, *51*, 455−500.
(35) Froese Fischer, C. *The Hartree−Fock Method for Atoms: A Numerical Approach*; John Wiley and Sons: New York, 1977.
(36) Losilla, S. A.; Sundholm, D.; Jusélius, J. The direct approach to gravitation and electrostatics method for periodic systems. *J. Chem. Phys.* **2010**, *132*, 024102.
(37) Singer, K. The use of Gaussian (exponential quadratic) wave functions in molecular problems. I. General formulae for the evaluation of integrals. *Proc. R. Soc. A* **1960**, *258*, 412−420.
(38) Boys, S. F. The integral formulae for the variational solution of the molecular many-electron wave equations in terms of Gaussian functions with direct electronic correlation. *Proc. R. Soc. A* **1960**, *258*, 402−411.
(39) Losilla, S.; Mehine, M.; Sundholm, D. Construction of the two-electron contribution to the Fock matrix by numerical integration. *Mol. Phys.* **2012**, *110*, 2569−2578.
(40) Mehine, M. M.; Losilla, S. A.; Sundholm, D.; Taylor, P. An efficient algorithm to calculate three-electron integrals for Gaussian-type orbitals using numerical integration. *Mol. Phys.* **2013**, *111*, 2536−2543.
(41) Blackford, L. S.; Demmel, J.; Dongarra, J.; Duff, I.; Hammarling, S.; Henry, G.; Heroux, M.; Kaufman, L.; Lumsdaine, A.; Petitet, A.; Pozo, R.; Remington, K.; Whaley, R. C. An updated set of basic linear algebra subprograms (BLAS). *ACM Trans. Math. Software* **2002**, *28*, 135−151.
(42) Vogt, L.; Olivares-Amaya, R.; Kermes, S.; Shao, Y.; Amador-Bedolla, C.; Aspuru-Guzik, A. Accelerating resolution-of-the-identity second-order Møller−Plesset quantum chemistry calculations with graphical processing units. *J. Phys. Chem. A* **2008**, *112*, 2049−2057.
(43) Olivares-Amaya, R.; Watson, M. A.; Edgar, R. G.; Vogt, L.; Shao, Y.; Aspuru-Guzik, A. Accelerating correlated quantum chemistry calculations using graphical processing units and a mixed precision matrix multiplication library. *J. Chem. Theory Comput.* **2010**, *6*, 135−144.
(44) Watson, M.; Olivares-Amaya, R.; Edgar, R. G.; Aspuru-Guzik, A. Accelerating correlated quantum chemistry calculations using graphical processing units. *Comput. Sci. Eng.* **2010**, *12*, 40−51.
(45) DePrince, A. E.; Hammond, J. R. Coupled cluster theory on graphics processing units I. The coupled cluster doubles method. *J. Chem. Theory Comput.* **2011**, *7*, 1287−1295.
(46) Ahlrichs, R.; Bär, M.; Häser, M.; Horn, H.; Kölmel, C. Electronic structure calculations on workstation computers: the program system TURBOMOLE. *Chem. Phys. Lett.* **1989**, *162*, 165−169. For the current version, see http://www.turbomole.com.
(47) Vosko, S. H.; Wilk, L.; Nusair, M. Accurate spin-dependent electron liquid correlation energies for local spin-density calculations—a critical analysis. *Can. J. Phys.* **1980**, *58*, 1200−1211.
(48) Perdew, J. P. Density-functional approximation for the correlation energy of the inhomogeneous electron gas. *Phys. Rev. B* **1986**, *33*, 8822−8824.
(49) Becke, A. D. Density-functional exchange-energy approximation with correct asymptotic behavior. *Phys. Rev. A* **1988**, *38*, 3098−3100.
(50) Schäfer, A.; Horn, H.; Ahlrichs, R. Fully optimized contracted Gaussian-basis sets for atoms Li to Kr. *J. Chem. Phys.* **1992**, *97*, 2571−2577.
(51) Jensen, S. R.; Jusélius, J.; Durdek, A.; Flå, T.; Wind, P.; Frediani, L. Linear scaling Coulomb interaction in the multiwavelet basis, a parallel implementation. *Int. J. Model., Simul., Sci Comput.* **2014**, *05*, 1441003.
(52) Toivanen, E. A.; Losilla, S. A.; Sundholm, D. The grid-based fast multipole method—a massively parallel numerical scheme for calculating two-electron interaction energies. *Phys. Chem. Chem. Phys.* **2015**, submitted.
(53) Greengard, L.; Rokhlin, V. A fast algorithm for particle simulations. *J. Comput. Phys.* **1987**, *73*, 325−348.

DOI: 10.1021/ct501128u
J. Chem. Theory Comput. 2015, 11, 2053−2062