LINEAR SCALING COULOMB INTERACTION IN THE MULTIWAVELET BASIS, A PARALLEL IMPLEMENTATION

STIG RUNE JENSEN

Centre for Theoretical and Computational Chemistry, Department of Physics and Technology, The Arctic University of Norway, N-9037 Tromsø, Norway

JONAS JUSÉLIUS

Centre for Theoretical and Computational Chemistry, High Performance Computing Group, The Arctic University of Norway, N-9037 Tromsø, Norway

ANTOINE DURDEK and TOR FLÅ

Centre for Theoretical and Computational Chemistry, Department of Mathematics and Statistics, The Arctic University of Norway, N-9037 Tromsø, Norway

PETER WIND and LUCA FREDIANI

Centre for Theoretical and Computational Chemistry, Department of Chemistry, The Arctic University of Norway, N-9037 Tromsø, Norway

luca.frediani@uit.no

We present a parallel and linear scaling implementation of the calculation of the electrostatic potential arising from an arbitrary charge distribution. Our approach is making use of the multi-resolution basis of multiwavelets. The potential is obtained as the direct solution of the Poisson equation in its Green's function integral form. In the multiwavelet basis the formally non-local integral operator decays rapidly to negligible values away from the main diagonal, yielding an effectively banded structure where the bandwidth is only dictated by the requested accuracy. This sparse operator structure has been exploited to achieve linearly scaling and parallel algorithms. Parallelization has been achieved both through the shared memory (OpenMP) and the message passing (MPI) paradigm.

Our implementation has been tested by computing the electrostatic potential of the electronic density of long-chain alkanes and diamond fragments showing (sub)linear scaling with the system size and efficent parallelization.

Keywords: Multiwavelets, electrostatic potentials, Poisson equation, integral operators, linear scaling, parallel implementation.

1. Introduction

One of the most widespread yet challenging physical problems is the calculation of the electromagnetic interactions between charge distributions.[1] Its applications range from engineering problems, to physics, chemistry and biology. The full solution is in principle described by the four Maxwell equations coupling the electric field, the magnetic field, the charge density and the charge current. Moreover, the presence of media and their interaction with the fields renders the problem even more complicated.

If we restrict ourselves to the realm of electrostatic interactions (time independent fields, and no current), the Maxwell equations can be reduced to the Poisson equation, linking the scalar electrostatic potential V to the charge density ρ:

$$\nabla \cdot \epsilon(r)\nabla V(r) = -4\pi\rho(r) \tag{1}$$

where the permittivity ϵ in general is position dependent. Most commonly, the charge distribution ρ is known and one is interested in obtaining its effect on the surroundings by computing the electrostatic potential. The formal solution for the free boundary problem with constant ϵ can be written in a closed form by making use of a Green's kernel formalism:

$$V(r) = \int_{\mathrm{R}^3} \frac{1}{|r - r'|}\rho(r')\,\mathrm{d}r' \tag{2}$$

From a computational point of view the challenge in modeling such a problem is twofold. On the one hand, the Green's kernel in Eq. (2) is non-separable: the Cartesian coordinates are coupled and the integral cannot be decomposed into the product of mono-dimensional integrals. On the other hand, the electrostatic interaction represented by the $1/r$ Green's kernel has a singularity at short distances and is decaying slowly for long distances; the singularity makes accurate computation of electrostatic interactions challenging and the long-range interaction makes the scaling with the system size challenging for straightforward computational approaches.

Several strategies have been devised to address the problem, depending e.g. on the boundary conditions and on the type of charge distribution. When possible the electrostatic equations are recasted onto a boundary problem thus limiting the system size to a two-dimensional surface instead of the whole three-dimensional space.[2;3;4;5] This is the preferred approach in case e.g. of solute-solvent interactions.[3;6] For charge distributions that can be conveniently described in terms of distributed multipoles, a very promising approach is constituted by the Fast Multipole Methods (FMM).[7;8;9;10] For more general cases real-space mesh methods must be employed.[11;12;13;14;15;16;17;18] Besides the generality, the advantage of such an approach is that it is well suited for modern parallel computing architectures: the parallelization is achieved by distributing the mesh among the compute nodes, although care must be taken in order to ensure a balanced workload.

The main disadvantage of real-space mesh methods is constituted by the large storage requirements implied.[13] This problem can be alleviated by making use of an adaptive multiwavelet approach where functions are represented on adaptive multi-resolution grids, the local refinement being dictated by a preselected accuracy.[19] The other advantage of a multiwavelet basis is constituted by the efficient representation of the integral operator in Eq. (2) which can be described in terms of narrow-banded matrices in the so called Non-Standard form.[20;21] By approximating the integral kernel in Eq. (2) as a sum of Gaussian functions, the operator is decomposed into several components, each of which is Cartesian separable and non-singular in the short-range limit. The prohibitive scaling in the long-range limit is taken care of by the multi-resolution analysis, as each operator component is further decomposed into different length scales. In this work we will present a practical implementation of such a method, with focus on computational efficiency and application to molecular systems.

2. Mathematical Background

We will briefly introduce the mathematical background of our approach which is based on multi-resolution analysis and the multiwavelet basis.[22]

2.1. *The multiwavelet basis*

Following the original construction from Alpert,[23] a one-dimensional multiwavelet basis is obtained by defining the multi-resolution scaling space V_k^n as the space of piecewise polynomials on the unit interval

$$\begin{aligned} V_k^n \overset{\text{def}}{=} \{\, & f : \text{all polynomials of degree } \leq k \\ & \text{on } (2^{-n}l, 2^{-n}(l+1)) \text{ for } 0 \leq l < 2^n, \\ & f \text{ vanishes elsewhere } \} \end{aligned} \tag{3}$$

From this definition Alpert shows that each space V_k^n is fully contained in all spaces of higher resolution $V_k^m, m > n$, and we can therefore consider a so called "ladder of spaces" with increasing flexibility

$$V_k^0 \subset V_k^1 \subset \ldots \subset V_k^n \subset \ldots \tag{4}$$

It is well known that the basis obtained by taking the limit for $k \to \infty$ is dense in the L_2 norm sense, and it has also been shown that the limit $n \to \infty$ is dense. In other words, any function of L_2 can be represented within any given accuracy by making use of a polynomial of sufficiently high order at a given spatial refinement, or with sufficiently high refinement for a given polynomial order. In practical applications it is generally convenient to find a good balance by increasing both simultaneously.

The wavelet spaces W_k^n can be formally constructed by taking the orthogonal complement between two successive scaling spaces:

$$W_k^n \oplus V_k^n = V_k^{n+1} \tag{5}$$

By construction the wavelet space W_k^n is orthogonal to all scaling spaces $V_k^m,\ m \leq n$. In other words the first $k+1$ moments of a wavelet basis ψ^n of the space W_k^n are zero

$$\int_0^1 x^m \psi^n(x)\,\mathrm{d}x = 0, \qquad m = 0, \ldots, k \tag{6}$$

which means that the basis is very efficient for the representation of smooth functions.

Eq. (5) does not completely define the basis. This flexibility can be exploited in order to obtain a basis with useful properties, e.g. additional vanishing moments and symmetry (see Alpert[23] for details).

The change of basis from the scaling basis ϕ_l^{n+1} defining V_k^{n+1} to the compound scaling ϕ_l^n and wavelet ψ_l^n basis at scale n is undertaken by making use of a unitary, local transformation called filters:

$$\begin{pmatrix} \psi_l^n \\ \phi_l^n \end{pmatrix} = \begin{pmatrix} G^{(1)} & G^{(0)} \\ H^{(1)} & H^{(0)} \end{pmatrix} \begin{pmatrix} \phi_{2l+1}^{n+1} \\ \phi_{2l}^{n+1} \end{pmatrix} \tag{7}$$

The locality of the transformation ensures that it can be performed in linear complexity. In addition, it also ensures that it can easily be implemented on distributed memory architectures.

Although the construction described applies to the one-dimensional case, the multi-dimensional extension can be obtained by making use of tensor-product bases, which in three dimensions yields

$$\Phi_{ijk}^n(x,y,z) = \phi_i^n(x)\phi_j^n(y)\phi_k^n(z) \tag{8}$$

To summarize, the scaling basis is equivalent with the more commonly employed basis of the finite element method (FEM): the three-dimensional unit cube is uniformly subdivided into cubic cells, each with a polynomial basis. What separates the multiwavelet basis from FEM is the additional wavelet (or difference) basis, which allows for adaptive (non-uniform) grids.

2.1.1. *Function representation*

Functions are formally represented in a multiwavelet basis by projection, and we define P_k^n and Q_k^n as the projection operators onto V_k^n and W_k^n, respectively. It is useful to point out that on the unit interval

$$P_k^n + Q_k^n = P_k^{n+1} \tag{9}$$

and

$$\lim_{n\to\infty} P_k^n = 1 \qquad \lim_{n\to\infty} Q_k^n = 0 \tag{10}$$

Eq. (9) follows from the construction of the wavelet basis, whereas Eq. (10) is due to the completeness in the L_2 sense. The projection of an arbitrary function f is denoted

$$f^n \overset{\text{def}}{=} P_k^n f \qquad \mathrm{d}f^n \overset{\text{def}}{=} Q_k^n f \tag{11}$$

Any smooth function can be approximated to any finite precision by a scaling projection with sufficiently high resolution N, and by recursive application of Eq. (9), this approximation can be decomposed into its multi-resolution components

$$f \approx f^N = f^0 + \sum_{n=0}^{N-1} \mathrm{d}f^n \tag{12}$$

Although mathematically equivalent, the representation in the combined (multi-resolution) scaling and wavelet basis in (RHS in Eq. (12)) has several advantages over the pure (high-resolution) scaling representation f^N. Because of the vanishing moments property of the wavelet basis (Eq. 6), the sum in Eq. (12) is rapidly converging for smooth functions and can be locally truncated to a predefined precision ϵ based on the wavelet norm $\|\,\mathrm{d}f_l^n\|$ at each interval $2^{-n}(l, l+1)$

$$\|\,\mathrm{d}f_l^n\| < \frac{\epsilon}{2^{n/2}}\|f\| \tag{13}$$

which can be used to build adaptive, function specific grids that dramatically reduces the number of coefficients needed to represent the function to the given accuracy.

The vanishing moments also means that the Coulomb interaction between charges represented in the wavelet basis decays very rapidly with distance, as the leading multipole order of this interaction is $k+1$. This is what ultimately allows for linear scaling algorithms, as the full (long-ranged) interaction is decomposed into different length scales, where the interaction is "local" at each length scale separately.

2.2. *Operator representation*

In order to apply the Poisson operator efficiently in three dimensions it is crucial to achieve an optimal representation. Several properties need to be considered:

(1) In order to exploit the tensorial representation of the basis, a representation that separates the Cartesian coordinates needs to be employed.
(2) Coupling between different length scales should be avoided in order to limit communication and to exploit adaptivity for function representations.
(3) The operator representation should ideally be banded in order to limit the coupling to the minimum necessary.

The first point is achieved by constructing a separated representation of the Poisson kernel in terms of Gaussian functions:

$$\frac{1}{r} \simeq K_M(r) = \sum_{i=1}^{M} a_i e^{-\alpha_i r^2} \tag{14}$$

where the components of the expansion are determined by an efficient quadrature scheme.[24] Although the Poisson kernel is not separable, each term in Eq. (14) is, and the expansion can be made arbitrarily accurate since for any given ϵ it is possible to find a rank M such that

$$\left| \frac{K_M(r) - 1/r}{1/r} \right| < \epsilon \tag{15}$$

is fulfilled within any given interval $r \in [r_0, r_1]$, where r_0 is chosen so that the contribution due to the integration at the singularity can be neglected,[25] and r_1 is the longest possible distance in the domain ($\sqrt{3}$ for the unit cube).

The second and third points are achieved through the so called Non-Standard form of the operator.[20;21] The application of an operator T can be written as

$$g(x) = [Tf](x) \tag{16}$$

which can be discretized in a Galerkin scheme by projecting both the functions and the operator on a given scaling space V_k^n

$$T_k^n \overset{\text{def}}{=} P_k^n T P_k^n \tag{17}$$

By recursive application of Eq. (9), T_k^N can be rewritten in a telescopic series:

$$\begin{aligned} T_k^N &= P_k^N T P_k^N && (18)\\ &= P_k^0 T P_k^0 + \sum_{n=0}^{N-1} (P_k^{n+1} T P_k^{n+1} - P_k^n T P_k^n) && (19)\\ &= P_k^0 T P_k^0 + \sum_{n=0}^{N-1} (P_k^n + Q_k^n) T (P_k^n + Q_k^n) - P_k^n T P_k^n && (20)\\ &= P_k^0 T P_k^0 + \sum_{n=0}^{N-1} Q_k^n T Q_k^n + Q_k^n T P_k^n + P_k^n T Q_k^n && (21)\\ &= T_k^0 + \sum_{n=0}^{N-1} (A_k^n + B_k^n + C_k^n) && (22) \end{aligned}$$

where we have implicitly defined the following components:

$$A_k^n \overset{\text{def}}{=} Q_k^n T Q_k^n \qquad B_k^n \overset{\text{def}}{=} Q_k^n T P_k^n \qquad C_k^n \overset{\text{def}}{=} P_k^n T Q_k^n \tag{23}$$

The full operator can in theory be recovered by taking the limit to infinite refinement and by making use of Eq. (10):

$$T = \lim_{N\to\infty} T_k^N = T_k^0 + \sum_{n=0}^{\infty} (A_k^n + B_k^n + C_k^n) \tag{24}$$

and by truncating the infinite sum we arrive at a multi-resolution operator with finite precision. If we introduce the following auxiliary functions:

$$\begin{aligned} \hat{g}^n &\overset{\text{def}}{=} T_k^n f^n && (25)\\ \tilde{g}^n &\overset{\text{def}}{=} C_k^n \,\mathrm{d}f^n && (26)\\ \mathrm{d}\tilde{g}^n &\overset{\text{def}}{=} (A_k^n + B_k^n)(f^n + \mathrm{d}f^n) && (27) \end{aligned}$$

the operator application can be written

$$\hat{g}^N = T_k^N f^N = T_k^0 f^0 + \sum_{n=0}^{N-1} (A_k^n + B_k^n + C_k^n)(f^n + \mathrm{d}f^n) \tag{28}$$

$$= \hat{g}^0 + \sum_{n=0}^{N-1} \tilde{g}^n + \mathrm{d}\tilde{g}^n \tag{29}$$

In the Non-Standard form the operator is applied one scale at the time, starting from scale zero. In this way the function g can be built adaptively in the same way as for the projection of functions, refining locally based on the wavelet norm at scale n and translation l

$$\| \mathrm{d}\tilde{g}_l^n \| \leq \frac{\epsilon}{2^{n/2}} \|g\| \tag{30}$$

At each scale $\tilde{g}^n$ and $\mathrm{d}\tilde{g}^n$ are computed whereas $\hat{g}$ is only computed at the coarsest scale and reconstructed by recursion at all scales $n > 0$:

$$\hat{g}^n = \hat{g}^{n-1} + \mathrm{d}\tilde{g}^{n-1} + \tilde{g}^{n-1} \qquad n \leq N \tag{31}$$

Eq. (31) shows how the coupling between scales is then achieved by propagating the result from the coarsest to the finest scale. This is done using the filter operations of Eq. (7) in linear complexity.

The main advantage of such a construction is connected to the vanishing moments property of the multiwavelet basis. Apart from the coarsest scale where the full operator is applied, the pure scaling component T_k^n of the operator is never used. The other components contain at least one projection onto the wavelet basis (B_k^n and C_k^n) or two (A_k^n) and it can be shown that these terms decay rapidly with the spatial separation between two grid cells[20;26] and are hence diagonally dominated, opening the way for linear scaling algorithms with reduced communication requirements for parallel implementations.

2.3. *Extension to several dimensions*

The extension to several dimensions can formally be achieved by a standard tensor-product structure (Eq. 8). The main challenge arises from the so called "curse of dimensionality": the storage and computing costs scales exponentially with the dimension d. The tensor product structure of functions and operators makes the computational cost per grid cell scale as Mdk^{d+1} instead of k^{2d}, where M is the separation rank and k is the order of the polynomial basis, which reduces the exponent and complexity significantly, making it feasible to use the described approach up to $d = 3$. For dimensionality higher than 3, it would however be unavoidable to make use of rank reduction techniques as described originally by Beyklin *et al.* [24;27;28] and recently employed by Bischoff and Valeev.[29] It is beyond the scope of the present paper to describe the details of the multi-dimensional implementation which have been presented in detail elsewhere.[25] Suffice to say that the number of components of the Non-Standard form becomes 2^{2d} and all but the pure scaling component are employed in the operator application. What makes it feasible to use such an approach is again the tensor product structure coupled with the vanishing moments which significantly reduce the cost of applying the different components.

3. Implementation

Implementing the mathematical formalism outlined in section 2 in an efficient computer code is a complex task. In grid based, real-space methods the amount of grid data required grows rapidly with the complexity of the function. The computational requirements increase similarly with the number of grid points. For the method to be feasible, it is important to utilize algorithms which reduce the memory footprint, lower the computational demands and have favorable parallel scaling properties. Finding a good balance between these requirements can be challenging. We have addressed these issues by exploiting three major algorithmic ideas: automatic grid adaptation, use of sparsity and parallelization. Our computer code has been implemented in C++, due to the demands for high performance, as well as the complex data structures involved. The code has been written using a fully object-oriented approach, using generic programming and polymorphic classes.

3.1. *Grid adaptation*

In order to drastically reduce the memory footprint we exploit the refinement properties of multiwavelets to automatically generate fully problem adapted grids. One of the major problems encountered in FEM that employs a uniform (Eq. **??**) distribution of grid points is that the overall precision of the representation is limited by the regions where the representation is poor, typically in regions where the function is changing rapidly. With uniform grids large parts of the function will be excessively overrepresented, if good precision is to be achieved. For molecular systems it is well known that the problematic regions are in the vicinity of the nuclear positions, and the problem has previously been addressed by projecting out the spherically symmetric part around each nucleus and treating this separately.[30;31;32] The remaining part of the solution is then treated much more efficiently by a uniform FEM grid. The main issues with this approach are the lack of generality and the complication that arises from the coupling between the different parts. Instead, by expanding the functions in the multi-resolution wavelet basis (Eq. 12), and truncating this expansion locally according to Eq. (13), we achieve grid adaptation that largely solves the problem by locally varying the grid density according to the actual needs. However, much of the code complexity arises from the grid adaptation in a tensorial basis, since the grids cannot be easily stored in large blocks or arrays. Instead the grids are stored in sub-blocks, or `nodes`, whose spatial extensions are dependent on the level of refinement. Each `node` has a fixed number of grid points, typically $10 \times 10 \times 10$ points, and is uniquely addressable by four integers (scale and translation in three dimensions). From an implementation point of view, it is practical to store the grid blocks in a `tree` structure.

3.2. *Sparsity*

In the Non-Standard multiwavelet representation of integral operators the operator matrices become sparse and diagonally dominant. The sparsity implies that the operator has a limited bandwidth at each scale, and is negligible outside the bandwidth. By exploiting the sparsity one avoids calculating trivial zero contributions, in addition to reducing the data access. We will show that by properly exploiting the sparsity, the amount of computation scales linearly with the size of the problem.

Fully exploiting the operator sparsity is rather complex in a general, n-dimensional tensorial basis. In tensorial form the operator has 4^d components where d is the dimensionality of the problem (A, B, C and T of Eq. (24) in each dimension). The structure of the operator is further complicated by the fact that different operator components have different bandwidths (that also changes depending on the length scale), and in the current implementation sparsity is fully utilized by treating all 64 (for $d = 3$) operator components separately.

3.3. *Parallelization*

As pointed out above, any grid based, real-space method is faced with two major problems: The data storage requirements that quickly exceed the available memory, and the computational time that increases accordingly. Thus, calculations rapidly become intractable. Both problems can be addressed simultaneously by parallel processing. By having more processors working on the same problem the computation time can be reduced without additional memory requirements on a given computational device. Furthermore, on cluster type machines with distributed memory, the memory requirements on the individual compute nodes (hosts) can be reduced by only storing parts of the function representation.

In the current version of the code we use two parallelization schemes: Utilizing symmetric multi-processing (SMP) and shared memory using the standard OpenMP paradigm, as well as distributed processing using the Message Passing Interface (MPI) for communication between computational hosts.

3.3.1. *Symmetric multi-processing and OpenMP*

Modern computers have multiple, independent computational cores sharing a common memory space, and using OpenMP technology it does not require much in terms of implementation to efficiently utilize all available processors on a given host. Using multiwavelets, there is by construction no overlap between basis functions located on different grid cells, so each `node` is formally independent of the other `nodes` in the `tree`. Efficient parallelization is achieved by traversing the `tree` structures and defining tasks for each `node`, letting the OpenMP runtime system handle the scheduling. For function representation (projection of a function onto the multiwavelet basis), the tasks are truly independent and the parallelization is simple and highly efficient. Since every OpenMP thread has a fair amount of computation to perform, the parallelization overhead is negligible.

Due to the grid adaptation, parallelization of the operator application is somewhat more complicated. Applying an operator with a non-zero bandwidth, implies accessing neighboring `nodes` that may not be available at the given scale, since the grid refinement might have stopped at a coarser scale for the neighboring part of the function. By applying the wavelet transform (Eq. 7), information about the function at the correct scale can be generated. However, generating the corresponding coefficients is costly, and they often get reused many times. It is wasteful to generate the coefficients every time, and instead they are generated once when needed, and kept until explicitly released. This complicates the parallelization of the operator application, since a generated `node` might be within the bandwidth of two resultant `nodes` being processed simultaneously, causing a data race. By carefully using explicit data locking at the deepest possible location in the code, and by tuning task scheduling, we have been able to achieve very low impact from locking, achieving good parallel performance.

3.3.2. *Distributed processing and MPI*

While SMP reduces the total computational time, it does not reduce the local memory requirements. By using homogeneous clusters of computers, we can reduce both the computational time and the local memory footprint. However, contrary to SMP, distributed processing adds significant complexity to the code.

When dealing with distributed data structures, one must consider not only how the data should be distributed, but also how it should be accessed. In the current version of the code we employ a series of techniques to distribute data in an efficient manner. One of the key elements is a redundant representation of the `tree` structure. Each host has a complete and identical, skeleton `tree` structure. Each `node` in the `tree` has a label identifying the host it belongs to, and only the owner of a `node` has allocated memory for it, and has coefficients, although some redundancy in storage is allowed.

When applying an operator on a distributed `tree` (both input and output functions are distributed), one finds that some of the required data is located on different MPI hosts and needs to be transferred. Because of a non-zero bandwidth of the operator, each `node` of the output function (the potential) will get contributions from several of the surrounding `nodes` of the input function (the charge density), and at the same time each `node` of the input function will give contribution to several surrounding `nodes` of the output function.

In such situations there are two possible communication strategies: one can focus on the data distributions of either the output or the input function. Each MPI host will be allocated a number of `nodes` of the potential to calculate, and one can then choose to fetch all the data needed of the charge distribution to calculate these `nodes`. This requires MPI communication only prior to operator application, but will result in a redundant distribution of the charge density `nodes` (several MPI hosts will have the same data).

On the other hand, all MPI hosts also have a number of allocated `nodes` of the charge density, and one can instead choose to calculate the potential arising from these `nodes` directly. This requires no initial communication step, but will result in an incomplete potential for all MPI hosts, and the result needs to be communicated and added up by the appropriate MPI host afterwards. We find that the latter strategy generally requires less data communication, so this will be our method of choice, although both strategies are implemented.

In both schemes it is equally important to ensure data localization to minimize communication between hosts. When the real-space domains of all MPI hosts are well localized, communication is reduced drastically, and when the number of MPI hosts gets large the limited bandwidth of the operator will ensure that the communication will be limited to only near neighbors. In the current code, good localization is achieved by first building a skeleton `tree` structure without coefficients, which tries to estimate the final `tree` structure as closely as possible. Then the `tree` is traversed through a so-called space filling curve: a path constructed recursively such that its fractal limit coincide with the whole multi-dimensional space. In this way, an ordering of the `nodes` in a `tree` structure is introduced and the data can easily be distributed by partitioning the curve into contiguous chunks. Moreover, the lengths of such portions should ideally reflect the computational work load in subsequent computations.

There are several ways to construct such a space filling curve, but to ensure maximum locality Griebel[33] suggests Hilbert curves. The construction of the Hilbert curve starts at scale $n = 1$, connecting all 2^d `nodes` in a specific sequence. In particular, addressing each `node` using a bit notation (one bit for each direction), the fundamental Hilbert curve will start at a given `node`, and end at one of its adjacent `nodes` (only one bit different from the starting `node`). Each step of the curve can be defined by a bit switch: only one bit of the sequence changes every time. The procedure is continued recursively through all `nodes` of the `tree`, but the sequence through which the children `nodesare` connected is changing in such a way that the curve remains continuous. Figures 1 and 2 show the difference between the

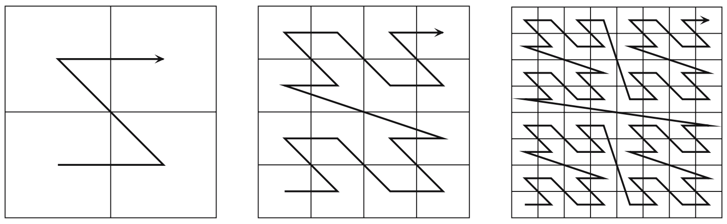

Fig. 1. Three refinement levels in the construction of the Lebesgue curve in 2D.

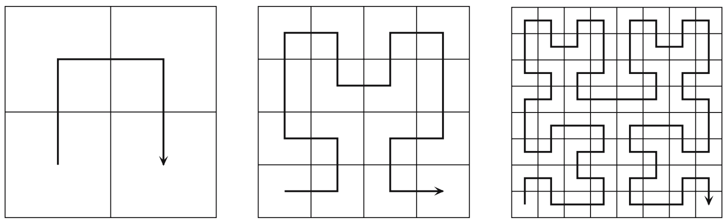

Fig. 2. Three refinement levels in the construction of the Hilbert curve in 2D.

Lebesgue curve, which is induced by the natural ordering of the bit sequences (00, 01, 10, 11) and the Hilbert curve (00, 10, 11, 01) for three refinement levels in two dimensions. The natural bit ordering of the Lebesgue curve implies "jumps" in the sequence (shown by the Z shape) which will ultimately make the curve converge to its space-filling limit point-wise to a discontinuous limiting curve. The Hilbert curve converges uniformly to a continuous space filling curve. As a consequence, employing a Hilbert curve to partition the domain will result in connected domains with good locality properties which will eventually minimize the communication overhead.

Besides the communication overhead, the main concern in distributed processing is to ensure a balanced work load between MPI hosts. This is not a straightforward task as the work needed to calculate a given `node` can vary a lot, depending on the bandwidth (in each dimension, at a specific length scale) of the operator as well as the local norms of both the input and output functions. This means that simply dividing the Hilbert curve into equally sized portions (which is the way the data is initially distributed) does usually not give the best work distribution, and some tuning is needed prior to operator application. This is done by assigning a work load to each `node` in the `tree` and shifting the domain boundary along the Hilbert curve in such a way that the total work gets evenly distributed. In this way the communication is done only between neighboring MPI hosts. However, our experience show that this load balancing algorithm is not always flexible enough, and it requires that the work load is quite well distributed to begin with, and it will probably be insufficient when the number of MPI hosts gets large.

3.4. *Algorithm*

Finally we present in Alg. 1 the hybrid MPI/OpenMP parallel algorithm for the operator application in the Non-Standard form in the adaptive multiwavelet basis. It is important to separate between the input and output functions, as the `tree` building algorithm is driven by the `nodes` of the output function, while the MPI work load distribution is based on the `nodes` of the input function. We will denote the `nodes` that are owned by the given MPI host as *local*, while *all* `nodes` represents the full `tree`, regardless of MPI ownership.

The initial `tree` skeleton for the output function that appears in line 2 of Alg. 1 can be chosen in several ways: Beylkin *et al.*[20] suggests to simply copy the `tree` of the input function, but we find this to be a suboptimal choice for the types of operators we are dealing with, as their ”smoothing” properties in general leads to output grids that are coarser but wider than the grids of the input function. For applications in iterative solution algorithms, which are commonly encountered in computational chemistry, the same functions are computed with minor modifications in each iteration, and a good choice in this case could be to reuse the grid from the previous iteration. We would like to emphasize that the final `tree` structure obtained at the end of the operator application will not depend on the choice of initial `tree` skeleton, as the adaptive algorithm will always build the same `tree` in the end. The initial choice is only important for obtaining a good and efficient data distribution, and becomes particularly important as all further refinement beyond the initial `tree` will happen *locally* on the given MPI host (line 16).

The presented algorithm will build a `tree` structure for the output function starting from the initial empty skeleton, adding non-uniform layers of refinement until the requested accuracy is obtained in terms of the wavelet norm. In each iteration (line 5) *all* new (empty) `nodes` of the output function is traversed by all MPI hosts, but only the `nodes` that get contributions from one or more *local* `nodes` of the input function will actually be calculated. Afterwards the calculated `nodes` that belong to another host is transferred to (and added up by) its rightful owner (line 12). When all contributions to a given output `node` are collected, the owning host will decide whether or not the `node` needs to be further refined (line 15).

The actual computation of the scaling and wavelet coefficients in line 10 of Alg. 1 involves rather complicated combinatorics in multiple dimensions, as the 4^d operator components will couple the 2^d scaling/wavelet components of all input `nodes` within the bandwidth to the 2^d scaling/wavelet components of the output `node`. Details can be found in Frediani *et al.*[25] for the implementation without parallelization.

4. Results

In this section we will demonstrate the performance of our code with various test calculations on realistic molecular systems, specifically linear alkane chains C_nH_{2n+2} for $n = \{2,\ldots,70\}$ and pyramidal diamond fragments $C_{(2n+3)(n+2)(n+1)/6}H_{2(n+2)(n+1)}$ for $n = \{1,\ldots,9\}$. All systems were constructed with a constant C-C bond length of 0.154 nm, and terminal H atoms were attached. The electron densities were precomputed at Density Functional Theory (DFT) level (BLYP[34;35;36] functional, Dunning's DZ[37;38] Gaussian basis) using the LSDalton[39] program and then projected onto the multiwavelet basis. Similar calculations were performed by Watson and Hirao[40] using a spectral-element method with a high-

Algorithm 1 Adaptive operator application in the Non-Standard form

```
 1: MPI: assume input tree distributed among hosts through Hilbert path
 2: create tree skeleton of empty nodes for the output function
 3: MPI: distribute output nodes among hosts through Hilbert path
 4: create list of all output nodes
 5: while number of output nodes in the current list $N > 0$ do
 6:   OpenMP: divide the list of output nodes among available processors
 7:   for each output node in the current list do
 8:     fetch local input nodes within bandwidth of the operator
 9:     prune list of input nodes based on Cauchy-Schwartz screening
10:     compute scaling and wavelet coefficients of output node
11:   end for
12:   MPI: communicate the nodes of the output function to the appropriate host
13:   for each local node of the output function do
14:     remove node from current list
15:     if node needs to be refined then
16:       allocate children nodes (inherits MPI ownership from parent)
17:       add children nodes to the current list
18:     end if
19:   end for
20:   MPI: collect list of all output nodes for the next iteration
21: end while
```

order Chebyshev polynomial basis.

All computations have been performed on a cluster consisting of 2×8 cores Intel Xeon E5-2670 processors with 16 GB memory, connected by an infiniband network.

4.1. *Accuracy*

One attractive property of the multiwavelet basis that separates it from other bases commonly used in quantum chemistry calculations is the strict error control in both function projections and operator applications. By truncating the wavelet expansions *locally* according to Eq. (13) we can control the *global* error of the calculations. We demonstrate this by calculating the Coulomb self-repulsion energy of a charge distribution $\rho(r)$:

$$E = \int \rho(r)V(r)dr \tag{32}$$

where the potential $V(r)$ is obtained by solving the Poisson equation (Eq. 2). There are several critical issues in order to guarantee the precision of the calculated energy: The charge distribution as well as the Poisson operator (e.i. the kernel expansion in Eq. (14)) must be represented with sufficient accuracy, and the operator must be applied in such a way that no significant contributions are omitted, while at the same time thresholding as much as possible for efficiency. A detailed discussion of the accuracy parameters that appear in the operator construction and application is presented in a separate study.[25]

Test calculations were performed on the smaller alkane systems, $n = \{2, 4, 6, 8, 10\}$. The Coulomb energy was calculated analytically in the Gaussian basis by LSDalton, and the accuracy of our Poisson solver is tested against the analytical result. For each system we calculate the energy to three different target accuracies $\epsilon = \{10^{-5}, 10^{-7}, 10^{-9}\}$ and the numbers are presented in table 1. The error in the charge integral gives an indication of the accuracy of the projection of the Gaussian basis onto the multiwavelet basis, and we see that the errors are well within the given threshold. The errors in the Coulomb energy are also consistently below the requested precision and are more or less independent of the size of the system (at least for the more accurate calculations).

The calculations were performed using a polynomial order $k \geq -\log(\epsilon) + 2$ that increases with increasing accuracy. This is a slightly less conservative choice than what we have found to be a safe compromise to guarantee the accuracy of the operator application ($k \geq -1.5\log(\epsilon) + 2$, details can be found in Ref. 25). In principle the increasing accuracy should *not* be a result of the increasing polynomial order, and a fixed, low polynomial order should give similar precision. This however requires that the formally correct thresholding of wavelet coefficient is employed.

With reference to Eq. (13), for a multivariate function the factor $2^{n/2}$ should be exchanged for $2^{dn/2}$ where d is the dimensionality. This choice is however too strict, yielding much larger function representations than necessary. In prctical cases it is more convenient to make use of $d = 1$ in Eq. (13) and at the same time make use of a polynomial order which is large enough to ensure that the requested accuracy ϵ is achieved. A smaller k would not yield the required accuracy, whereas a larger one would increase the size of the representation.

Table 1: Accuracy of Coulomb energy and charge integral of small alkane molecules. Densities are precomputed at DFT (BLYP) level and analytic energies are calculated in the Gaussian basis (Dunning DZ) using the LSDalton program. Computation times are for a single processor.

Requested	Time	Coulomb energy	Relative error	
precision	(sec)	(hartree)	Energy	Charge
		C_2H_6		
ϵ = 1.0e-05	12	80.085849034	3.2e-06	5.9e-07
ϵ = 1.0e-07	41	80.086102499	1.4e-08	2.9e-09
ϵ = 1.0e-09	607	80.086103631	1.3e-10	2.2e-11
Analytic	-	80.086103641	-	-
		C_4H_{10}		
ϵ = 1.0e-05	13	205.548292326	2.0e-06	4.4e-07
ϵ = 1.0e-07	52	205.548692277	1.4e-08	2.4e-09
ϵ = 1.0e-09	774	205.548695185	1.3e-10	1.6e-11
Analytic	-	205.548695213	-	-
		C_6H_{14}		
ϵ = 1.0e-05	16	355.995462175	3.4e-06	4.2e-07
ϵ = 1.0e-07	63	355.996666233	1.4e-08	1.8e-09
ϵ = 1.0e-09	792	355.996671114	1.4e-10	1.3e-11
Analytic	-	355.996671163	-	-
		C_8H_{18}		
ϵ = 1.0e-05	18	523.397878651	4.1e-06	3.8e-07
ϵ = 1.0e-07	73	523.400041990	1.4e-08	1.2e-09
ϵ = 1.0e-09	1035	523.400049203	1.4e-10	9.7e-12
Analytic	-	523.400049277	-	-
		$C_{10}H_{22}$		
ϵ = 1.0e-05	20	703.6228144331	7.1e-06	3.4e-07
ϵ = 1.0e-07	85	703.6277668016	1.5e-08	1.3e-09
ϵ = 1.0e-09	1148	703.6277770551	1.5e-10	8.4e-12
Analytic	-	703.6277771588	-	-

Table 2: Absolute and relative errors in Coulomb energies calculated to requested relative accuracy 10^{-6}, and requested absolute accuracy 10^{-3}. Densities are precomputed at DFT (BLYP) level and analytic energies are calculated in the Gaussian basis (Dunning DZ) using the LSDalton program. Computation times are for a single processor.

Alkane system n in C_nH_{2n+2}

		Req. rel. precision 1.0e-6			Req. abs. precision 1.0e-3		
n	Coulomb energy	Time	Error		Time	Error	
	(hartree)	(sec)	Relative	Absolute	(sec)	Relative	Absolute
2	80.08610364	24.9	1.0e-07	8.3e-06	7.7	2.7e-06	2.1e-04
10	703.62777716	42.8	1.1e-07	8.1e-05	30.9	2.4e-07	1.7e-04
20	1752.56975975	62.6	1.7e-07	3.0e-04	72.0	1.0e-07	1.8e-04
30	2936.25648588	84.0	4.4e-07	1.3e-03	110.8	2.0e-07	6.0e-04
40	4211.56647408	101.6	4.9e-07	2.1e-03	151.6	1.7e-07	7.3e-04
50	5555.20166293	116.3	5.5e-07	3.0e-03	190.4	1.6e-07	8.9e-04
60	6953.41087253	130.9	6.9e-07	4.8e-03	225.9	6.6e-08	4.6e-04
70	8396.93995214	143.0	8.8e-07	7.5e-03	259.0	6.4e-08	5.3e-04

4.2. *Linear scaling*

A direct consequence of the banded structure of the operator matrix is that the computation time of applying the operator should scale linearly with system size. We test this property by calculating the Coulomb energy on the full range of alkane systems, $n = \{2, \ldots, 70\}$. By making use of the Fast Multipole Method and adaptive resolution (adaptive in polynomial order, not cell size) Watson and Hirao[40] were able to achieve linear scaling of the calculation of the Coulomb energy for both the alkane and the diamond systems, with a relative error of around 1 ppm.

In trying to reproduce these calculations we choose a 9th order polynomial basis, with a target relative accuracy of $\epsilon = 10^{-6}$, and the numbers are presented in table 2. We see that although the relative accuracy is somewhat degrading as the system size increase, all numbers are within the requested precision. The computation times are plotted to the left in figure 3 and we see that the scaling is in fact

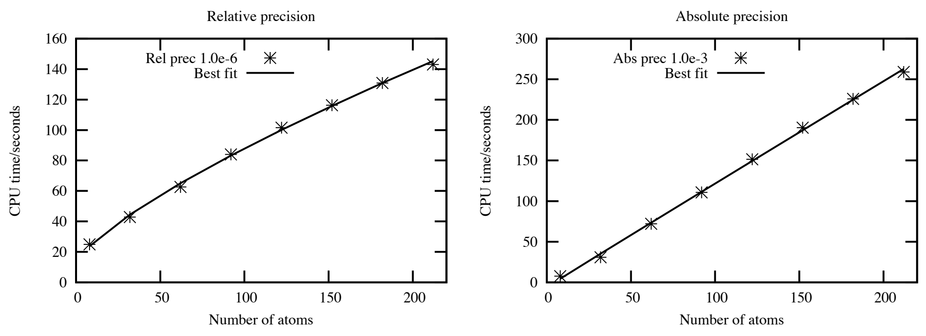

Fig. 3. Scaling of computation time on a single processor for linear alkane systems when requested accuracy is relative (left) and absolute (right). Best fit curves are Eqs. (33) and (34), respectively.

sublinear. The solid line in the plot is obtained using the Levenberg-Marquardt [41;42] algorithm to fit the timings t with respect to the number of atoms n to the function

$$t(n) = 12.5 + 2.34n^{0.754} \tag{33}$$

which shows an exponent significantly lower than one. This behavior can be understood by the use of relative rather than absolute precision in combination with the automatic adaptivity of the grids in our calculations. As the system size increases the global norm of the functions involved increase accordingly, and the local truncation criterion in Eq. (13) is gradually relaxed. This means that the local resolution of the grid around each CH_2 fragment is gradually getting coarser, while the overall *relative* accuracy of the function is maintained. As the computation time is expected to be directly related to the overall number of grid cells, one can expect the scaling to less than linear as the system size increases.

If we on the other hand are interested in an absolute accuracy, we need to maintain the local high resolution around each fragment as the system size is increased. In this case the number of grid cells should grow linearly with the size of the system, and one can expect the computation time to do the same. This would correspond to the calculations done by Watson and Hirao, [40] as their truncation is based on the local rather than global norm of the function and the grid of the full system is more or less the union of the grids around its constituent atoms.

Even though our code is based on relative precision, we can simulate absolute precision by gradually increasing the relative accuracy as the system size grows in such a way that the absolute error remains approximately constant. We present in table 2 also the numbers for such calculations, where the absolute error is kept below 10^{-3}, and the computation times are shown to the right of figure 3. We see that the slope of this curve is steeper than the corresponding curve with relative precision, as the absolute error criterion progressively becomes more demanding than the relative one. The Levenberg-Marquardt algorithm gives a best fit

$$t(n) = -6.0 + 1.33n^{0.991} \tag{34}$$

where the exponent now is very close to one. Comparing with Watson and Hirao, we see a factor of 8-10 reduction of the computation time while the accuracy is about an order of magnitude higher.[a] However, it should be pointed out that their objective was only to demonstrate the linear scaling behavior of their method, and their parameters were not optimized for computational speed.

If one is satisfied with sticking only to a relative accuracy, the calculations quickly becomes much more favorable, and for the biggest alkane system $C_{70}H_{142}$ (562 electrons) the calculation is more than 15 times faster than Watson and Hirao with comparable accuracy.

4.3. *Parallel performance*

We test the parallel performance of our code by calculating the electrostatic potential of the pyramid shaped diamond fragments $C_{(2n+3)(n+2)(n+1)/6}H_{2(n+2)(n+1)}$ for $n = \{1, \ldots, 9\}$, where the biggest system contains more than 600 atoms. The calculations were done using a 9th order polynomial basis with a requested relative

[a]Timings are not adjusted for differences in hardware performance. Watson and Hirao ran on IBM Power 5+ 2.1GHz processors. Although the architectures are not identical, they are to a large extent comparable in performance.

Table 3: Wall clock computation time in seconds for parallel calculation of electronic potential of diamond fragments using pure OpenMP, pure MPI and hybrid MPI/OpenMP strategies. Densities are precomputed at DFT (BLYP) level using the LSDalton program.

Number of CPUs			Diamond system n in $C_{(2n+3)(n+2)(n+1)/6}H_{2(n+2)(n+1)}$								
MPI	OMP	TOT	1	2	3	4	5	6	7	8	9
1	1	1	29.4	46.2	64.7	97.8	133.9	166.8	213.4	269.4	332.0
1	2	2	15.5	25.5	33.0	51.3	69.5	87.2	110.0	138.9	163.4
1	4	4	8.0	12.8	17.4	27.0	36.1	47.3	57.8	73.7	90.1
1	8	8	4.2	6.5	8.8	13.9	18.6	23.5	29.4	36.7	45.0
1	16	16	2.4	3.5	4.7	7.5	9.6	12.2	15.4	19.1	23.3
2	1	2	17.9	28.4	35.8	59.2	81.6	93.0	121.7	147.5	176.5
4	1	4	10.1	16.3	21.2	32.5	51.6	53.8	60.8	85.3	100.7
8	1	8	6.1	9.4	11.8	19.9	25.6	28.0	35.4	43.7	50.7
16	1	16	5.0	6.2	8.2	12.4	15.0	17.3	23.0	26.1	31.3
32	1	32	3.6	4.2	5.4	9.5	9.2	10.9	14.1	16.8	20.0
64	1	64	3.1	3.9	4.3	6.5	6.6	7.4	10.3	11.4	13.4
128	1	128	5.2	5.7	6.4	7.6	7.8	8.8	10.9	10.9	11.9
2	16	32	1.5	2.2	2.9	4.8	6.0	7.0	9.6	11.4	13.7
4	16	64	1.1	1.6	2.0	3.2	4.3	4.7	6.3	7.3	7.9
8	16	128	1.0	1.4	1.8	2.9	3.3	3.8	4.8	5.9	6.4
16	16	256	0.9	1.3	1.6	2.2	2.9	3.3	4.2	4.9	6.0
32	16	512	1.1	1.3	1.5	2.0	2.4	2.8	3.4	4.0	4.8

accuracy of $\epsilon = 10^{-6}$.

Figure 4 shows to the left the walltime of the diamond calculations with 1, 2, 4, 8 and 16 shared memory processors. There are several things to note about this figure. Firstly, we see that the scaling with respect to system size of the single processor calculation is again sublinear. The Levenberg-Marquardt algorithm gives

$$t(n) = 11.6 + 1.84n^{0.805} \tag{35}$$

which is quite close to what we had for the linear alkane systems (up to the biggest alkane system of 200 atoms the difference is less than 2%). This indicates that the

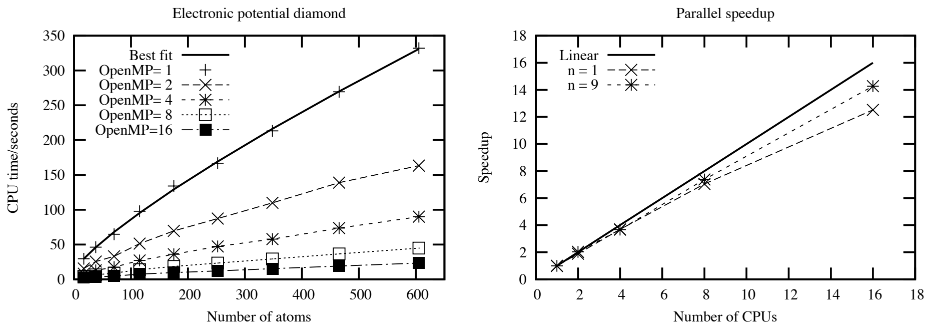

Fig. 4. Left: Scaling of computation time for the calculation of the electronic potential of diamond fragments with number of processors $1, \ldots, 16$ using OpenMP. Numbers are taken from table 3. Best fit curve is Eq. (35). Right: Parallel efficiency of the operator application for diamond systems 1 and 9.

computation time depends solely on the size of the system, not its geometry.

Secondly, we can see a reduction of computation time by roughly a factor of two when the number of processors is doubled, while the scaling properties with respect to the number of atoms is kept also for the parallel computations. To the right of figure 4 we see the parallel speedup for the smallest and the biggest system, which shows an efficiency of 80 and 90 %, respectively, on 16 processors. It is worth noticing that for the biggest system the performance does not fall off significantly up to 16 processors, and it would be interesting to test the code on computers with even more shared memory processors available.

Even though the shared memory parallelization shows very good performance there are still reasons for utilizing distributed memory (MPI) parallelization techniques. Firstly, the memory requirements for representing three-dimensional functions in the multiwavelet basis is still rather big, even with automatic grid adaptation, and the biggest diamond calculation presented in figure 4 more or less exhausts the available resources (16 GB). By adding another layer of parallelization the functions can be distributed over the memory of several MPI hosts, and we can reach even bigger systems. It is expected that the parallelization overhead is more substantial for the MPI implementation, but with an efficient OpenMP implementation there are prospects of efficiently utilizing thousands of processors in a single calculation using a hybrid MPI/OpenMP strategy.

Table 3 shows the computation time for the diamond fragments using different numbers of processors, both in pure OpenMP and pure MPI, as well as using a hybrid MPI/OpenMP strategy, and we see that the wall clock computation time for the biggest diamond system $C_{385}H_{220}$ (2530 electrons) can be pushed down to 5 seconds (which is almost 1000 times faster than Watson and Hirao's single processor calculations) if one throws enough processors on the problem. However, the parallel efficiency in this case is less than 20%, and we would need even bigger systems to fully utilize the capacity of modern parallel computer clusters.

The effect of the Hilbert curve partition in the calculations presented above is not substantial, but it is expected to become more important as the system size increases. We do observe a slight decrease of post operator communication using the Hilbert curve for up to 128 MPI hosts, but the work load distribution of the Hilbert vs. Lebesgue curve is very different, and the overall effect is quite ambiguous. It seems difficult to obtain a strong scaling and reduce the wall clock computation time to less than a few seconds, which means that bigger systems are needed in order to efficiently utilize thousands of MPI processes, where the Hilbert curve is expected to really make a difference, but for such calculations there are other bottlenecks like work load estimation and balancing, and the current implementation cannot efficiently balance more than a few hundred MPI processes. However, the hybrid implementation is rather efficient and a thousand processors is then still within reach, which is satisfactory for the current applications of the code.

5. Conclusions

We have shown that by making use of the properties of multi-resolution analysis and the multiwavelet basis the electrostatic potential arising from arbitrary charge distributions can be calculated efficiently and with guaranteed precision. Test calculations on linear alkane chains demonstrates the inherent linear scaling of the application of integral operators in the multiwavelet formalism. In fact, by virtue of the automatic grid adaptation in our implementation, the scaling becomes sublinear if only a relative accuracy is maintained, as the local accuracy criterion gradually becomes more relaxed as the norm of the function increases. This was demonstrated both for linear alkanes and pyramidal diamond fragments, which showed similar scaling behavior, indicating that this property is independent of the geometry of the system.

The code has been successfully parallelized using both a shared memory (OpenMP) and a distributed memory (MPI) strategy, as well as a combination of these, and it is shown that a hybrid MPI/OpenMP strategy is preferable for a given number of processors. We show a significant improvement in computation time compared to previously reported numbers: an order of magnitude for single-processor calculations, and three orders of magnitude reduction in wall time if parallelization is added.

6. Acknowledgments

This work has been supported by the Research Council of Norway through a Centre of Excellence Grant (Grant No. 179568/V30) and from the Norwegian Supercomputing Program (NOTUR) through a grant of computer time (Grant No. NN4654K).

References

1. Jackson J. D. *Classical Electrodynamics.* Wiley, 1998.
2. Benighaus T. and Thiel W. Efficiency and accuracy of the generalized solvent boundary potential for hybrid qm/mm simulations: Implementation for semiempirical hamiltonians. *J. Chem. Theory Comput.*, 4(10):1600–1609, Oct 2008.
3. Tomasi J., Mennucci B., and Cammi R. Quantum mechanical continuum solvation models. *Chem. Rev.*, 105(8):2999–3093, Jan 2005.
4. Zhou H. X. Boundary element solution of macromolecular electrostatics: interaction energy between two proteins. *Biophysical Journal*, 65(2):955–963, Aug 2005.
5. Liang J. and Subramaniam S. Computation of molecular electrostatics with boundary element methods. *Biophysical Journal*, 73(4):1830–1841, Jul 2005.
6. Tomasi J. and Persico M. Molecular interactions in solution: An overview of methods based on continuous distributions of the solvent. *Chem. Rev.*, 94(7):2027–2094, 1994.
7. Rokhlin V. Rapid solution of integral equations of classical potential theory. *J. Comput. Physics*, 60(2):187–207, 1985.
8. Greengard L. and Rokhlin V. A fast algorithm for particle simulations. *J. Comput. Physics*, 73(2):325–348, 1987.
9. White C. A. and Head-Gordon M. Derivation and efficient implementation of the fast multipole method. *J. Chem. Phys.*, 101(8):6593–6605, 1994.
10. Choi C., Ruedenberg K., and Gordon M. S. New parallel optimal-parameter fast multipole method (opfmm). *J. Comput. Chemistry*, 22(13):1484–1501, 2001.
11. Losilla S. A., D Sundholm D., and Jusélius J. The direct approach to gravitation and electrostatics method for periodic systems. *J. Chem. Phys.*, 132(2):024102, Jan 2010.
12. Jusélius J. and Sundholm D. Parallel implementation of a direct method for

calculating electrostatic potentials. *J. Chem. Phys.*, 126(9):094101, Jan 2007.

13. Berger R. J. F. and Sundholm D. A non-iterative numerical solver of poisson and helmholtz equations using high-order finite-element functions. *Advances in Quantum Chemistry*, 50:235–247, Jan 2005.
14. Beck T. L. Real-space mesh techniques in density-functional theory. *Reviews of Modern Physics*, 72(4):1041–1080, 2000.
15. Bylaska E. J., Holst M., and Weare J. H. Adaptive finite element method for solving the exact kohn-sham equation of density functional theory. *J. Chem. Theory Comput.*, 5(4):937–948, Jan 2009.
16. Chen L., Holst M., and Xu J. The finite element approximation of the nonlinear poisson-boltzmann equation. *Siam J. Numer. Anal*, 45(6):2298–2320, 2007.
17. Holst M., Baker N., and Wang F. Adaptive multilevel finite element solution of the poisson-boltzmann equation i. algorithms and examples. *J. Comput. Chemistry*, 21(15):1319–1342, 2000.
18. Genovese L., Deutsch T., Neelov A., Goedecker S., and Beylkin G. Efficient solution of poisson's equation with free boundary conditions. *J. Chem. Phys.*, 125:074105, 2006.
19. Alpert B. K., Beylkin G., Gines D., and Vozovoi L. Adaptive solution of partial differential equations in multiwavelet bases. *J. Comput. Physics*, 182(1):149–190, 2002.
20. Beylkin G., Cheruvu V., and Pérez F. Fast adaptive algorithms in the non-standard form for multidimensional problems. *Appl. and Comput. Harmonic Analysis*, 24(3):354–377, 2008.
21. Gines D., Beylkin G., and Dunn J. Lu factorization of non-standard forms and direct multiresolution solvers* 1. *Appl. and Comput. Harmonic Analysis*, 5(2):156–201, 1998.
22. Keinert F. *Wavelets and Multiwavelets*. Studies in advanced mathematics. Chapman and Hall/CRC, 2004.

23. Alpert B. K. A class of bases in l2 for the sparse representation of integral operators. *Siam J. Math. Anal.*, 24:246, 1993.
24. Beylkin G. and Monzón L. On approximation of functions by exponential sums. *Appl. and Comput. Harmonic Analysis*, 19(1):17–48, 2005.
25. Frediani L., Fossgaard E., Flå, and 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. *Molecular Physics*, 111(9-11):1143–1160, 2013.
26. Beylkin G., Coifman R., and Rokhlin V. Fast wavelet transforms and numerical algorithms i. *Comm. Pure Appl. Math.*, 44(2):141–183, 1991.
27. Beylkin G. and Mohlenkamp M. J. Algorithms for numerical analysis in high dimensions. *SIAM Journal on Scientific Computing*, 26(6):2133–2159, 2005.
28. Beylkin G. and Mohlenkamp M. J. Numerical operator calculus in higher dimensions. *Proceedings of the National Academy of Sciences*, 99(16):10246, 2002.
29. Bischoff F. A. and Valeev E. F. Low-order tensor approximations for electronic wave functions: Hartree–fock method with guaranteed precision. *J. Chem. Phys.*, 134:104104, 2011.
30. Kurashige Y., Nakajima T., and Hirao K. Gaussian and finite-element coulomb method for the fast evaluation of coulomb integrals. *J. Chem. Phys.*, 126:144106, 2007.
31. Watson M. A., Kurashige Y., Nakajima T., and Hirao K. Linear-scaling multipole-accelerated gaussian and finite-element coulomb method. *J. Chem. Phys.*, 128:054105, 2008.
32. Losilla S. A. and Sundholm D. A divide and conquer real-space approach for all-electron molecular electrostatic potentials and interaction energies. *J. Chem. Phys.*, 136:214104, 2012.
33. Griebel M., Zumbusch G., and Knapek S. *Tree algorithms for long-range potentials*, volume 5 of *Texts in Computational Science and Engineering.* Springer Berlin Heidelberg, 2007.

34. Becke A. D. Density-functional exchange-energy approximation with correct asymptotic behavior. *Phys. Rev. A*, 38:3098–3100, Sep 1988.
35. Lee C., Yang W., and Parr R. G. Development of the colle-salvetti correlation-energy formula into a functional of the electron density. *Phys. Rev. B*, 37:785–789, Jan 1988.
36. Johnson B. G., Gill P. M. W., and Pople J. A. The performance of a family of density functional methods. *J. Chem. Phys.*, 98(7):5612–5626, 1993.
37. Dunning Jr. T. H. Gaussian basis functions for use in molecular calculations. i. contraction of (9s5p) atomic basis sets for the first-row atoms. *J. Chem. Phys.*, 53(7):2823–2833, 1970.
38. Dunning Jr. T. H. Gaussian basis sets for the atoms gallium through krypton. *J. Chem. Phys.*, 66(3):1382–1383, 1977.
39. Lsdalton, a linear scaling molecular electronic structure program, release dalton 2011, see http://daltonprogram.org/.
40. Watson M. A. and Hirao K. A linear-scaling spectral-element method for computing electrostatic potentials. *J. Chem. Phys.*, 129(18):184107, Jan 2008.
41. Levenberg K. A method for the solution of certain nonlinear problems in least squares. *Q. Appl. Math*, 2(164), 1944.
42. Marquardt D. W. An algorithm for least squares estimation of nonlinear parameters. *J. Soc. Ind. Appl. Math*, 11(431), 1963.