Fast adaptive algorithms in the non-standard form for multidimensional problems ☆

Gregory Beylkin *, Vani Cheruvu, Fernando Pérez

Department of Applied Mathematics, University of Colorado, Boulder, CO 80309-0526, USA

Abstract

We present a fast, adaptive multiresolution algorithm for applying integral operators with a wide class of radially symmetric kernels in dimensions one, two and three. This algorithm is made efficient by the use of separated representations of the kernel. We discuss operators of the class $(-\Delta + \mu^2 I)^{-\alpha}$, where $\mu \geqslant 0$ and $0 < \alpha < 3/2$, and illustrate the algorithm for the Poisson and Schrödinger equations in dimension three. The same algorithm may be used for all operators with radially symmetric kernels approximated as a weighted sum of Gaussians, making it applicable across multiple fields by reusing a single implementation. This fast algorithm provides controllable accuracy at a reasonable cost, comparable to that of the Fast Multipole Method (FMM). It differs from the FMM by the type of approximation used to represent kernels and has the advantage of being easily extendable to higher dimensions.

© 2007 Elsevier Inc. All rights reserved.

Keywords: Separated representation; Multiwavelets; Adaptive algorithms; Integral operators; Fast Multipole Method

1. Introduction

For a number of years, the Fast Multipole Method (FMM) [1–3] has been the method of choice for applying integral operators to functions in dimensions $d \leqslant 3$. On the other hand, multiresolution algorithms in wavelet and multiwavelet bases introduced in [4] for the same purpose were not efficient enough to be practical in more than one dimension. Recently, with the introduction of separated representations [5–7], practical multiresolution algorithms in higher dimensions [8–11] became available as well. In this paper we present a new fast, adaptive algorithm for applying a class of integral operators with radial kernels in dimensions $d = 1, 2, 3$, and we briefly discuss its extension to higher dimensions.

In physics, chemistry and other applied fields, many important problems may be formulated using integral equations, typically involving Green's functions as their kernels. Often such formulations are preferable to those via partial

---

* Corresponding author.

E-mail address: beylkin@colorado.edu (G. Beylkin).

differential equations (PDEs). For example, evaluating the integral expressing the solution of the Poisson equation in free space (the convolution of the Green's function with the mass or charge density) avoids issues associated with the high condition number of a PDE formulation. Integral operators appear in fields as diverse as electrostatics, quantum chemistry, fluid dynamics and geodesy; in all such applications fast and accurate methods for evaluating operators on functions are needed.

The FMM and our approach both employ approximate representations of operators to yield fast algorithms. The main difference lies in the type of approximations that are used. For example, for the Poisson kernel $1/r$ in dimension $d = 3$, the FMM [3] uses a plane wave approximation starting from the integral

$$\frac{1}{r} = \frac{1}{2\pi} \int\limits_0^\infty e^{-\lambda(z-z_0)} \int\limits_0^{2\pi} e^{i\lambda((x-x_0)\cos\alpha+(y-y_0)\sin\alpha)} \, d\alpha \, d\lambda, \tag{1}$$

where $r = \sqrt{(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2}$. The elegant approximation derived from this integral in [3] is valid within a solid angle, and thus requires splitting the application of an operator into directional regions; the number of such regions grows exponentially with dimension. For the same kernel, our approach starts with the integral

$$\frac{1}{r} = \frac{2}{\sqrt{\pi}} \int\limits_{-\infty}^\infty e^{-r^2 e^{2s}+s} \, ds, \tag{2}$$

and its discretization with finite accuracy ϵ yields a spherically symmetric approximation as a weighted sum of Gaussians. Other radial kernels can be similarly treated by a suitable choice of integrals. The result is a separated representation of kernels and, therefore, an immediate reduction in the cost of their application. This difference in the choice of approximation dictates the differences in the corresponding algorithms. In dimension $d \leqslant 3$ both approaches are practical and yield comparable performance. The key advantage of our approach is its straightforward extensibility to higher dimensions [6,7].

Given an arbitrary accuracy ϵ, we effectively represent kernels by a set of exponents and weights describing the terms of the Gaussian approximation of integrals like that in (2). The number of terms in such sum is roughly proportional to $\log(\epsilon^{-1})$, or a low power of $\log(\epsilon^{-1})$, depending on the operator (see examples in [23]). Since operators are fully described up to an accuracy ϵ by the exponents and weights of the sum of Gaussians, a single algorithm applies all such operators. These include operators such as $(-\Delta + \mu^2 I)^{-\alpha}$, where $\mu \geqslant 0$ and $0 < \alpha < 3/2$, and certain singular operators such as the projector on divergence-free functions. Since many physically significant operators depend only on the distance between interacting objects, our approach is directly applicable to problems involving a wide class of operators with radial kernels.

We combine separated and multiresolution representations of kernels and use multiwavelet bases [12] that provide *inter alia* a method for discretizing integral equations, as is the case in quantum chemistry [8–11]. This choice of multiresolution bases accommodates integral and differential operators as well as a wide variety of boundary conditions, without degrading the order of the method [13,14]. Multiwavelet bases retain the key desired properties of wavelet bases, such as vanishing moments, orthogonality, and compact support. Due to the vanishing moments, wide classes of integro–differential operators have an *effectively sparse* matrix representation, i.e., they differ from a sparse matrix by an operator with small norm. Some of the basis functions of multiwavelet bases are discontinuous, similar to those of the Haar basis and in contrast to wavelets with regularity (see, e.g., [15,16]). The usual choices of scaling functions for multiwavelet bases are either the scaled Legendre or interpolating polynomials. Since these are also used in the discontinuous Galerkin and discontinuous spectral elements methods, our approach may also be seen as an adaptive extension of these methods.

The algorithm for applying an operator to a function starts with computing its adaptive representation in a multiwavelet basis, resulting in a 2^d-tree with blocks of coefficients at the leaves. Then the algorithm adaptively applies the (modified) separated non-standard form [4] of the operator to the function by using only the necessary blocks as dictated by the function's tree representation. We note that in higher dimensions, $d \gg 3$, functions need to be in a separated representation as well, since the usual constructions via bases or grids are prohibitive (see [6,7]).

We start in Section 2 by recalling the basic notions of multiresolution analysis, non-standard operator form and adaptive representation of functions underlying our development. We then consider the separated representation for radially symmetric kernels in Section 3, and use it to efficiently extend the modified ns-form to multiple spatial

dimensions in Section 4. We pay particular attention to computing the band structure of the operator based on one-dimensional information. We use this construction in Section 5 to introduce the adaptive algorithm for application of multidimensional operators in the modified ns-form, and illustrate its performance in Section 6. We consider two examples: the Poisson equation in free space and the ground state of the Hydrogen atom. We conclude with a brief discussion in Section 7.

2. Preliminary considerations

This section and Appendix A are provided for the convenience of the reader in order to keep this paper reasonably self-contained. We provide background material and introduce necessary notation.

The essence of our approach is to decompose the operator using projectors on a multiresolution analysis (MRA), and to efficiently apply its projections using a separated representation. We use the decomposition of the operator into the ns-form [4], but we organize it differently (thus, *modified* ns-form) to achieve greater efficiency. This modification becomes important as we extend this algorithm to higher dimensions.

In this section we introduce notation for MRA, describe the adaptive representation of functions and associated data structures, introduce the modified ns-form and an algorithm for its adaptive application in dimension $d = 1$ as background material for the multidimensional case.

2.1. Multiresolution analysis

Let us consider the multiresolution analysis as a decomposition of $L^2([0, 1]^d)$ into a chain of subspaces

$$\mathbf{V}_0 \subset \mathbf{V}_1 \subset \mathbf{V}_2 \subset \cdots \subset \mathbf{V}_n \subset \cdots,$$

so that $L^2([0, 1]^d) = \overline{\bigcup_{j=0}^{\infty} \mathbf{V}_j}$. We note that our indexing of subspaces (increasing towards finer scales) follows that in [13], and is the reverse of that in [4,15]. On each subspace $\mathbf{V}_j$, we use the tensor product basis of scaling functions obtained using the functions $\phi_{kl}^j(x)$ ($k = 0, \ldots, p - 1$) which we briefly describe in Appendix A.

The wavelet subspaces $\mathbf{W}_j$ are defined as the orthogonal complements of $\mathbf{V}_j$ in $\mathbf{V}_{j-1}$, thus

$$\mathbf{V}_n = \mathbf{V}_0 \bigoplus_{j=0}^{n} \mathbf{W}_j.$$

Introducing the orthogonal projector on $\mathbf{V}_j$, $\mathbf{P}_j : L^2([0, 1]^d) \to \mathbf{V}_j$ and considering an operator $\mathbf{T} : L^2([0, 1]^d) \to L^2([0, 1]^d)$, we define its projection $\mathbf{T}_j : \mathbf{V}_j \to \mathbf{V}_j$ as $\mathbf{T}_j = \mathbf{P}_j \mathbf{T} \mathbf{P}_j$. We also consider the orthogonal projector $\mathbf{Q}_j : L^2([0, 1]^d) \to \mathbf{W}_j$, defined as $\mathbf{Q}_j = \mathbf{P}_{j+1} - \mathbf{P}_j$.

2.2. Adaptive representation of functions

Let us describe an adaptive refinement strategy for construction multiresolution representations of functions $f : B \to B$, where $B = [0, 1]^d$. We proceed by recursive binary subdivision of the box B, so the basic structure representing our functions is a 2^d-tree with arrays of coefficients stored at the leaves (terminal nodes) and no data stored on internal nodes. On each box obtained via this subdivision, our basis is a tensor product of orthogonal polynomials of degree $k = 0, \ldots, p - 1$ in each variable, as described in Appendix A. Therefore, the leaves carry d-dimensional arrays of p^d coefficients which may be used to approximate function values anywhere in the box corresponding to the spatial region covered by it, via (A.4) or its equivalent for higher values of d. For conciseness, we will often refer to these d-dimensional arrays of coefficients stored at tree nodes as *function blocks*.

This adaptive function decomposition algorithm is similar to that used in [17]. Such construction formally works in any dimension d. However, since its complexity scales exponentially with d, its practical use is restricted to fairly low dimension, e.g. $d \lesssim 4$. In higher dimensions, alternate representation strategies for functions such as [6,7] should be considered. In high dimensions, these strategies deal with the exponential growth of complexity by using controlled approximations that have linear cost in d.

For simplicity, we will describe the procedure for the one-dimensional case since the extension to dimensions $d = 2, 3$ is straightforward. Since we cannot afford to construct our representation by starting from a fine scale (especially

in $d = 3$), we proceed by successive refinements of an initial coarse sampling. This approach may result in a situation where the initial sampling is insufficient to resolve a rapid change in a small volume; however, in practical applications such situations are rare and may be avoided by an appropriate choice of the initial sampling scale.

Let $B_l^j = [2^{-j}l, 2^{-j}(l+1)]$, $l = 0, \ldots, 2^j - 1$, represent a binary subinterval on scale j. We denote by $f_l^j = \{f_l^j(x_k)\}_{k=0}^{p-1}$ the vector of values of the function f on the Gaussian nodes in B_l^j. From these values we compute the coefficients $\{s_{kl}^j\}_{k=0}^{p-1}$ (see (A.6) in Appendix A) and interpolate $f(x)$ for any $x \in B_l^j$ by using (A.4). We then subdivide B_l^j into two child intervals, B_{2l}^{j+1} and B_{2l+1}^{j+1}, and evaluate the function f on the Gaussian nodes in B_{2l}^{j+1} and B_{2l+1}^{j+1}. We then interpolate f by using the coefficients $\{s_{kl}^j\}_{k=0}^{p-1}$ from their parent interval and denote by $\tilde{f}_{2l}^{j+1}$ and $\tilde{f}_{2l+1}^{j+1}$ the vectors of interpolated values on the two subintervals. Now, if for a given tolerance ϵ either $\|f_{2l}^{j+1} - \tilde{f}_{2l}^{j+1}\| > \epsilon$ or $\|f_{2l+1}^{j+1} - \tilde{f}_{2l+1}^{j+1}\| > \epsilon$, we repeat the process recursively for both subintervals, B_{2l}^{j+1} and B_{2l+1}^{j+1}; otherwise, we keep the coefficients $\{s_{kl}^j\}_{k=0}^{p-1}$ to represent the function on the entire interval B_l^j. This interval then becomes a leaf in our tree.

At this stage we use the ℓ^∞ norm, thus constructing an approximation $\tilde{f}$ to the original function f such that $\|f - \tilde{f}\|_\infty < \epsilon$, which immediately implies that $\|f - \tilde{f}\|_2 < \epsilon$. This estimate clearly extends to any dimension. Once the approximation with ℓ^∞ norm is constructed, the corresponding tree may be pruned if an application only requires the approximation to be valid in the ℓ^2 norm. We start this process on the finest scale and simply remove all blocks whose cumulative contribution is below ϵ. Other norms, such as H_1, can be accommodated by appropriately weighing the error tolerance with a scale-dependent factor in the initial (coarse to fine) decomposition process.

The complete decomposition algorithm proceeds by following the above recipe, starting with an initial coarse scale (typically $j = 0$) and continuing recursively until the stopping criterion is met for all subintervals. In practice, we choose a stopping scale $j_{\max}$, beyond which the algorithm will not attempt to subdivide any further. Reaching $j_{\max}$ means that the function has significant variations which are not accurately resolved over an interval of width $2^{-j_{\max}}$ using a basis of order p. A pseudo-code listing of this process is presented as Algorithm 1.

Algorithm 1. Adaptive function decomposition.

Start at a coarse scale, typically $j = 0$.
Recursively, for all boxes b^j on scale j, proceed as follows:
Construct the list C of 2^d child boxes b^{j+1} on scale $j + 1$.
Compute the values of the function $f(b^j)$ at the p^d Gauss–Legendre quadrature nodes in b^j.
for all boxes $b^{j+1} \in C$ **do**
    From $f(b^j)$, interpolate to the Gauss–Legendre quadrature nodes in b^{j+1}, producing values $\tilde{f}(b^{j+1})$.
    Compute the values of $f(b^{j+1})$ at the Gauss–Legendre nodes of b^{j+1}, by direct evaluation.
    if $\|f(b^{j+1}) - \tilde{f}(b^{j+1})\|_\infty > \epsilon$ **then**
        Recursively repeat the entire process for all boxes $b^{j+1} \in C$.
    end if
end for
Getting here means that the interpolation from the parent was successful for all child boxes. We store the parent's coefficients from b^j in the function tree.

2.2.1. Tree structures for representing functions

The decomposition Algorithm 1 naturally leads to a tree data structure to represent functions, with the leaves of the tree corresponding to the spatial intervals over which the multiwavelet basis provides a sufficiently accurate local approximation. By using (A.4) or its higher-dimensional extensions, the only data needed to approximate $f(x)$ anywhere in B is the array of basis coefficients on these leaves. Thus we use a tree structure where the leaves store these coefficients and the internal nodes do not contain any data (and are effectively removed since we use hash tables for storage). We will refer to this structure as an *adaptive tree*. Each level in the tree corresponds to a scale in the MRA, with the root node corresponding to the coarsest projection $f_0 \in \mathbf{V}_0$.

Now, in order to apply the modified non-standard form of an operator to a function, we will show in the next section that we also need the basis coefficients corresponding to internal nodes of the tree. Hence, from the adaptive tree data

(a) Function

(b) Pointwise error

$N_{nod} = 8, \epsilon = 1.0e - 04, N_{blocks} = 17$

$N_{nod} = 8, \epsilon = 1.0e - 04, N_{blocks} = 33$

(c) Adaptive tree

(d) Redundant tree
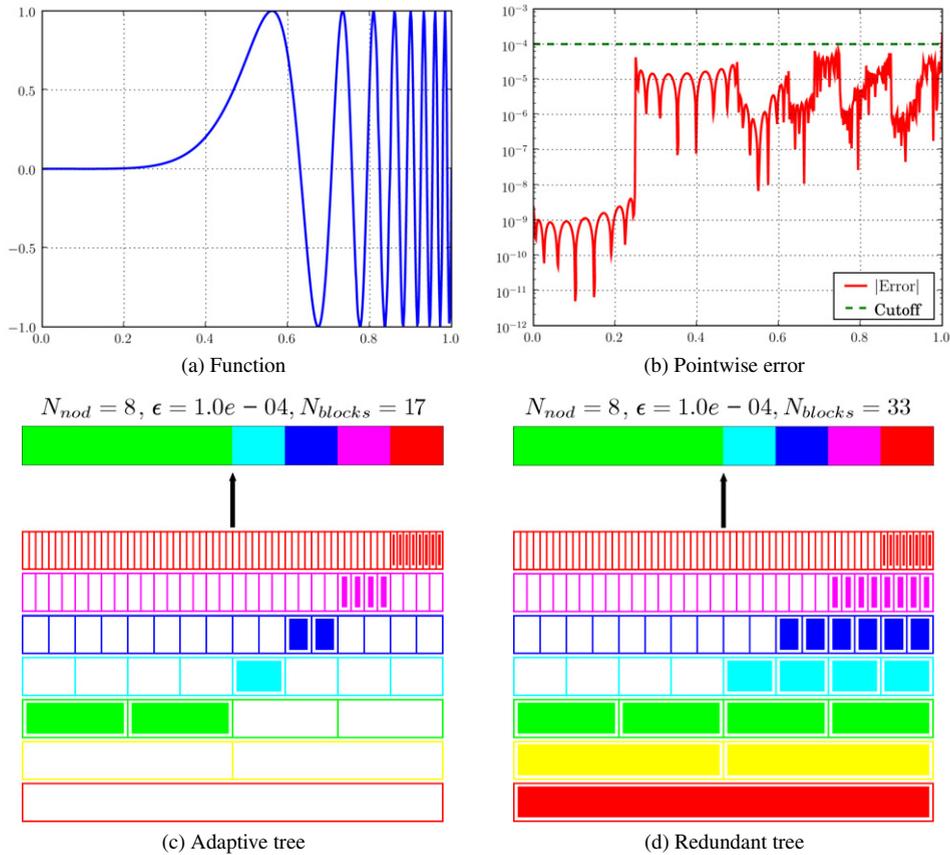
Fig. 1. The function $f(x) = \sin(16\pi x^6)$ shown in (a), is decomposed with $p = 8$ and $\epsilon = 10^{-4}$; (b) shows the pointwise approximation error. (c) is the resulting adaptive tree, where smaller subdivisions are required in regions with higher frequency content. (d) is the redundant tree associated with this adaptive decomposition, where all internal nodes have been filled with data.

structure we will compute a similar tree but where we do keep the coefficients of scaling functions on *all* nodes (leaves and internal).

The coefficients on the internal nodes are redundant since they are computed from the function blocks stored in the leaves. We will thus refer to the tree containing coefficients on all nodes as a *redundant tree*. It is constructed recursively starting from the leaves, by projecting the scaling coefficients from all sibling nodes onto their parent node, using the decomposition (A.15).

Figure 1 shows both the adaptive and the redundant trees for a sample function. This figure displays the coarsest scales at the bottom and progressively finer ones further up, with filled boxes representing nodes where scaling coefficients are stored and empty boxes indicating nodes with no data in them (these do not need to be actually stored in the implementation).

2.3. Modified ns-form

The non-standard form [4] (see also [13] for the version specialized for multiwavelets) of the operator $\mathbf{T}$ is the collection of components of the telescopic expansion

$$\mathbf{T}_n = (\mathbf{T}_n - \mathbf{T}_{n-1}) + (\mathbf{T}_{n-1} - \mathbf{T}_{n-2}) + \cdots + \mathbf{T}_0 = \mathbf{T}_0 + \sum_{j=0}^{n-1} (\mathbf{A}_j + \mathbf{B}_j + \mathbf{C}_j), \tag{3}$$

where $\mathbf{A}_j = \mathbf{Q}_j \mathbf{T} \mathbf{Q}_j$, $\mathbf{B}_j = \mathbf{Q}_j \mathbf{T} \mathbf{P}_j$, and $\mathbf{C}_j = \mathbf{P}_j \mathbf{T} \mathbf{Q}_j$. The main property of this expansion is that the rate of decay of the matrix elements of the operators $\mathbf{A}_j$, $\mathbf{B}_j$ and $\mathbf{C}_j$ away from the diagonal is controlled by the number of vanishing

moments of the basis and, for a finite but arbitrary accuracy ϵ, the matrix elements outside a certain band can be set to zero resulting in an error of the norm less than ϵ. Such behavior of the matrix elements becomes clear if we observe that the derivatives of kernels of Calderón–Zygmund and pseudo-differential operators decay faster than the kernel itself. If we use the Taylor expansion of the kernel to estimate the matrix elements away from the diagonal, then the size of these elements is controlled by a high derivative of the kernel since the vanishing moments eliminate the lower order terms [4]. We note that for periodic kernels the band is measured as a periodic distance from the diagonal, resulting in filled-in 'corners' of a matrix representation.

Let us introduce notation to show how the telescopic expansion (3) is used when applying an operator to a function. If we apply the projection of the operator $\mathbf{T}_{j-1}$ not on its "natural scale" $j-1$, but on the finer scale j, we denote its upsampled version as $\uparrow(\mathbf{T}_{j-1})$. In the matrix representation of $\mathbf{T}_{j-1}$, this operation results in the doubling of the matrix size in each direction. This upsampling $\uparrow(\cdot)$ and downsampling $\downarrow(\cdot)$ notation will also be used for projections of functions.

With this notation, computing $g = \mathbf{T}f$ via (3) splits across scales,

$$
\begin{aligned}
\hat{g}_0 &= \mathbf{T}_0 f_0, \\
\hat{g}_1 &= \big[\mathbf{T}_1 - \uparrow(\mathbf{T}_0)\big] f_1, \\
\hat{g}_2 &= \big[\mathbf{T}_2 - \uparrow(\mathbf{T}_1)\big] f_2, \\
&\;\;\vdots \\
\hat{g}_j &= \big[\mathbf{T}_j - \uparrow(\mathbf{T}_{j-1})\big] f_j, \\
&\;\;\vdots
\end{aligned}
\tag{4}
$$

where $f_j = \mathbf{P}_j f$.

As in the application of the usual ns-form in [4], to obtain g_n after building the set $\{\hat{g}_0, \hat{g}_1, \ldots, \hat{g}_n\}$, we have to compute

$$
g_n = \hat{g}_n + \uparrow\big(\hat{g}_{n-1} + \uparrow\big(\hat{g}_{n-2} + \uparrow\big(\hat{g}_{n-3} + \cdots + (\uparrow \hat{g}_0)\cdots\big)\big)\big).
\tag{5}
$$

The order of the parentheses in this expression is essential, as it indicates the order of the actual operations which are performed starting on the coarsest subspace $\mathbf{V}_0$. For example, if the number of scales $n = 4$, then (5) yields $g_4 = \hat{g}_4 + \uparrow(\hat{g}_3 + \uparrow(\hat{g}_2 + \uparrow(\hat{g}_1 + (\uparrow \hat{g}_0))))$, describing the sequence of necessary operations.

Unfortunately, the sparsity of the non-standard form induced by the vanishing moments of bases is not sufficient for fast practical algorithms in dimensions other than $d = 1$. For algorithms in higher dimensions, we need an additional structure for the remaining non-zero coefficients of the representation. We will use separated representations (see Section 3) introduced in [6,7] and first applied in a multiresolution setting in [8–11]. Within the retained bands, the components of the non-standard form are stored and applied in a separated representation and, as a result, the numerical application of operators becomes efficient in higher dimensions.

2.4. Modified ns-form in 1D

Let us describe a one-dimensional construction for operators on $L^2([0, 1])$ to introduce all the features necessary for a multidimensional algorithm. Since we use banded versions of operators, we need to introduce the necessary bookkeeping.

The "template" for the band structure on scale j comes from the band on the previous scale $j-1$. For each block on scale $j-1$, the upsampling operation $\uparrow(\mathbf{T}_{j-1})$ creates four blocks (all combinations of even/odd row and column indices). We insist on maintaining the strict correspondence between these four blocks and those of $\mathbf{T}_j$. For this reason the description of the retained blocks of $\mathbf{T}_j$ involves the parity of their row and column indices. Let us denote the blocks in the matrix representing $\mathbf{T}_j$ by $t^{j;ll'}$, where $l, l' = 0, \ldots, 2^{j}-1$. Individual elements within these blocks are indexed as $t_{ii'}^{j;ll'}$, where $i, i' = 0, \ldots, p-1$, and p is the order of the multiwavelet basis. For a given width of the band b_j, we keep the operator blocks $t^{j;ll'}$ with indices satisfying

$$
\begin{aligned}
l - b_j + 1 \leqslant l' \leqslant l + b_j, &\quad \text{for even } l, \\
l - b_j \leqslant l' \leqslant l + b_j - 1, &\quad \text{for odd } l.
\end{aligned}
\tag{6}
$$

We denote the banded operators where we keep only blocks satisfying (6) as $\mathbf{T}_j^{b_j}$ and $\uparrow(\mathbf{T}_{j-1})^{b_j}$. If we downsample the operator $\uparrow(\mathbf{T}_{j-1})^{b_j}$ back to its original scale $j-1$, then (6) leads to the band described by the condition

$$l - \lfloor b_j/2 \rfloor \leqslant l' \leqslant l + \lfloor b_j/2 \rfloor, \tag{7}$$

where $\lfloor b_j/2 \rfloor$ denotes the integer part of $b_j/2$. We denote the banded operator on scale $j-1$ as $\mathbf{T}_{j-1}^{\lfloor b_j/2 \rfloor}$, where we retain blocks satisfying (7).

If we now rewrite (4) *keeping only blocks within the bands* on each scale, we obtain

$$\begin{aligned}
\hat{g}_0 &= \mathbf{T}_0 f_0, \\
\hat{g}_1 &= \left[\mathbf{T}_1^{b_1} - \uparrow(\mathbf{T}_0)^{b_1}\right] f_1 = \mathbf{T}_1^{b_1} f_1 - \uparrow(\mathbf{T}_0)^{b_1} f_1, \\
\hat{g}_2 &= \left[\mathbf{T}_2^{b_2} - \uparrow(\mathbf{T}_1)^{b_2}\right] f_2 = \mathbf{T}_2^{b_2} f_2 - \uparrow(\mathbf{T}_1)^{b_2} f_2, \\
&\vdots \\
\hat{g}_j &= \left[\mathbf{T}_j^{b_j} - \uparrow(\mathbf{T}_{j-1})^{b_j}\right] f_j = \mathbf{T}_j^{b_j} f_j - \uparrow(\mathbf{T}_{j-1})^{b_j} f_j, \\
&\vdots
\end{aligned} \tag{8}$$

For any arbitrary but finite accuracy, instead of applying the full $[\mathbf{T}_j - \uparrow(\mathbf{T}_{j-1})]$, we will only apply its banded approximation.

A simple but important observation is that

$$\downarrow\left(\left[\uparrow(\mathbf{T}_{j-1})\right] f_j\right) = \mathbf{T}_{j-1} f_{j-1}, \tag{9}$$

which follows from the fact that $\mathbf{Q}_j \mathbf{P}_j = \mathbf{P}_j \mathbf{Q}_j = 0$, since these are orthogonal projections. Thus, we observe that $\downarrow(\uparrow(\mathbf{T}_{j-1})^{b_j}) = \mathbf{T}_{j-1}^{\lfloor b_j/2 \rfloor}$; so instead of applying $\uparrow(\mathbf{T}_{j-1})^{b_j} f_j$ on scale j, we can obtain the same result using (9), so that $\uparrow(\mathbf{T}_{j-1}^{\lfloor b_j/2 \rfloor} f_{j-1}) = \uparrow(\mathbf{T}_{j-1})^{b_j} f_j$. Therefore, we will compute only $\mathbf{T}_{j-1}^{\lfloor b_j/2 \rfloor} f_{j-1}$ on scale $j-1$ and combine it with computing $\mathbf{T}_{j-1}^{b_{j-1}} f_{j-1}$. Incorporating this into (5), we arrive at

$$g_n = \mathbf{T}_n^{b_n} f_n + \uparrow\left[\left(\mathbf{T}_{n-1}^{b_{n-1}} - \mathbf{T}_{n-1}^{\lfloor b_n/2 \rfloor}\right) f_{n-1} + \uparrow\left[\left(\mathbf{T}_{n-2}^{b_{n-2}} - \mathbf{T}_{n-2}^{\lfloor b_{n-1}/2 \rfloor}\right) f_{n-2} + \cdots + \left[\uparrow\left[\left(\mathbf{T}_0 - \mathbf{T}_0^{\lfloor b_1/2 \rfloor}\right) f_0\right]\right] \cdots\right]\right]. \tag{10}$$

Using this expression yields an efficient algorithm for applying an operator, as on each scale j, $\mathbf{T}_j^{b_j} - \mathbf{T}_j^{\lfloor b_{j+1}/2 \rfloor}$ is a sparse object, due to the cancellation which occurs for most of the blocks. In particular, $\mathbf{T}_j^{b_j} - \mathbf{T}_j^{\lfloor b_{j+1}/2 \rfloor}$ is missing the blocks near the diagonal, and we will refer to it as an *outer band matrix*. We will call $\mathbf{T}_j^{b_j}$ a *whole band matrix* as it contains both the inner and outer bands.

The structure of these matrices is illustrated in Fig. 2. The two empty scales $j = 0, 1$ arise due to the complete cancellation of blocks on these scales. We note that the modified non-standard form is constructed adaptively in the number of scales necessary for a given function. For just two scales, this construction will have the scale $j = 1$ non-empty. Given an adaptive decomposition of a function on n scales, we precompute the modified non-standard form (depicted in Fig. 2 for five scales) on all scales $n, n-1, \ldots, 1$. For matrices requiring $2^n \cdot 2^n$ blocks on the finest scale n, we need to keep and apply only $\mathcal{O}(2^n)$ blocks, as with the original non-standard form in [4].

2.4.1. Adaptive application

Let us show how to use the multiscale representation in (10) to apply the operator T to a function f with controlled accuracy ϵ. We describe an adaptive application of the operator to a function, where we assume (as is often the case) that the tree structure of the input is sufficient to adequately describe the output with accuracy ϵ. This assumption will be removed later.

Our algorithm uses the structure shown in Fig. 2 in an adaptive fashion. We copy the structure of the redundant tree for the input function, and use that as a template to be filled for the output g. For each node of the output tree we determine whether it is a leaf or an internal node: for leaves, we must apply a whole band matrix on the scale of that node, such as the leftmost matrix for $j = 5$ in the example shown in Fig. 2. For internal nodes, we apply an outer band matrix (for that scale). We note that our construction of the operator produces both whole and outer band matrices
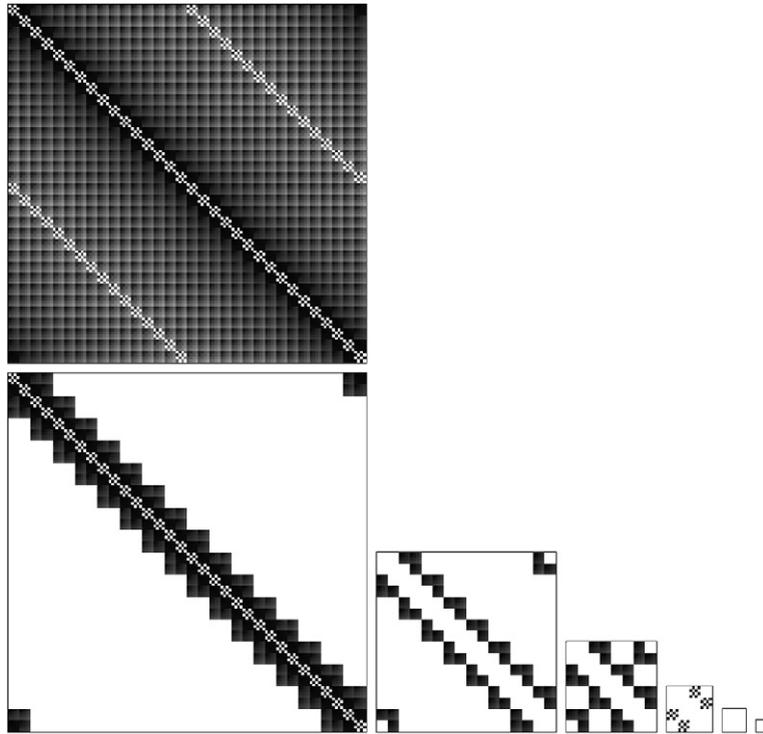
Fig. 2. Modified non-standard form of the convolution operator in (11) in a multiwavelet basis, with white representing 0 and black representing large values. The top matrix is the projection of this operator on $\mathbf{V}_5$, resulting in a dense matrix. The lower half depicts the multiresolution representation in (10) with only the blocks that are actually retained for a given accuracy. We will call the leftmost matrix in this series a *whole band matrix* and all others *outer band matrices*. The two empty outer band matrices on scales $j = 0, 1$ are explained in the main text.

for all scales, and we simply choose the appropriate kind for each node of output as needed. Upon completion of this process, we apply the projection (5) to construct the final adaptive tree representing the output.

Algorithm 2 returns an adaptive tree representing the function g. This tree contains sufficient information to evaluate g at arbitrary points by interpolation and may be used as an input in further computations.

We note that Step 2 in Algorithm 2 naturally resolves the problem that is usually addressed by mortar methods, see, e.g., [18–21]. Since adaptive representations have neighboring blocks of different sizes, they encounter difficulties when applying non-diagonal operators, as they require blocks which do not exist on that scale. Our approach simply constructs these as needed and caches them for reuse, without requiring any additional consideration on the part of the user.

Algorithm 2. Adaptive non-standard form operator application in $d = 1$, $g = \mathbf{T} f$.

    Initialization: Construct the redundant tree for f and copy it as skeleton tree for g (see Section 2.2.1).
    for all scales $j = 0, \ldots, n - 1$ **do**
      for all function blocks g_l^j in the tree for g at scale j **do**
        Step 1. Determine the list of all contributing blocks of the modified ns-form $\mathbf{T}_{ll'}^j$ (*see Section* 2.3):
        if g_l^j belongs to a *leaf* **then**
          Read operator blocks $\mathbf{T}_{ll'}^j$ from row l of *whole band matrix* $\mathbf{T}_j^{b_j}$.
        else
          Read operator blocks $\mathbf{T}_{ll'}^j$ from row l of *outer band matrix* $\mathbf{T}_j^{b_j} - \mathbf{T}_j^{[b_{j+1}/2]}$.
        end if

Step 2. Find the required blocks $f_{l'}^j$ of the input function f:

if function block $f_{l'}^j$ exists in the redundant tree for f **then**
   Retrieve it.
else
   Create it by interpolating from a coarser scale and cache for reuse.
end if
Step 3. Output function block computation:
Compute the resulting output function block according to $\hat{g}_l^j = \sum_{l'} \mathbf{T}_{ll'}^j f_{l'}^j$, where the operation $\mathbf{T}_{ll'}^j f_{l'}^j$
indicates a regular matrix–vector multiplication.
   end for
end for
Step 4. Adaptive projection:
Project resulting output function blocks $\hat{g}_l^j$ on all scales into a proper adaptive tree by using Eq. (5).
Discard from the resulting tree unnecessary function blocks at the requested accuracy.
Return: the function g represented by its adaptive tree.

2.4.2. Numerical example

Let us briefly illustrate the application of the modified ns-form with an example of a singular convolution on the unit circle, the operator with the kernel $K(x) = \cot(\pi x)$,

$$(Cf)(y) = \text{p.v.} \int_0^1 \cot(\pi(y - x)) f(x) \, dx, \tag{11}$$

a periodic analogue of the Hilbert transform. In order to find its representation in multiwavelet bases, we compute

$$r_{ii'}^{j;l} = 2^{-j} \int_{-1}^1 K(2^{-j}(x + l)) \Phi_{ii'}(x) \, dx = 2^{-j} \int_{-1}^1 \cot(\pi 2^{-j}(x + l)) \Phi_{ii'}(x) \, dx, \tag{12}$$

where $\Phi_{ii'}(x)$, $i, i' = 0, \ldots, k - 1$ are cross-correlation functions described in Appendix A.4 and $l = 0, \pm 1, \pm 2$, $\ldots, 2^j - 1$. We compute $r_{ii'}^{j;l}$ using the convergent integrals

$$r_{ii'}^{j;l} = 2^{-j} \sum_{k=i'-i}^{i'+i} c_{ii'}^k \int_0^1 \Phi_{k,0}^+(x) \left(\cot(\pi 2^{-j}(x + l)) + (-1)^{i+i'} \cot(\pi 2^{-j}(-x + l)) \right) dx,$$

where $\Phi_{k,0}^+$ is a polynomial described in Appendix A.4. In our numerical experiment, we apply (11) to the periodic function on $[0, 1]$,

$$f(x) = \sum_{k \in \mathbb{Z}} e^{-a(x+k-1/2)^2},$$

which yields

$$(Cf)(y) = -\sum_{k \in \mathbb{Z}} e^{-a(y+k-1/2)^2} \text{Erfi}\left[\sqrt{a}(y + k - 1/2)\right] = i\sqrt{\frac{\pi}{a}} \sum_{n \in \mathbb{Z}} \text{sign}(n) e^{-n^2\pi^2/a} e^{2\pi i n y}, \tag{13}$$

where $e^{-y^2} \text{Erfi}(y) = \frac{2}{\sqrt{\pi}} \int_0^y e^{s^2 - y^2} \, ds$. Expression (13) is obtained by first observing that the Hilbert transform of e^{-ax^2} is $-e^{-ay^2} \text{Erfi}(\sqrt{a}y)$, and then evaluating the lattice sum, noting that (see [22, formula 4.3.91])

$$\cot(\pi x) = \frac{1}{\pi} \left(\frac{1}{x} + \sum_{k=1}^\infty \frac{2x}{x^2 - k^2} \right).$$

Table 1
Results from evaluating (13) with our algorithm

p	Scales	N_{blocks}	ϵ	E_2
5	[2,3,4]	8	10^{-3}	1.5×10^{-4}
8	[2,4,5]	12	10^{-6}	1.3×10^{-7}
11	[2,4,5]	14	10^{-9}	1.1×10^{-10}
14	[3,4,5]	16	10^{-12}	4.4×10^{-13}

Notes. The order of the basis p is adjusted as a function of the requested precision ϵ. The second column indicates scales present in the adaptive tree for the input. The third column shows the total number of blocks of coefficients in this tree. The last column (E_2) shows the actual error of the computed solution in the ℓ^2 norm.
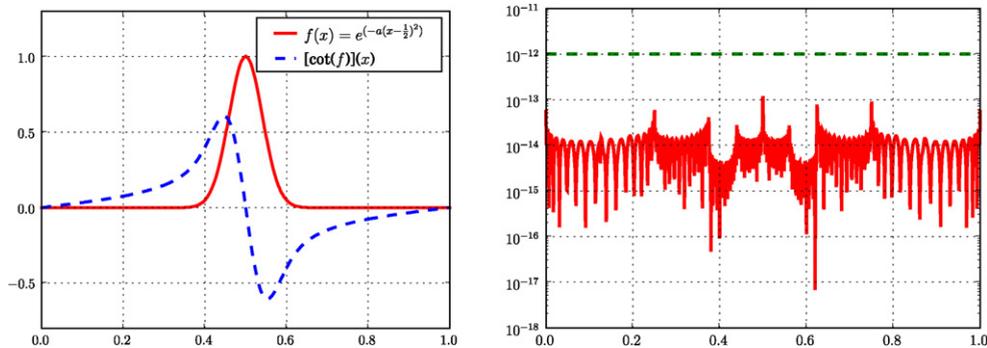
Fig. 3. Results of applying the cotangent kernel to a periodized Gaussian using basis of order $p = 14$ (the last row in Table 1). The pointwise error is shown on the right for a requested accuracy of $\epsilon = 10^{-12}$.

Table 1 summarizes the numerical construction of this solution for $a = 300$, at various requested precisions. Optimal performance is obtained by adjusting the order of the basis p as a function of the requested precision, to ensure that the operator remains a banded matrix with small band, and that the adaptive representation of the input function requires a moderate number of scales. The resulting numerical error (as compared to the exact analytical solution), measured in the ℓ^2 norm, is shown in the last column. Figure 3 shows the input and results for this example, as well as the point-wise error for the case where $\epsilon_{\text{req}} = 10^{-12}$ and $p = 14$ (the last row in the table).

3. Separated representations of integral kernels

The approach we have discussed so far does not efficiently generalize to the application of non-separable multidimensional integral kernels. Since several physically important kernels belong to this category (e.g., the Poisson kernels in $d = 2$ and $d = 3$), additional tools are needed. We now describe the key idea that allows us to perform this generalization to $d > 1$.

We use the separated representation of operators introduced in [6,23] to reduce the computational cost of the straightforward generalization of the multiresolution approach in [4]. Such representations are particularly simple for convolution operators and are based on approximating kernels by a sum of Gaussians [8–11,23]. This approximation has a multiresolution character by itself and requires a remarkably small number of terms. In fact, our algorithm uses the coefficients and the exponents of the Gaussian terms as the only input from which it selects the necessary terms, scale by scale, according the desired accuracy threshold ϵ. Therefore, our algorithm works for all operators with kernels that admit approximation by a sum of Gaussians. Examples of such operators include the Poisson and the bound state Helmholtz Green's functions, the projector on divergence free functions, as well as regular and fractional derivative operators. Let us consider a particular family of operators $(-\Delta + \mu^2 I)^{-\alpha}$, where $\mu \geqslant 0$ and $0 < \alpha < 3/2$. The kernel of this operator

$$K_{\mu,\alpha}(r) = 2^{-\frac{3}{2}+\alpha} \cdot C_\alpha \cdot \left(\frac{\mu}{r}\right)^{\frac{3}{2}-\alpha} \mathrm{K}_{\frac{3}{2}-\alpha}(\mu r),$$
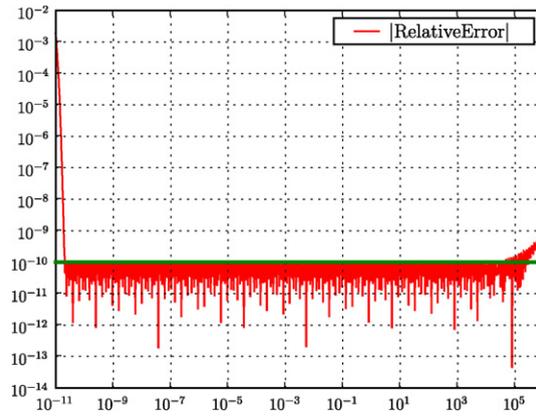
Fig. 4. Relative error of the Gaussian approximation for the Poisson kernel in 3 dimensions. This unoptimized expansion uses 299 terms to cover a dynamic range of roughly 15 decades with $\epsilon = 10^{-10}$ relative accuracy.

where $K_{\frac{3}{2}-\alpha}$ is the modified Bessel function, $r = \|\mathbf{x} - \mathbf{y}\|$ and $C_\alpha = 2 \cdot 2^{-2\alpha} \pi^{-\frac{3}{2}} / \Gamma(\alpha)$, has an integral representation

$$K_{\mu,\alpha}(r) = C_\alpha \int_{-\infty}^{\infty} e^{-r^2 e^{2s}} e^{-\frac{1}{4}\mu^2 e^{-2s} + (3-2\alpha)s} \, ds. \tag{14}$$

Using the trapezoidal rule, we construct an approximation valid over a range of values $\delta \leqslant r \leqslant R$ with accuracy ϵ, of the form

$$\left| K_{\mu,\alpha}(r) - \sum_{m=1}^{M} w_m e^{-\tau_m r^2} \right| \leqslant \epsilon K_{0,\alpha}(r) = \epsilon \frac{\Gamma(3/2 - \alpha) \cdot C_\alpha}{2r^{3-2\alpha}}, \tag{15}$$

where $\tau_m = e^{2s_m}$, $w_m = hC_\alpha e^{-\mu^2 e^{-2s_m}/4 + (3-2\alpha)s_m}$, $h = (B-A)/M$ and $s_m = A + mh$. The limits of integration, A, B and the step size h are selected as indicated in [23], where it is shown that for a fixed accuracy ϵ the number of terms M in (15) is proportional to $\log(R\delta^{-1})$. Although it is possible to select δ and R following the estimates in [24] and optimize the number of terms for a desired accuracy ϵ, in this paper we start with an approximation that has an obviously excessive range of validity and thus, an excessive number of terms.

An example of such approximation is shown in Fig. 4. For a requested tolerance of $\epsilon = 10^{-10}$, roughly 300 terms are enough to provide a valid approximation over a range of 15 decades. We then let the algorithm choose the necessary terms, scale by scale, to satisfy the user-supplied accuracy requirement ϵ. This approach may end up with a few extra terms on some scales in comparison with that using a nearly optimal number of terms [8–11]. Whereas the cost of applying a few extra terms is negligible, we gain significantly in having a much more flexible and general algorithm.

We note that approximation in (15) clearly reduces the problem of applying the operator to that of applying a sequence of Gauss transforms [25,26], one by one. From this point of view, the algorithm that we present may be considered as a procedure for applying a linear combination of Gauss transforms simultaneously.

In order to represent the kernel K of the operator in multiwavelet bases, we need to compute the integrals,

$$r^{j;\ell}_{i_1 i'_1, i_2 i'_2, i_3 i'_3} = \sum_{m=1}^{M} 2^{-3j} \int_B w_m e^{-\tau_m \|2^{-j}(\mathbf{x}+\ell)\|^2} \Phi_{i_1 i'_1}(x_1) \Phi_{i_2 i'_2}(x_2) \Phi_{i_3 i'_3}(x_3) \, d\mathbf{x}, \tag{16}$$

where $\Phi_{ii'}(x)$ are the cross-correlations of the scaling functions (see Appendix A). We obtain

$$r^{j;\ell}_{i_1 i'_1, i_2 i'_2, i_3 i'_3} = \sum_{m=1}^{M} w_m F^{j;m,l_1}_{i_1 i'_1} F^{j;m,l_2}_{i_2 i'_2} F^{j;m,l_3}_{i_3 i'_3}, \tag{17}$$

where

$$F_{ii'}^{j;m,l} = \frac{1}{2^j} \int\limits_{-1}^{1} e^{-\tau_m(x+l)^2/4^j} \Phi_{ii'}(x) \, dx. \tag{18}$$

Since the Gaussian kernel is not homogeneous, we have to compute integrals (18) for each scale. Although in principle $l \in \mathbb{Z}$, in the next section we explain how to restrict it to a limited range on each scale, for a given accuracy ϵ.

4. Modified ns-form of a multidimensional operator

In this section, we describe how the separated representation approximations of Section 3 can be used to construct a multidimensional extension of the ns-form representation from Section 2.3, using only one-dimensional quantities and norm estimates. This makes our approach viable for $d > 1$. We use the modified ns-form as in the one-dimensional case described in Section 2.3. We find the ns-form essential for adaptive algorithms in more than one dimension, since:

(1) Scales do not interact as the operator is applied. All interactions between scales are accounted for by the (inexpensive) projection at the final step of the algorithm.
(2) For the same reason, the subdivision of space at different scales naturally maps into the supporting data structures. We note that one of the main difficulties in developing adaptive algorithms is in organizing computations with blocks of an adaptive decomposition of a function from different scales but with a common boundary. The methods for such computations are known as mortar elements methods. In our approach this issue does not present any obstacle, as all relevant interactions are naturally accounted for by the data structures.

The key feature that makes our approach efficient in dimensions $d \geqslant 2$ is the separated structure of the modified ns-form. Namely, the blocks of $\mathbf{T}_j - \uparrow(\mathbf{T}_{j-1})$ are of the form (for $d = 3$)

$$\mathbf{T}_j^\ell - \uparrow(\mathbf{T}_{j-1}^\ell) = \sum_{m=1}^{M} w_m F^{j;ml_1} F^{j;ml_2} F^{j;ml_3} - \uparrow\left(\sum_{m=1}^{M} w_m F^{j-1;ml_1} F^{j-1;ml_2} F^{j-1;ml_3}\right)$$

$$= \sum_{m=1}^{M} w_m F^{j;ml_1} F^{j;ml_2} F^{j;ml_3} - \sum_{m=1}^{M} w_m \uparrow(F^{j-1;ml_1}) \cdot \uparrow(F^{j-1;ml_2}) \cdot \uparrow(F^{j-1;ml_3}). \tag{19}$$

As in the case $d = 1$, the norm of the operator blocks of $\mathbf{T}_j^\ell - \uparrow(\mathbf{T}_{j-1}^\ell)$ decays rapidly with $\|\ell\|$, $\ell = (l_1, l_2, l_3)$, and the rate of decay depends on the number of vanishing moments of the basis [4]. Moreover, we limit the range of shift indices $\|\ell\|$ using only one-dimensional estimates of the differences

$$F_{ii'}^{j;m,2l} - \uparrow F_{ii'}^{j;m,l} \quad \text{and} \quad F_{ii'}^{j;m,2l+1} - \uparrow F_{ii'}^{j;m,l} \tag{20}$$

of operator blocks computed via (18). The norms of individual blocks $F_{ii'}^{j;m,l}$ are illustrated in Fig. 5a, for scale $j = 1$.

By selecting the number of vanishing moments for a given accuracy, it is sufficient to use $\|\ell\|_\infty \leqslant 2$ in practical applications that we have encountered. Also, not all terms in the Gaussian expansion of an operator need to be included since, depending on the scale j, their contribution may be negligible for a given accuracy, as shown in Fig. 5b. Below we detail how we select terms of the Gaussian expansion on a given scale as well as the significant blocks of $\mathbf{T}_j^\ell - \uparrow(\mathbf{T}_{j-1}^\ell)$. This procedure establishes the band structure of the operator. We then project the *banded* operator $\uparrow(\mathbf{T}_{j-1}^{b_j})$ back to the scale $j - 1$ and then combine blocks on the natural scale for each projection in order to apply the operator efficiently as was explained in Section 2.3 for the one-dimensional case.

We note that in deciding which terms to keep in (19), we do not compute the difference between the full three-dimensional blocks as it would carry a high computational cost; instead we use estimates based on the one-dimensional blocks of the separated representation. We note that since the resulting band structure depends only on the operator and the desired accuracy of its approximation, one of the options is to store such band information as it is likely to be reused.
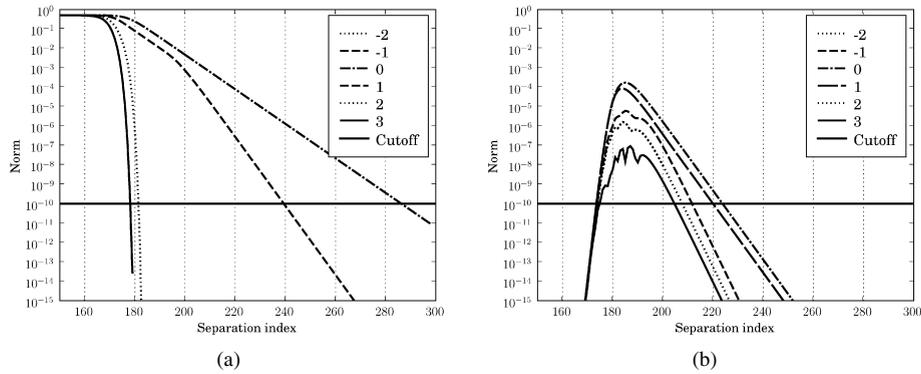
Fig. 5. Comparison of norms of matrix blocks generated by the Gaussian approximation for the Poisson kernel in dimension $d = 3$. In each picture, the curves correspond to the different offsets l for which blocks are generated. Part (b) illustrates the estimate in (23) (see main text for details). (a) Norms of each one-dimensional block computed via (18) for scale $j = 1$, as a function of the index m in the separated representation. (b) Norm estimates (23) for scale $j = 1$ as a function of the index m in the separated representation. Based on these estimates, only terms above the cutoff are actually applied.

In order to efficiently identify the significant blocks in $\mathbf{T}_j^\ell - \uparrow(\mathbf{T}_{j-1}^\ell)$ as a function of ℓ, we develop norm estimates based only on the one-dimensional blocks. The difference between two terms of the separated representation, say $F_1 F_2 F_3 - G_1 G_2 G_3$, may be written as

$$F_1 F_2 F_3 - G_1 G_2 G_3 = (F_1 - G_1) F_2 F_3 + G_1 (F_2 - G_2) F_3 + G_1 G_2 (F_3 - G_3).$$

We average six different combinations of the three terms to include all directions in a symmetric manner, which results in the norm estimate

$$\| F_1 F_2 F_3 - G_1 G_2 G_3 \| \leqslant \frac{1}{6} \operatorname{sym}\big[\|F_1 - G_1\| \|F_2\| \|F_3\| + \|G_1\| \|F_2 - G_2\| \|F_3\| + \|G_1\| \|G_2\| \|F_3 - G_3\|\big], \quad (21)$$

where the symmetrization is over the three directions and generates 18 terms. For the rotationally symmetric operators with Gaussian expansion as in (15) computing the right-hand side in this estimate involves just three types of one dimensional blocks and their norms,

$$N_{\mathrm{dif}}^{j;m;l} = \big\| F^{j;m;l} - \uparrow\big(F^{j-1;m;l}\big)\big\|, \qquad N_F^{j;m;l} = \big\| F^{j;m;l}\big\|, \qquad N_{\uparrow F}^{j;m;l} = \big\|\uparrow\big(F^{j-1;m;l}\big)\big\|, \quad (22)$$

where index j indicates the scale, m the term in the Gaussian expansion (15), and l the position of the block in a given direction.

These estimates allow us to discard blocks whose norm falls below a given threshold of accuracy, namely, for each multi-index ℓ, we estimate

$$\big\|\mathbf{T}_j^\ell - \uparrow\big(\mathbf{T}_{j-1}^\ell\big)\big\| \leqslant \frac{1}{6} \sum_{m=1}^{M} w_m \operatorname{sym}\big[N_{\mathrm{dif}}^{j;m;l_1} N_F^{j;m;l_2} N_F^{j;m;l_3} + N_{\uparrow F}^{j;m;l_1} N_{\mathrm{dif}}^{j;m;l_2} N_F^{j;m;l_3}$$
$$+ N_{\uparrow F}^{j;m;l_1} N_{\uparrow F}^{j;m;l_2} N_{\mathrm{dif}}^{j;m;l_3} \big]. \quad (23)$$

For each scale j and each block $\mathbf{T}_j^\ell - \uparrow(\mathbf{T}_{j-1}^\ell)$ labeled by the multi-index $\ell = (l_1, l_2, l_3)$, we compute all terms of the sum in (23) and identify the range $[m_1, m_2]$ which we need to keep for that block, by discarding from the sum terms whose cumulative contribution is below ϵ. If the entire sum falls below ϵ, this range may be empty and the entire $\mathbf{T}_j^\ell - \uparrow(\mathbf{T}_{j-1}^\ell)$ is discarded. The range $[m_1, m_2]$ differs significantly depending whether or not the block is affected by the singularity of the kernel as is illustrated in Fig. 5. In Fig. 5a the rate of decay for the blocks with shift $|l| = 2, 3$ is significantly faster than for the blocks with $|l| \leqslant 1$ affected by the singularity. We note that all blocks of the first 150 terms in the separated representation (17) have norm 1 (and rank 1 as matrices) and are not shown in Fig. 5a.

Since the difference in (23) involves blocks upsampled from a coarser scale, all shifts $|l| \leqslant 3$ are affected by the singularity. Figure 5b shows the r.h.s. of the estimate in (23) for different shifts along one of the directions, where the blocks along the other two directions are estimated by the maximum norm over all possible shifts.

After discarding blocks with norms less that ϵ using the estimate in (23), we downsample the remaining blocks of $\uparrow(\mathbf{T}^{\ell}_{j-1})$ back to the original scale. This leaves only blocks of $\mathbf{T}^{\ell}_{j}$ on the scale j and we remove additional blocks of $\mathbf{T}^{\ell}_{j}$ for the shifts $|l| = 2, 3$ where the decay is sufficiently fast to make their contribution less than ϵ.

This leads us to arrange the blocks on each scale into several subsets by the effect the singularity of the kernel has on them and find the appropriate range $[m_1, m_2]$ for each subset. There are three such sets in dimension $d = 2$ and four sets in dimension $d = 3$. For each index l, we will say that the index belongs to the *core* if $l = -1, 0, 1$ and to the *boundary* otherwise. The core indices correspond to one-dimensional blocks whose defining integrals include the singularity of the kernel. We then divide all possible values of the multi-index $\ell = (l_1, l_2, l_3)$, according to the number of core indices it has. In $d = 3$ this gives us four sets:

- Core: all indices (l_1, l_2, l_3) belong to the core.
- Boundary-1: two of the indices belong to the core and one to the boundary.
- Boundary-2: one of the indices belongs to the core, the other two to the boundary.
- Boundary-3: all indices (l_1, l_2, l_3) belong to the boundary.

We then find the range $[m_1, m_2]$ for each subset and apply blocks of each subset separately, thus avoiding unnecessary computations with blocks whose contribution is negligible. This range analysis only needs to be done once per operator and the desired accuracy, and the results may be saved for repeated use.

5. Multidimensional adaptive application of ns-form

In this section we present an algorithm for applying the modified non-standard form which is an extension of (2) (based on (8) and (10)) to higher dimensions. We are now seeking to compute

$$g(\mathbf{x}) = [Tf](\mathbf{x}) = \int K(\mathbf{y} - \mathbf{x}) f(\mathbf{y}) \, d\mathbf{y},$$

where $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ for $d = 2, 3$. The separated approximation (19) reduces the complexity of applying the operator by allowing partial factorization of the nested loops in each scale indicated by the order of summation and illustrated for $d = 3$,

$$g^{j;l_1 l_2 l_3} = \sum_m w_m \left\{ \sum_{l'_1} F^{j;m;l_1 - l'_1} \sum_{l'_2} F^{j;m,l_2 - l'_2} \sum_{l'_3} F^{j;m,l_3 - l'_3} \right.$$
$$\left. - \uparrow \left[\sum_{l'_1} F^{j-1;m;l_1 - l'_1} \sum_{l'_2} F^{j-1;m,l_2 - l'_2} \sum_{l'_3} F^{j-1;m,l_3 - l'_3} \right] \right\} f^{j;l'_1 l'_2 l'_3}.$$

As described in the previous section, this evaluation is done by regions of indices. These regions of indices are organized so that (for a given accuracy) the number of retained terms of the separated representation is roughly the same for all blocks within each region. Thus, we perform the summation over the terms of the separated representation last, applying only the terms that actually contribute to the result above the requested accuracy threshold, according to estimate (23).

Just as in the one-dimensional case, we use (19) in a 'natural scale' manner. That is, blocks belonging to scale j are only applied on that scale. As in one-dimensional case, the interaction between scales is achieved by the multidimensional version of the projection (10) that redistributes blocks accumulated in this manner properly between the scales to obtain the adaptive tree for the resulting output function. The overall approach is the same as described in Algorithm 2. We note that, as expected, the separated representation requires more detailed bookkeeping when constructing the data structures for the operator.

Remark 1. Our multiresolution decomposition corresponds to the geometrically varying refinement in finite element methods. In this case the adjoining boxes do not necessarily share common vertices, leading to a non-conforming grid. In finite element methods such situation requires additional construction provided by the mortar element methods. Mortar element methods were introduced by Patera and his associates, see e.g. [18–21]. These methods permit

coupling discretizations of different types in non-overlapping domains. Such methods are fairly complicated and involve, for example, the introduction of interface conditions through an L^2 minimization. In our approach we do not face these issues at all and do not have to introduce any additional interface conditions. The proper construction for adjoining boxes is taken care by the redundant tree data structure and Step 2 of Algorithm 3 for applying the kernel, which generates the necessary missing boxes on appropriate scales.

Remark 2. Although Algorithm 3 applies convolution operators, only minor changes are needed to use it for non-convolutions. Of course in such case, the separated representation for the modified ns-form should be constructed appropriately.

Algorithm 3. Adaptive non-standard form operator application in multiple dimensions (illustrated for $d = 2$), $g = \mathbf{T}f$.

Initialization: Construct the redundant tree for f and copy it as skeleton tree for g (see Section 2.2.1).
for all scales $j = 0, \ldots, n - 1$ **do**
  for all function blocks $g_{l_1 l_2}^j$ in the tree for g at scale j **do**
    Step 1. Determine the list of all Core, Boundary-1 and Boundary-2 contributing operator blocks of the modified ns-form $F^{j;m;l_1-l_1'}$, $F^{j;m;l_2-l_2'}$ (see Section 4):
    if $g_{l_1 l_2}^j$ belongs to a *leaf* **then**
      Read operator blocks $F^{j;m;l_1-l_1'}$, $F^{j;m;l_2-l_2'}$ for Core, Boundary-1 and Boundary-2, and their weights w_m and corresponding ranges from the separated representation.
    else
      Read operator blocks $F^{j;m;l_1-l_1'}$, $F^{j;m;l_2-l_2'}$ for Boundary-1 and Boundary-2, and their weights w_m and corresponding ranges from the separated representation.
    end if
    Step 2. Find the required blocks $f_{l_1' l_2'}^j$ of the input function f:
    if function block $f_{l_1' l_2'}^j$ exists in the redundant tree for f **then**
      Retrieve it.
    else
      Create it by interpolating from a coarser scale and cache for reuse.
    end if
    Step 3. Output function block computation:
    For each set *S* of indices determined in *Step 1* (*Core, Boundary-1, Boundary-2*) *and the corresponding ranges of terms in the separated representation, compute the sum*
$$\hat{g}_{l_1 l_2}^{j;S} = \sum_m w_m \sum_{l_1'} F^{j;m;l_1-l_1'} \sum_{l_2'} F^{j;m;l_2-l_2'} f_{l_1' l_2'}^j,$$
    Add all computed sums to obtain $\hat{g}_{l_1 l_2}^j$.
  end for
end for
Step 4. Adaptive projection:
Project resulting output function blocks $\hat{g}_{l_1 l_2}^j$ on all scales into a proper adaptive tree by using Eq. (5).
Discard from the resulting tree unnecessary function blocks at the requested accuracy.
Return: the function g represented by its adaptive tree.

5.1. Operation count estimates

5.1.1. Adaptive decomposition of functions

The cost of adaptively decomposing a function in d dimensions is essentially that of an adaptive wavelet transform. Specifically, it takes $\mathcal{O}(N_{\text{blocks}} \cdot p^{d+1})$ operations to compute such representation, where N_{blocks} is the final number of

significant blocks in the representation and p is the order of multiwavelet basis chosen. In comparison with the usual wavelet transform, it appears to be significantly more expensive. However, these $\mathcal{O}(p^{d+1})$ operations process $\mathcal{O}(p^d)$ points, thus in counting significant coefficients as it is done in the usual adaptive wavelet transform, we end up with $\mathcal{O}(p)$ operations per point.

5.1.2. Operator application

The cost of applying an operator in the modified ns-form is $\mathcal{O}(N_{\text{blocks}}Mp^{d+1})$, where N_{blocks} is the number of blocks in the adaptive representation of the input function, M is the separation rank of the kernel in (15) and p is the order of the multiwavelet basis. For a given desired accuracy ϵ, we typically select $p \propto \log \epsilon^{-1}$; M has been shown to be proportional to $(\log \epsilon^{-1})^\nu$, where ν depends on the operator [23]. In our numerical experiments, M is essentially proportional to $\log \epsilon^{-1}$, since we never use the full separated representation, as discussed in Section 4 and illustrated in Fig. 5.

This operation count can be potentially reduced to $\mathcal{O}(N_{\text{blocks}}Mp^d)$ by using the structure of the matrices in 18, and we plan to address this in the future.

5.1.3. Final projection

After the operator has been applied to a function in a scale-independent fashion, a final projection step is required as discussed in Section 2.3. This step requires $\mathcal{O}(N_{\text{blocks}} \cdot p^d)$ operations, the same as in the original function decomposition. In practice, this time is negligible compared to the actual operator application.

6. Numerical examples

6.1. The Poisson equation

We illustrate the performance of the algorithm by solving the Poisson equation in $d = 3$

$$\Delta \phi(\mathbf{r}) = -\rho(\mathbf{r}) \tag{24}$$

with free space boundary conditions, $\phi(\mathbf{r}) \to 0$ and $\partial \phi / \partial r \to 0$ as $r \to \infty$. We write the solution as

$$\phi(\mathbf{r}) = \frac{1}{4\pi} \int_{\mathbb{R}^3} \frac{1}{|\mathbf{r} - \mathbf{r}'|} \rho(\mathbf{r}') \, d\mathbf{r}'$$

and adaptively evaluate this integral. We note that our method can equally be used for $d = 2$, since the corresponding Green's function can also be represented as a sum of Gaussians, and the operator application algorithm has been implemented in for both $d = 2$ and $d = 3$.

For our test we select

$$\phi(\mathbf{r}) = \sum_{i=1}^{3} e^{-\alpha|\mathbf{r} - \mathbf{r}'|^2},$$

so that we solve the Poisson equation with

$$\rho(\mathbf{r}) = -\Delta \phi(\mathbf{r}) = -\sum_{i=1}^{3} \left(4\alpha^2|\mathbf{r} - \mathbf{r}_i|^2 - 6\alpha\right) e^{-\alpha|\mathbf{r} - \mathbf{r}_i|^2}.$$

Our parameters are chosen as follows: $\alpha = 300$, $r_1 = (0.5, 0.5, 0.5)$, $r_2 = (0.6, 0.6, 0.5)$ and $r_3 = (0.35, 0.6, 0.5)$. These ensure that $\rho(r)$ is well below our requested thresholds on the boundary of the computational domain. All numerical experiments were performed on a Pentium-4 running at 2.8 GHz, with 2 GB of RAM. The results are summarized in Table 2.

In order to gauge the speed of algorithm in reasonably computer-independent terms, we use a similar approach to that of [17] and also provide timings of the fast Fourier transform (FFT). Specifically, we display timings for two FFTs as an estimate of the time needed to solve the Poisson equation with a smooth right-hand side and periodic boundary conditions in a cube. As in [17], we compute the rate that estimates the number of processed points per second. We

Table 2
Accuracy and timings for the adaptive solution of the Poisson equation in (24)

p	E_2	E_∞	Time (s)	Rate (pts/s)
Requested $\epsilon = 10^{-3}$				
6	5.0×10^{-3}	7.9×10^{-1}	1.2	7.2×10^4
8	1.7×10^{-3}	1.2×10^{-1}	0.51	7.3×10^4
10	4.4×10^{-4}	3.7×10^{-2}	0.68	1.1×10^5
Requested $\epsilon = 10^{-6}$				
10	4.7×10^{-6}	3.6×10^{-4}	10.3	5.7×10^4
12	8.5×10^{-6}	4.3×10^{-5}	13.5	7.5×10^4
14	6.9×10^{-8}	5.2×10^{-6}	20.0	8.0×10^4
Requested $\epsilon = 10^{-9}$				
16	2.5×10^{-10}	2.2×10^{-8}	68.1	3.5×10^4
18	7.7×10^{-11}	3.5×10^{-9}	100.3	3.4×10^4
20	1.2×10^{-10}	1.8×10^{-8}	133.4	3.5×10^4

Table 3
Timings of two 3D FFTs to estimate the speed of a non-adaptive, periodic Poisson solver on a cube for smooth functions

Size	Time (s)	Rate (pts/s)
32^3	0.02	1.7×10^6
64^3	0.22	1.2×10^6
128^3	2.21	9.5×10^5
256^3	20.7	8.1×10^5
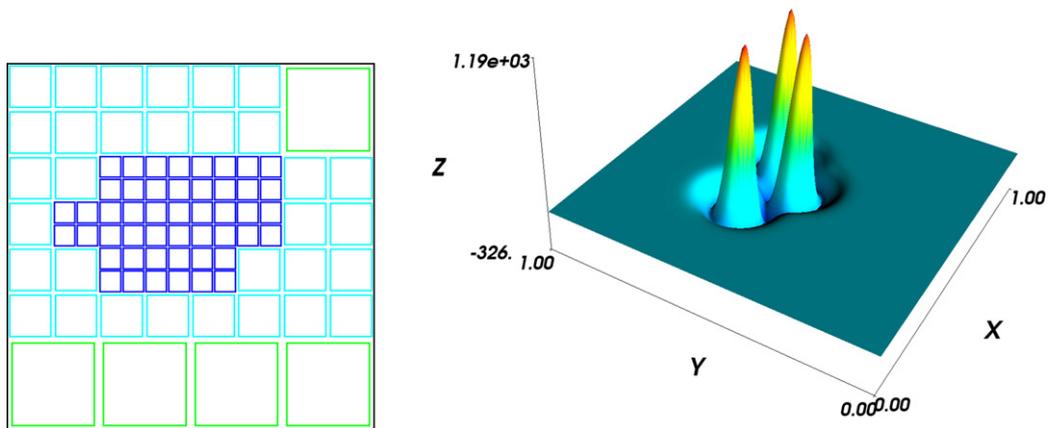
Fig. 6. A two-dimensional slice of the three-dimensional subdivision of space by the scaling functions and an illustration of the source term for the Poisson equation (24).

observe that for our adaptive algorithm such rate varies between 3.4×10^4 and 1.1×10^5 (see Table 2), whereas for the FFTs it is around 10^6 (see Table 3). We note that our algorithm is not fully optimized, namely, we do not use the structure of the matrices in (18) and the symmetries afforded by the radial kernels. We expect a substantial impact on the speed by introducing these improvements and will report them separately.

We note that the multigrid method (see, e.g., [27,28]) is frequently used as a tool for solving the Poisson equation (and similar problems) in differential form. The FFT-based gauge suggested in [17] is useful for comparisons with these algorithms as well.
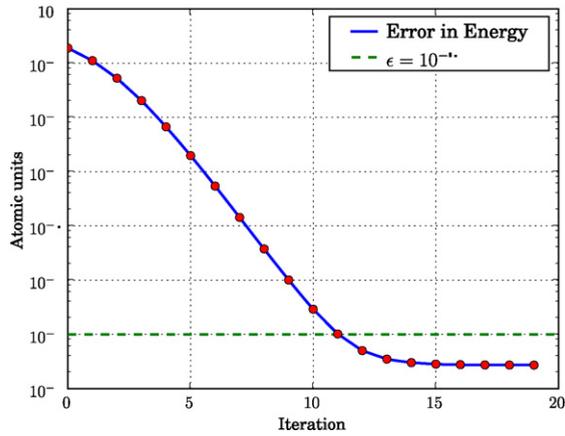
Fig. 7. Convergence of the iteration to obtain the ground state of the hydrogen atom computed via formulation in (26) for the non-relativistic Schrödinger equation. The requested accuracy in applying the Green's function is set to 10^{-6}.

6.2. The ground state of the hydrogen atom

A simple example of computing the ground state of the hydrogen atom illustrates the numerical performance of the algorithms developed in this paper, and their utility for constructing more complex codes. The eigenfunctions ψ for the hydrogen atom satisfy the time-independent Schrödinger equation (written in atomic units and spherical coordinates),

$$-\frac{1}{2}\Delta\psi - \frac{1}{r}\psi = E\psi, \tag{25}$$

where E is the energy eigenvalue. For the ground state, $E = -1/2$ and the (unnormalized) wave function is $\psi = e^{-r}$. Following [29], we write

$$\phi = -2G_\mu V\phi, \tag{26}$$

where $G_\mu = (-\Delta + \mu^2 I)^{-1}$ is the Green's function for some μ and $V = -1/r$ is the nuclear potential. For $\mu = \sqrt{-2E}$ the solution ϕ of (26) has $\|\phi\|_2 = 1$ and coincides with that of (25). We solve (26) by a simple iteration starting from some value μ_0 and changing μ to obtain the solution with $\|\phi\|_2 = 1$. The algorithm proceeds as follows:

(1) Initialize with some value μ_0 and function ϕ. The number of iterations of the algorithm is only weakly sensitive to these choices.
(2) Compute the product of the potential V and the function ϕ.
(3) Apply the Green's function G_μ to the product $V\phi$ via the algorithm of this paper to compute

$$\phi_{\text{new}} = -2G_\mu V\phi.$$

(4) Compute the energy for ϕ_{new},

$$E_{\text{new}} = \frac{\frac{1}{2}\langle\nabla\phi_{\text{new}}, \nabla\phi_{\text{new}}\rangle + \langle V\phi_{\text{new}}, \phi_{\text{new}}\rangle}{\langle\phi_{\text{new}}, \phi_{\text{new}}\rangle}.$$

(5) Set $\mu = \sqrt{-2E_{\text{new}}}$, $\phi = \phi_{\text{new}}/\|\phi_{\text{new}}\|$ and return to Step 2.

The iteration is terminated as the change in μ and $\|\phi\| - 1$ falls below the desired accuracy. The progress of the iteration is illustrated in Fig. 7. The computations in Steps 2 and 4 use the three-dimensional extension of the approach described in [13], to compute point-wise multiplications of adaptively represented functions and weak differential operators of the same.

This example illustrates an application of our algorithm to problems in quantum chemistry. Multiresolution quantum chemistry developed in [8–11] also uses separated representations. The main technical difference with [8–11]

is that we use the modified non-standard form and apply operator blocks on their "natural" scale, thus producing a fully adaptive algorithm. We are currently using this algorithm as part of a new method for solving the multiparticle Schrödinger equation and will report the results elsewhere.

7. Discussion and conclusions

We have shown that a combination of separated and multiresolution representations of operators yields a new multidimensional algorithm for applying a class of integral operators with radial kernels. We note that the same algorithm is used for all such operators as they are approximated by a weighted sum of Gaussians. This fact makes our approach applicable across multiple fields, where a single implementation of the core algorithm can be reused for different specific problems. The algorithm is fully adaptive and avoids issues usually addressed by mortar methods.

The method of approximation underlying our approach is distinct from that of the FMM, has similar efficiency and has the advantage of being more readily extendable to higher dimensions. We also note that semi-analytic approximations via weighted sums of Gaussians provide additional advantages in some applications. Although we described the application of kernels in free space, there is a simple extension to problems with radial kernels subject to periodic, Dirichlet or Neumann boundary conditions on a cube that we will describe separately.

The algorithm may be extended to classes of non-convolution operators, e.g., the Calderon–Zygmund operators. For such extensions the separated representation may not be available in analytic form, as it is for the operators of this paper, and may require a numerical construction. The separated representation of the kernel permits further generalization of our approach to dimensions $d \gg 3$ for applying operators to functions in separated representation.

A notable remaining challenge is an efficient, high order extension of this technique to the application of operators on domains with complicated geometries and surfaces.

Appendix A

A.1. Scaling functions

We use either the Legendre polynomials $P_0, \ldots, P_{p-1}$ or the interpolating polynomials on the Gauss–Legendre nodes in $[-1, 1]$ to construct an orthonormal basis for each subspace $\mathbf{V}_j$ [12,13].

Let us briefly describe some properties of the Legendre scaling functions ϕ_k, $k = 0, \ldots, p-1$, defined as

$$\phi_k(x) = \begin{cases} \sqrt{2k+1} P_k(2x - 1), & x \in [0, 1], \\ 0, & x \notin [0, 1], \end{cases} \tag{A.1}$$

and forming a basis for $\mathbf{V}_0$. The subspace $\mathbf{V}_j$ is spanned by $2^j p$ functions obtained from $\phi_0, \ldots, \phi_{p-1}$ by dilation and translation,

$$\phi_{kl}^j(x) = 2^{j/2} \phi_k(2^j x - l), \quad k = 0, \ldots, p-1, \ l = 0, \ldots, 2^j - 1. \tag{A.2}$$

These functions have support on $[2^{-j}l, 2^{-j}(l+1)]$ and satisfy the orthonormality condition

$$\int_{-\infty}^{\infty} \phi_{kl}^j(x) \phi_{k'l'}^j(x) \, dx = \delta_{kk'} \delta_{ll'}. \tag{A.3}$$

A function f, defined on $[0, 1]$, is represented in the subspace $\mathbf{V}_j$ by its normalized Legendre expansion

$$f(x) = \sum_{l=0}^{2^j-1} \sum_{k=0}^{p-1} s_{kl}^j \phi_{kl}^j(x), \tag{A.4}$$

where the coefficients s_{kl}^j are computed via

$$s_{kl}^j = \int_{2^{-j}l}^{2^{-j}(l+1)} f(x) \phi_{kl}^j(x) \, dx. \tag{A.5}$$

As long as $f(x)$ is smooth enough and is well approximated on $[2^{-j}l, 2^{-j}(l+1)]$ by a polynomial of order up to $2p - 1$, we may use Gauss–Legendre quadratures to calculate the s_{kl}^j via

$$s_{kl}^j = 2^{-j/2} \sum_{i=0}^{p-1} f\left(2^{-j}(x_i + l)\right) \phi_k(x_i) w_i, \tag{A.6}$$

where $x_0, \ldots, x_{p-1}$ are the roots of $P_p(2x - 1)$ and $w_0, \ldots, w_{p-1}$ are the corresponding quadrature weights.

In more than one dimension, the above formulas are extended by using a tensor product basis in each subspace. For example, in two dimensions Eq. (A.4) becomes

$$f(x, x') = \sum_{l=0}^{2^j-1} \sum_{k=0}^{p-1} \sum_{l'=0}^{2^j-1} \sum_{k'=0}^{p-1} s_{kk'll'}^j \phi_{kl}^j(x) \phi_{k'l'}^j(x'). \tag{A.7}$$

A.2. Multiwavelets

We use piecewise polynomial functions $\psi_0, \ldots, \psi_{p-1}$ as an orthonormal basis for $\mathbf{W}_0$ [12,13],

$$\int_0^1 \psi_i(x)\psi_j(x)\, dx = \delta_{ij}. \tag{A.8}$$

Since $\mathbf{W}_0 \perp \mathbf{V}_0$, the first p moments of all $\psi_0, \ldots, \psi_{p-1}$ vanish:

$$\int_0^1 \psi_i(x) x^i\, dx = 0, \quad i, j = 0, 1, \ldots, p - 1. \tag{A.9}$$

The space $\mathbf{W}_j$ is spanned by $2^j p$ functions obtained from $\psi_0, \ldots, \psi_{p-1}$ by dilation and translation,

$$\psi_{kl}^j(x) = 2^{j/2} \psi_k(2^j x - l), \quad k = 0, \ldots, p - 1,\ l = 0, \ldots, 2^j - 1, \tag{A.10}$$

and supported in the interval $I_{jl} = [2^{-j}l, 2^{-j}(l+1)]$. A function $f(x)$ defined on $[0, 1]$ is represented in the multiwavelet basis on n scales by

$$f(x) = \sum_{k=0}^{p-1} s_{k,0}^0 \phi_k(x) + \sum_{j=0}^{n-1} \sum_{l=0}^{2^j-1} \sum_{k=0}^{p-1} d_{kl}^j \psi_{kl}^j(x) \tag{A.11}$$

with the coefficients d_{kl}^j computed via

$$d_{kl}^j = \int_{2^{-j}l}^{2^{-j}(l+1)} f(x)\psi_{kl}^j(x)\, dx. \tag{A.12}$$

A.3. Two-scale relations

The relation between subspaces, $\mathbf{V}_0 \oplus \mathbf{W}_0 = \mathbf{V}_1$, is expressed via the two-scale difference equations,

$$\phi_k(x) = \sqrt{2} \sum_{k'=0}^{p-1} \left(h_{kk'}^{(0)}\phi_{k'}(2x) + h_{kk'}^{(1)}\phi_{k'}(2x - 1)\right), \quad k = 0, \ldots, p - 1, \tag{A.13}$$

$$\psi_k(x) = \sqrt{2} \sum_{k'=0}^{p-1} \left(g_{kk'}^{(0)}\phi_{k'}(2x) + g_{kk'}^{(1)}\phi_{k'}(2x - 1)\right), \quad k = 0, \ldots, p - 1, \tag{A.14}$$

where the coefficients $h_{ij}^{(0)}$, $h_{ij}^{(1)}$ and $g_{ij}^{(0)}$, $g_{ij}^{(1)}$ depend on the type of polynomial basis used (Legendre or interpolating) and its order p. The matrices of coefficients

$$H^{(0)} = \{h_{kk'}^{(0)}\}, \qquad H^{(1)} = \{h_{kk'}^{(1)}\}, \qquad G^{(0)} = \{g_{kk'}^{(0)}\}, \quad G^{(1)} = \{g_{kk'}^{(1)}\}$$

are the multiwavelet analogues of the quadrature mirror filters in the usual wavelet construction, e.g., [15]. These matrices satisfy a number of important orthogonality relations and we refer to [13] for complete details, including the construction of the H, G matrices themselves. Let us only state how these matrices are used to connect the scaling s_{kl}^j and wavelet d_{kl}^j coefficients on neighboring scales j and $j+1$. The *decomposition procedure* ($j+1 \to j$) is based on

$$s_{kl}^j = \sum_{k'=0}^{p-1} \left(h_{kk'}^{(0)} s_{k,2l}^{j+1} + h_{kk'}^{(1)} s_{k,2l+1}^{j+1} \right), \tag{A.15}$$

$$d_{kl}^j = \sum_{k'=0}^{p-1} \left(g_{kk'}^{(0)} s_{k,2l}^{j+1} + g_{kk'}^{(1)} s_{k,2l+1}^{j+1} \right); \tag{A.16}$$

the *reconstruction* ($j \to j+1$) is based on

$$s_{k,2l}^{j+1} = \sum_{k'=0}^{p-1} \left(h_{kk'}^{(0)} s_{kl}^j + g_{kk'}^{(0)} d_{kl}^j \right), \tag{A.17}$$

$$s_{k,2l+1}^{j+1} = \sum_{k'=0}^{p-1} \left(h_{kk'}^{(1)} s_{kl}^j + g_{kk'}^{(1)} d_{kl}^j \right). \tag{A.18}$$

A.4. Cross-correlation of the scaling functions

For convolution operators we only need to compute integrals with the cross-correlation functions of the scaling functions,

$$\Phi_{ii'}(x) = \int_{-\infty}^{\infty} \phi_i(x+y) \phi_{i'}(y) \, dy. \tag{A.19}$$

Since the support of the scaling functions is restricted to [0, 1], the functions $\Phi_{ii'}$ are zero outside the interval $[-1, 1]$ and are polynomials on $[-1, 0]$ and $[0, 1]$ of degree $i + i' + 1$,

$$\Phi_{ii'}(x) = \begin{cases} \Phi_{ii'}^+(x), & 0 \leqslant x \leqslant 1, \\ \Phi_{ii'}^-(x), & -1 \leqslant x < 0, \\ 0, & 1 < |x|, \end{cases} \tag{A.20}$$

where $i, i' = 0, \ldots, p-1$ and

$$\Phi_{ii'}^+(x) = \int_0^{1-x} \phi_i(x+y) \phi_{i'}(y) \, dy, \qquad \Phi_{ii'}^-(x) = \int_{-x}^1 \phi_i(x+y) \phi_{i'}(y) \, dy. \tag{A.21}$$

We summarize relevant properties of the cross-correlation functions $\Phi_{ii'}$ in

Proposition 3.

(1) *Transposition of indices*: $\Phi_{ii'}(x) = (-1)^{i+i'} \Phi_{i'i}(x)$.
(2) *Relations between Φ^+ and Φ^-*: $\Phi_{i,i'}^-(-x) = (-1)^{i+i'} \Phi_{i,i'}^+(x)$ *for* $0 \leqslant x \leqslant 1$.
(3) *Values at zero*: $\Phi_{ii'}(0) = 0$ *if* $i \neq i'$, *and* $\Phi_{ii}(0) = 1$ *for* $i = 0, \ldots, p-1$.
(4) *Upper bound*: $\max_{x \in [-1,1]} |\Phi_{ii'}(x)| \leqslant 1$ *for* $i, i' = 0, \ldots, p-1$.

(5) *Connection with the Gegenbauer polynomials*:

$\Phi_{00}^+(x) = \frac{1}{2} C_1^{(-1/2)}(2x - 1) + \frac{1}{2}$ and $\Phi_{l0}^+(x) = \frac{1}{2}\sqrt{2l+1} C_{l+1}^{(-1/2)}(2x - 1)$, *for* $l = 1, 2, \ldots,$ *where* $C_{l+1}^{(-1/2)}$ *is the Gegenbauer polynomial.*

(6) *Linear expansion: if* $i' \geqslant i$ *then we have*

$$\Phi_{ii'}^+(x) = \sum_{l=i'-i}^{i'+i} c_{ii'}^l \Phi_{l0}^+(x), \tag{A.22}$$

where

$$c_{ii'}^l = \begin{cases} 4l(l+1) \int_0^1 \Phi_{ii'}^+(x)\Phi_{l0}^+(x)\left(1-(2x-1)^2\right)^{-1} dx, & i' > i, \\ 4l(l+1) \int_0^1 \left(\Phi_{ii}^+(x) - \Phi_{00}^+(x)\right)\Phi_{l0}^+(x)\left(1-(2x-1)^2\right)^{-1} dx, & i' = i, \end{cases} \tag{A.23}$$

for $l \geqslant 1$ *and* $c_{ii'}^0 = \delta_{ii'}.$

(7) *Vanishing moments: we have* $\int_{-1}^1 \Phi_{00}(x) dx = 1$ *and* $\int_{-1}^1 x^k \Phi_{ii'}(x) dx = 0$ *for* $i + i' \geqslant 1$ *and* $0 \leqslant k \leqslant i + i' - 1.$

Proof of these properties can be found in [24].

A.5. Separated representations of radial functions

As an example, consider approximating the function $1/r^\alpha$ by a collection of Gaussians. The number of terms needed for this purpose is mercifully small. We have [23]

Proposition 4. *For any* $\alpha > 0$, $0 < \delta \leqslant 1$, *and* $0 < \epsilon \leqslant \min\left\{\frac{1}{2}, \frac{8}{\alpha}\right\}$, *there exist positive numbers* τ_m *and* w_m *such that*

$$\left| r^{-\alpha} - \sum_{m=1}^M w_m e^{-\tau_m r^2} \right| \leqslant r^{-\alpha}\epsilon, \quad \text{for all } \delta \leqslant r \leqslant 1, \tag{A.24}$$

with

$$M = \log \epsilon^{-1}\left[c_0 + c_1 \log \epsilon^{-1} + c_2 \log \delta^{-1}\right], \tag{A.25}$$

where c_k *are constants that only depend on* α. *For fixed power* α *and accuracy* ϵ, *we have* $M = \mathcal{O}(\log \delta^{-1})$.

The proof of Proposition 4 in [23] is based on using the trapezoidal rule to discretize an integral representation of $1/r^\alpha$. Similar estimates may be obtained for more general radial kernels using their integral representations as in (14).

We note that approximations of the function $1/r$ via sums of Gaussians have been also considered in [30–32].

A.6. Estimates

By selecting appropriate ϵ and δ in the separated representations of a radial kernel K (as in Proposition 4), we obtain a separated approximation for the coefficients

$$t_{ii',jj',kk'}^{j;\ell} = 2^{-3j} \int_{[-1,1]^3} K\left(2^{-j}(\mathbf{x}+\ell)\right)\Phi_{ii'}(x_1)\Phi_{jj'}(x_2)\Phi_{kk'}(x_3) \, d\mathbf{x}. \tag{A.26}$$

Since the number of terms, M, depends logarithmically on ϵ and δ, we achieve any finite accuracy with a very reasonable number of terms. For example, for the Poisson kernel $K(r) = 1/4\pi r$, we have the following estimate [24].

Proposition 5. *For any* $\epsilon > 0$ *and* $0 < \delta \leqslant 1$ *the coefficients* $t_{ii',jj',kk'}^{j;\ell}$ *in* (A.26) *have an approximation with a low separation rank,*

$$r_{ii',jj',kk'}^{j;\ell} = \sum_{m=1}^M w_m F_{ii'}^{m,l_1} F_{jj'}^{m,l_2} F_{kk'}^{m,l_3}, \tag{A.27}$$

such that if $\max_i |l_i| \geqslant 2$, *then*

$$\left| t^{j;\ell}_{ii',jj',kk'} - r^{j;\ell}_{ii',jj',kk'} \right| \leqslant c_0 2^{-2j} \epsilon, \tag{A.28}$$

and if $\max_i |l_i| \leqslant 1$, *then*

$$\left| t^{j;\ell}_{ii',jj',kk'} - r^{j;\ell}_{ii',jj',kk'} \right| \leqslant 2^{-2j} \left(c_1 \delta^2 + c_0 \epsilon \right) \tag{A.29}$$

where ϵ, δ, M, τ_m, w_m, $m = 1, \ldots, M$ *are described in Proposition 4 for* $\alpha = 1$ *and* c_0 *and* c_1 *are (small) constants.*

As described in Section 4, our adaptive algorithm selects only some of the terms, as needed on a given scale for the desired accuracy ϵ.

A.7. Evaluation of integrals with the cross-correlation functions

We need to compute integrals in (18), where the cross-correlation functions $\Phi_{ii'}$ are given in (A.19). We note that using (A.22), it is sufficient to compute

$$F^{j;m,l}_{i0} = \frac{1}{2^j} \int\limits_{-1}^{1} e^{-\tau_m(x+l)^2/4^j} \Phi_{i0}(x) \, dx$$

for $0 \leqslant i \leqslant 2p-1$ rather than p^2 integrals in (18). Using the relation between Φ^- and Φ^+ in Proposition 3, we have

$$\int\limits_{-1}^{0} \Phi^-_{i0}(x) e^{-\tau_m(x+l)^2} \, dx = \int\limits_{0}^{1} \Phi^-_{i0}(-x) e^{-\tau_m(-x+l)^2} \, dx = (-1)^i \int\limits_{0}^{1} \Phi^+_{i0}(x) e^{-\tau_m(x-l)^2} \, dx,$$

so that

$$F^{j;m,l}_{i0} = \int\limits_{0}^{1} \left[e^{-\tau_m(x+l)^2} + (-1)^i e^{-\tau_m(x-l)^2} \right] \Phi^+_{i0}(x) \, dx.$$

References

[1] L. Greengard, V. Rokhlin, A fast algorithm for particle simulations, J. Comput. Phys. 73 (1) (1987) 325–348.
[2] J. Carrier, L. Greengard, V. Rokhlin, A fast adaptive multipole algorithm for particle simulations, SIAM J. Sci. Statist. Comput. 9 (4) (1986); Yale University Technical Report, YALEU/DCS/RR-496.
[3] H. Cheng, L. Greengard, V. Rokhlin, A fast adaptive multipole algorithm in three dimensions, J. Comput. Phys. 155 (2) (1999) 468–498.
[4] G. Beylkin, R. Coifman, V. Rokhlin, Fast wavelet transforms and numerical algorithms. I, Comm. Pure Appl. Math. 44 (2) (1991) 141–183; Yale University Technical Report YALEU/DCS/RR-696, August 1989.
[5] G. Beylkin, R. Cramer, A multiresolution approach to regularization of singular operators and fast summation, SIAM J. Sci. Comput. 24 (1) (2002) 81–117.
[6] G. Beylkin, M.J. Mohlenkamp, Numerical operator calculus in higher dimensions, Proc. Natl. Acad. Sci. USA 99 (16) (2002) 10246–10251.
[7] G. Beylkin, M.J. Mohlenkamp, Algorithms for numerical analysis in high dimensions, SIAM J. Sci. Comput. 26 (6) (2005) 2133–2159.
[8] R. Harrison, G. Fann, T. Yanai, G. Beylkin, Multiresolution quantum chemistry in multiwavelet bases, in: P.M.A. Sloot, et al. (Eds.), Computational Science—ICCS 2003, in: Lecture Notes in Comp. Sci., vol. 2660, Springer, 2003, pp. 103–110.
[9] R. Harrison, G. Fann, T. Yanai, Z. Gan, G. Beylkin, Multiresolution quantum chemistry: Basic theory and initial applications, J. Chem. Phys. 121 (23) (2004) 11587–11598.
[10] T. Yanai, G. Fann, Z. Gan, R. Harrison, G. Beylkin, Multiresolution quantum chemistry: Hartree–Fock exchange, J. Chem. Phys. 121 (14) (2004) 6680–6688.
[11] T. Yanai, G. Fann, Z. Gan, R. Harrison, G. Beylkin, Multiresolution quantum chemistry: Analytic derivatives for Hartree–Fock and density functional theory, J. Chem. Phys. 121 (7) (2004) 2866–2876.
[12] B. Alpert, A class of bases in L^2 for the sparse representation of integral operators, SIAM J. Math. Anal. 24 (1) (1993) 246–262.
[13] B. Alpert, G. Beylkin, D. Gines, L. Vozovoi, Adaptive solution of partial differential equations in multiwavelet bases, J. Comput. Phys. 182 (1) (2002) 149–190.
[14] G. Beylkin, Approximations and fast algorithms, in: A. Laine, M. Unser, A. Aldroubi (Eds.), Wavelets: Applications in Signal and Image Processing IX, in: Proc. SPIE, vol. 4478, SPIE, Bellingham, WA, 2001, pp. 1–9.
[15] I. Daubechies, Ten Lectures on Wavelets, CBMS-NSF Series in Applied Mathematics, SIAM, 1992.

[16] C. Chui, An Introduction to Wavelets, Academic Press, 1992.

[17] F. Ethridge, L. Greengard, A new fast-multipole accelerated Poisson solver in two dimensions, SIAM J. Sci. Comput. 23 (3) (2001) 741–760.

[18] Y. Maday, C. Mavriplis, A.T. Patera, Nonconforming mortar element methods: Application to spectral discretizations, in: Domain Decomposition Methods, Los Angeles, CA, 1988, SIAM, Philadelphia, PA, 1989, pp. 392–418.

[19] G. Anagnostou, Y. Maday, C. Mavriplis, A.T. Patera, On the mortar element method: Generalizations and implementation, in: Third International Symposium on Domain Decomposition Methods for Partial Differential Equations, Houston, TX, 1989, SIAM, Philadelphia, PA, 1990, pp. 157–173.

[20] C. Bernardi, Y. Maday, A.T. Patera, Domain decomposition by the mortar element method, in: Asymptotic and Numerical Methods for Partial Differential Equations with Critical Parameters, Beaune, 1992, in: NATO Adv. Sci. Inst. Ser. C Math. Phys. Sci., vol. 384, Kluwer Academic, Dordrecht, 1993, pp. 269–286.

[21] C. Bernardi, Y. Maday, A.T. Patera, A new nonconforming approach to domain decomposition: The mortar element method, in: Nonlinear Partial Differential Equations and Their Applications. Collège de France Seminar, vol. XI, Paris, 1989–1991, in: Pitman Res. Notes Math. Ser., vol. 299, Longman Sci. Tech., Harlow, 1994, pp. 13–51.

[22] M. Abramowitz, I.A. Stegun, Handbook of Mathematical Functions, ninth ed., Dover Publications, 1970.

[23] G. Beylkin, L. Monzón, On approximation of functions by exponential sums, Appl. Comput. Harmon. Anal. 19 (1) (2005) 17–48.

[24] G. Beylkin, R. Cramer, G. Fann, R. Harrison, Multiresolution separated representations of singular and weakly singular operators, Appl. Comput. Harmon. Anal. 23 (2) (2007) 235–253.

[25] L. Greengard, J. Strain, The fast Gauss transform, SIAM J. Sci. Stat. Comput. 12 (1) (1991) 79–94.

[26] L. Greengard, X. Sun, A new version of the fast Gauss transform, in: Proceedings of the International Congress of Mathematicians, vol. III, Berlin, 1998, pp. 575–584.

[27] A. Brandt, Multi-level adaptive solutions to boundary value problems, Math. Comp. 31 (1977) 333–390.

[28] A. Brandt, Multilevel computations of integral transforms and particle interactions with oscillatory kernels, Comput. Phys. Commun. 65 (1991) 24–38.

[29] M.H. Kalos, Monte Carlo calculations of the ground state of three- and four-body nuclei, Phys. Rev. (2) 128 (1962) 1791–1795.

[30] W. Kutzelnigg, Theory of the expansion of wave functions in a Gaussian basis, Internat. J. Quant. Chem. 51 (1994) 447–463.

[31] D. Braess, Asymptotics for the approximation of wave functions by exponential sums, J. Approx. Theory 83 (1) (1995) 93–103.

[32] D. Braess, W. Hackbusch, Approximation of $1/x$ by exponential sums in $[1, \infty)$, IMA J. Numer. Anal. 25 (4) (2005) 685–697.