Solution of Multiscale Partial Differential Equations Using Wavelets

Stefan Goedecker and Oleg Ivanov

Citation: Computers in Physics **12**, 548 (1998); doi: 10.1063/1.168739

View online: https://doi.org/10.1063/1.168739

View Table of Contents: https://aip.scitation.org/toc/cip/12/6

Published by the American Institute of Physics

AIP Conference Proceedings
FLASH WINTER SALE!
50% OFF ALL PRINT PROCEEDINGS
ENTER CODE 50DEC19 AT CHECKOUT

SOLUTION OF MULTISCALE PARTIAL DIFFERENTIAL EQUATIONS USING WAVELETS

Stefan Goedecker and
Oleg Ivanov

Department Editors:
Ralf Gruber
gruber@sic.epfl.ch
Jacques Rappaz
rappaz@dma.epfl.ch

Wavelets[1] have demonstrated their usefulness in several areas of physics.[2] In this article we show how wavelets can be used to solve partial differential equations that are characterized by widely varying length scales, and which are therefore hardly accessible by other numerical methods. The standard way to solve partial differential equations is to express the solution as a linear combination of so-called basis functions such as plane waves, Gaussians, or finite elements. Wavelets are just another basis set, but their use offers considerable advantages. By adding high-resolution wavelets to the basis set, we can systematically enhance the resolution in certain regions of space where the solution varies rapidly, and this trick can be repeated practically *ad infinitum* without causing numerical instabilities. Even in the case of nonhomogeneous resolution, the numerical effort to solve the differential equation scales strictly linearly with respect to the number of basis functions. This linear scaling can be obtained because the matrix representing the differential operator is sparse and because the number of iterations needed to solve the linear system obtained by the discretization of the differential equation is independent of the highest resolution if an appropriate preconditioning scheme is used. (***Note from the Editor:*** *Additional recent discussion of wavelets can be found in CIP 11:5, 1997, p. 429, and CIP 10:3, 1996, p. 247.)*

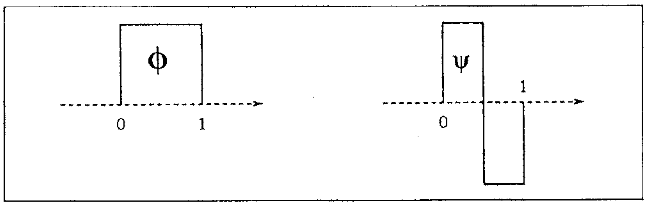

Figure 1. The Haar scaling function ϕ and wavelet ψ.

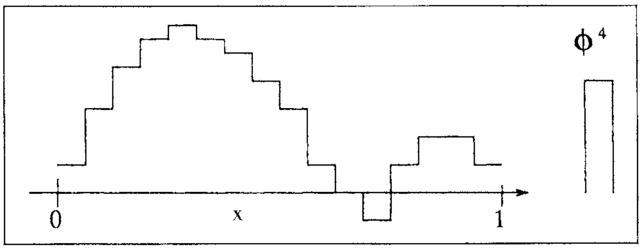

Figure 2. Function f at resolution level 4 together with the scaling function of the same resolution.

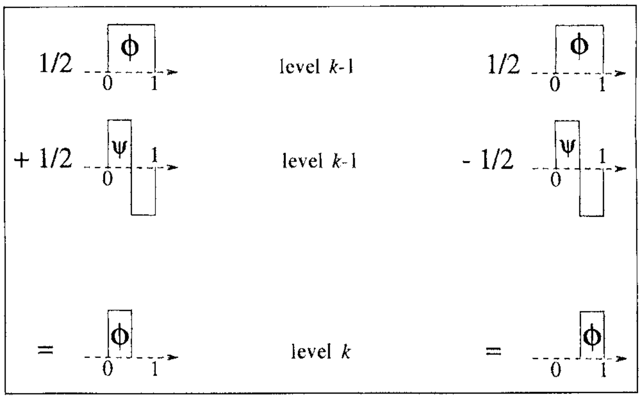

Figure 3. A skinny (level k) scaling function is a linear combination of a fat (level k − 1) scaling function and a fat wavelet.

Stefan Goedecker is the senior scientist at the Max Planck Institute for Solid State Research, Stuttgart, Germany. E-mail:goedeck@prr.mpi-stuttgart.mpg.de

Oleg Ivanov is the senior scientist at the P. N. Lebedev Physical Institute, Moscow, Russia. E-mail: ivanov@td.lpi.ac.ru

S0894-1866(98)00306-X

The Haar wavelet family

To illustrate the concepts of multiresolution analysis,[3] which is the formal theory behind wavelets, we first introduce the Haar wavelet family shown in Fig. 1. As in any other wavelet family, there are two fundamental functions in this family, namely, the scaling function ϕ and the wavelet ψ.

To obtain a basis set at a certain resolution level k, we can use all the integer translations of the scaling function:

$$\phi_i^k(x)=\phi(2^k x-i). \tag{1}$$

Note that with this convention, higher resolution corresponds to larger values of k. Exactly the same scaling and shifting operations can also be applied to the wavelets:

$$\psi_i^k(x)=\psi(2^k x-i). \tag{2}$$

In the following we explain how to use such wavelets in combination with scaling functions as a basis.

In the case of the Haar family, any function that can exactly be represented at any level of resolution is necessarily piecewise constant. One such function is shown in Fig. 2. This function can be written in a scaling-function representation as

$$f=\sum_{i=0}^{15} s_i^4\ \phi_i^4(x), \tag{3}$$

where $s_i^4=f(i/16)$.

Let us now introduce wavelets. Even though the resulting expansion contains both scaling functions and wavelets, we shall refer to such an expansion as a "wavelet representation" so as to distinguish it from the scaling-function representation of Eq. (3). A wavelet representation is possible because a scaling function at resolution level k is always a linear combination of a scaling function and a wavelet at the next coarser level $k-1$, as shown in Fig. 3.

Using this relation depicted in Fig. 3, we can write any linear combination of the two scaling functions $\phi_{2i}^k(x)$ and $\phi_{2i+1}^k(x)$ as a linear combination of $\phi_i^{k-1}(x)$ and $\psi_i^{k-1}(x)$. Hence we can write f as

$$f=\sum_{i=0}^{7} s_i^3\ \phi_i^3(x)+\sum_{i=0}^{7} d_i^3\ \psi_i^3(x)\ . \tag{4}$$

It is easy to verify that the transformation rule for the coefficients is given by

$$s_i^{k-1}=\tfrac{1}{2}s_{2i}^k+\tfrac{1}{2}s_{2i+1}^k\,;\quad d_i^{k-1}=\tfrac{1}{2}s_{2i}^k-\tfrac{1}{2}s_{2i+1}^k\,. \tag{5}$$

Thus, to calculate the expansion coefficients with respect to the scaling functions at the next coarser level, we have to take an average over expansion coefficients at the higher resolution level. Because we must take a weighted sum, these coefficients are denoted by s. To get the expansion coefficients with respect to the wavelet, we must take a weighted difference, and the coefficients are accordingly denoted by d. The wavelet part contains mainly high-frequency components. By doing this transformation, we therefore peel off the highly oscillatory parts of the function.

For any data set whose size is a power of 2, we can now apply this transformation repeatedly. In each step the number of s coefficients will be cut into half, and so we have to stop the procedure as soon as there is only one s coefficient left. Such a series of transformation steps is called a "forward Haar wavelet transform." The resulting wavelet representation of the function in Eq. (3) is then as follows:

$$f=s_0^0\phi_0^0(x)+d_0^0\psi_0^0(x)+\sum_{i=0}^{1} d_i^1\psi_i^1(x)+\sum_{i=0}^{3} d_i^2\psi_i^2(x)+\sum_{i=0}^{7} d_i^3\psi_i^3(x)\ . \tag{6}$$

A wavelet representation is possible because a scaling function at resolution level k is always a linear combination of a scaling function and a wavelet at the next coarser level k −1.

Note that in both cases we need exactly 16 coefficients to represent the function.

By doing a backward wavelet transform, we can go back to the original expansion of Eq. (3). Starting at the lowest resolution level, we have to express each scaling function and wavelet on the coarse level in terms of scaling functions at the finer level. This can be done exactly because wavelet families satisfy the so-called refinement relations depicted in Fig. 4 for the Haar family. It then follows that we have to backtransform the coefficients in the following way:

$$s_{2i}^{k+1}=s_i^k+d_i^k\,;\quad s_{2i+1}^{k+1}=s_i^k-d_i^k\,. \tag{7}$$

We can understand how wavelets are able to provide a compact representation of functions characterized by different length scales by looking at the function in Fig. 5. Since this function varies strongly only in the middle part, many of the high-resolution d coefficients in Eq. (6) are zero and can be discarded. In fact, to represent the function of Fig. 5, only one d coefficient is needed per resolution level. Five coefficients are thus enough to represent the function in-

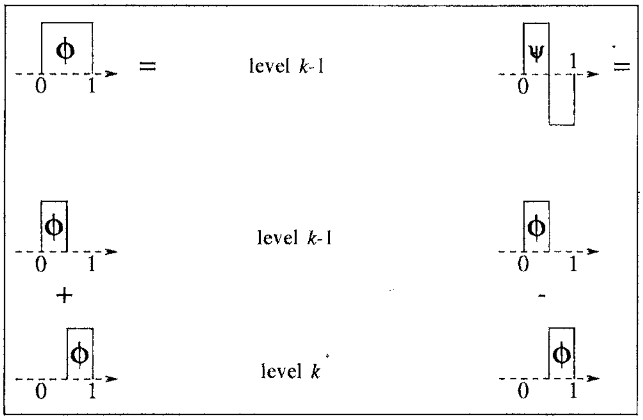

Figure 4. Fat scaling functions and wavelets are linear combinations of skinny scaling functions.

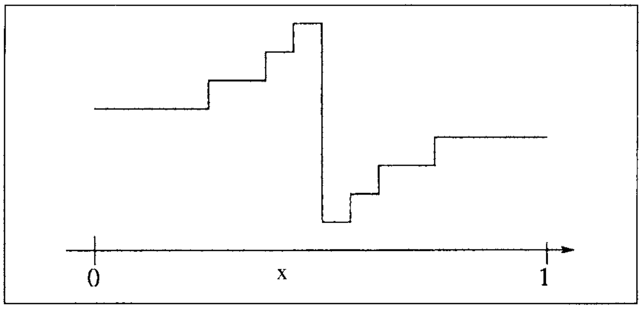

Figure 5. A function requiring nonuniform resolution.

stead of the 16 required by Eq. (3) in its most general form. Compact wavelet representations of this type corresponding to non-uniform resolution will therefore be the basic tool for the solution of multiscale differential equations.

Interpolating wavelets

We next describe a class of wavelets that are highly useful for numerical work, namely, biorthogonal wavelets. A biorthogonal wavelet family of degree m is characterized by four finite filters denoted by h_j, $\tilde{h}_j$, g_j, and $\tilde{g}_j$. A filter is just a short vector that is used in convolutions. The scaling functions and wavelet satisfy the refinement relations

$$\phi(x)=\sum_{j=-m}^{m} h_j\,\phi(2x-j);\quad \psi(x)=\sum_{j=-m}^{m} g_j\,\phi(2x-j), \tag{8}$$

which are a generalization of the relations represented pictorially in Fig. 4 for the Haar family. In addition to the scaling function ϕ and wavelet ψ, there are also the "dual quantities" $\tilde{\phi}$ and $\tilde{\psi}$. Construction of the dual quantities[3,4] will not be discussed in this article. They satisfy the orthogonality relations

$$\int \tilde{\phi}_i^k(x)\,\phi_j^k(x)dx=\delta_{i,j}, \tag{9}$$

$$\int \tilde{\psi}_i^k(x)\,\phi_j^q(x)dx=0\,,\quad k\geqslant q, \tag{10}$$

$$\int \psi_i^k(x)\,\tilde{\phi}_j^q(x)dx=0\,,\quad k\geqslant q, \tag{11}$$

$$\int \psi_i^k(x)\,\tilde{\psi}_j^q(x)dx=\delta_{k,q}\delta_{i,j}\,. \tag{12}$$

The expansion coefficients at different resolution levels are related by the wavelet transform equations. The analysis (forward) transform is given by

$$s_i^{k-1}=\sum_{j=-m}^{m} \tilde{h}_j s_{j+2i}^k,\quad d_i^{k-1}=\sum_{j=-m}^{m} \tilde{g}_j s_{j+2i}^k, \tag{13}$$

and a wavelet synthesis (backward) transform is given by

$$\begin{aligned} s_{2i}^{k+1}&=\sum_{j=-m/2}^{m/2} h_{2j}\,s_{i-j}^k+g_{2j}\,d_{i-j}^k,\\ s_{2i+1}^{k+1}&=\sum_{j=-m/2}^{m/2} h_{2j+1}\,s_{i-j}^k+g_{2j+1}\,d_{i-j}^k. \end{aligned} \tag{14}$$

These two equations are generalizations of Eqs. (5) and (7), which we derived in an intuitive way for the Haar family. We first look at the forward transform given by Eq. (13). The peeling off of the high-frequency components in the forward transform can be illustrated in the following way:

$$\begin{tikzcd} S^4 \arrow[r] \arrow[dr] & S^3 \arrow[r] \arrow[dr] & S^2 \arrow[r] \arrow[dr] & S^1 \arrow[r] \arrow[dr] & S^0 \\ & D^3 & D^2 & D^1 & D^0. \end{tikzcd}$$

By convention the data are arranged in the following order:

original data

$s_0^4\ s_1^4\ s_2^4\ s_3^4\ s_4^4\ s_5^4\ s_6^4\ s_7^4\ s_8^4\ s_9^4\ s_{10}^4 s_{11}^4 s_{12}^4 s_{13}^4 s_{14}^4 s_{15}^4$

$= S^4;$

after first sweep

$s_0^3\ s_1^3\ s_2^3\ s_3^3\ s_4^3\ s_5^3\ s_6^3\ s_7^3\ d_0^3\ d_1^3\ d_2^3\ d_3^3\ d_4^3\ d_5^3\ d_6^3\ d_7^3$

$= S^3,D^3;$

after second sweep

$s_0^2\ s_1^2\ s_2^2\ s_3^2\ d_0^2\ d_1^2\ d_2^2\ d_3^2\ d_0^3\ d_1^3\ d_2^3\ d_3^3\ d_4^3\ d_5^3\ d_6^3\ d_7^3$

$= S^2,D^2,D^3;$

after third sweep

$s_0^1\ s_1^1\ d_0^1\ d_1^1\ d_0^2\ d_1^2\ d_2^2\ d_3^2\ d_0^3\ d_1^3\ d_2^3\ d_3^3\ d_4^3\ d_5^3\ d_6^3\ d_7^3$

$= S^1,D^1,D^2,D^3;$

final data

$s_0^0\ d_0^0\ d_0^1\ d_1^1\ d_0^2\ d_1^2\ d_2^2\ d_3^2\ d_0^3\ d_1^3\ d_2^3\ d_3^3\ d_4^3\ d_5^3\ d_6^3\ d_7^3$

$= S^0,D^0,D^1,D^2,D^3.$

Note that this transformation from the "original data" to the "final data" corresponds exactly to the transformation done in an intuitive way to get from Eq. (3) to Eq. (6). Just as in the case of a fast Fourier transform, we have

$\log_2(n)$ sweeps to do a full transform. However, in the case of the wavelet transform, the active data set (the s coefficients) is cut into half in each sweep. If our filters h and g have length $2m$, the operation count is then given by $2m(n+n/2+n/4+\dots)$. Replacing the finite geometric series by its infinite value, we find that the total operation count is given by $4mn$. A backward transform [Eq. (14)] is accomplished by essentially inverting all the steps of the forward transform.

Because of their smoothness, "interpolating wavelets"[5] are highly useful as basis functions for partial differential equations. In addition, they are also conceptually rather simple. We therefore briefly describe their construction.

The construction of interpolating wavelets is closely connected to the question of how to construct a continuous function $f(x)$ if only its values f_i on a finite number of grid points i are known. One way to do this is by recursive interpolation. In a first step we interpolate the functional values on all the midpoints by using, for instance, the values of two grid points to the right and to the left of the midpoint. These four functional values allow us to construct a third-order polynomial, which we can evaluate at the midpoint. In the next step, we take this new data set, which is now twice as large as the original one, as the input for a new midpoint-interpolation procedure. This can be done recursively *ad infinitum* until we have a quasicontinuous function.

We now show how this interpolation prescription leads to a set of basis functions. Denoting by the Kronecker δ_{i-j} a data set that has a nonzero entry only at the jth position, we can write any initial data set also as a linear combination of such Kronecker data sets: $f_i=\Sigma_j f_j \delta_{i-j}$. Now the whole interpolation procedure is clearly linear; i.e., that is, the sum of two interpolated values of two separate data sets is equal to the interpolated value of the sum of these two data sets. This means that we can instead also take all the Kronecker data sets as the input for separate recursive interpolation procedures to obtain a set of functions $\phi(x-j)$. The final interpolated function is then identical to

$$f(x)=\sum_j f_j \phi(x-j). \qquad (15)$$

If the initial grid values f_i were the functional values of a polynomial of a degree less than four, we will have exactly reconstructed the original function from its values on the grid points. Since any smooth function can locally be well approximated by a polynomial, these functions $\phi(x)$ are good basis functions also for the case in which f is not a polynomial, and we shall use them as scaling functions to construct a wavelet family.

The first construction steps of an interpolating scaling function are shown on top of the left panel in Fig. 6 for the case of linear interpolation. The initial Kronecker data set is denoted by the big dots. The additional data points obtained after the first interpolation step are denoted by medium-size dots, and the additional data points obtained after the second step, by small dots.

Continuing this process *ad infinitum* will then result in the function shown in the left panel of Fig. 6. If a higher-order interpolation scheme is used, the function shown in

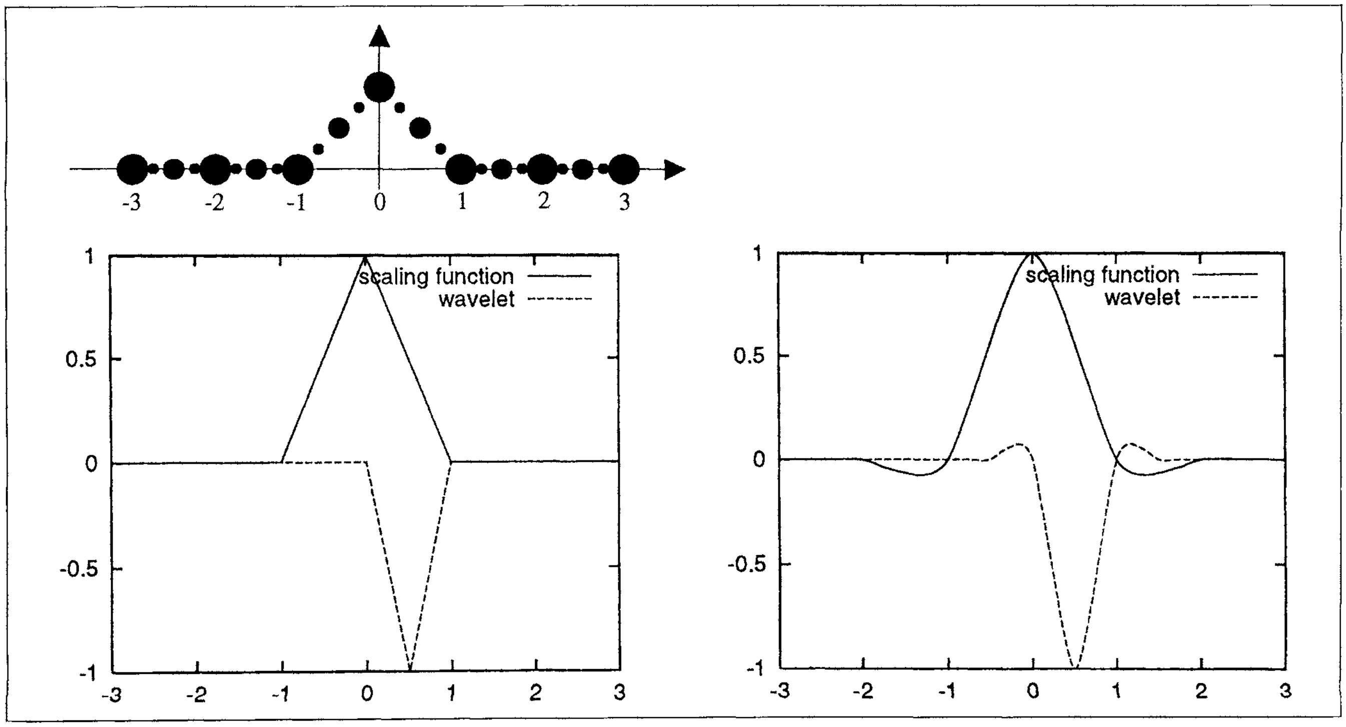

Figure 6. A Kronecker delta interpolated ad infinitum *with linear interpolation (left panel) and a third-order interpolation (right panel).*

the right panel of Fig. 6 is obtained. By a similar procedure,[4] the wavelets associated with the scaling function can be constructed. For the cases of first- and third-order interpolation, they are also shown in Fig. 6.

Wavelets can also be constructed in higher-dimensional spaces.[3] Higher-dimensional wavelets are essentially products of one-dimensional wavelets.

Expanding functions in a wavelet basis

As was demonstrated for the Haar wavelet, there are two possible representations of a function within the framework of wavelet theory. The scaling-function representation [corresponding to Eq. (3)] is given by

$$f(x)=\sum_j s_j^{K\mathrm{max}} \phi_j^{K\mathrm{max}}(x). \tag{16}$$

It follows from the orthogonality relation (9) that the coefficients $s_j^{K\mathrm{max}}$ can be calculated by integration:

$$s_j^K=\int \tilde{\phi}_j^K(x)\, f(x) dx. \tag{17}$$

Once we have a set of coefficients $s_j^{K\mathrm{max}}$, we can use a full forward wavelet transform to obtain the coefficients of the wavelet representation [corresponding to Eq. (6)]:

$$f(x)=\sum_j s_j^{K\mathrm{min}}\ \phi_j^{K\mathrm{min}}(x)+\sum_{K=K\mathrm{min}}^{K\mathrm{max}} \sum_j d_j^K\ \ \psi_j^K(x). \tag{18}$$

Alternatively, we could also directly calculate the d coefficients by integration:

$$d_j^K=\int \tilde{\psi}_j^K(x)\, f(x) dx. \tag{19}$$

Equation (19) again follows from the orthogonality relations (10)–(12).

So we see that if we want to expand a function either in scaling functions or wavelets, at some point we have to perform integrations to calculate the coefficients. For general wavelet families this integration can be difficult and computationally expensive.[6] The interpolating wavelets discussed above are the glorious exception. Since both the dual scaling function and the dual wavelets are delta functions,[4] one or just a few data points suffice to do the integration exactly. We therefore get exactly the same number of coefficients as data points and have an invertible one-to-one mapping between the functional values on the grid and the expansion coefficients. This is even true for nonuniform data sets, for which we necessarily have to calculate the s and d coefficients directly by integration using Eq. (19).

Standard and nonstandard operator forms

We shall now discuss how to apply a differential operator $\mathcal{A}$ such as the Laplacian to a function defined with respect to a wavelet basis set. Matrix×vector multiplications of this kind will be the basic ingredient for the conjugate-gradient techniques used later in this article on the solution of Poisson's equation. Given a vector **b** containing the expansion coefficients of the function, we thus want to calculate the vector **c**, where $\mathbf{c}=\mathbf{A}\mathbf{b}$. The matrix **A** is the Petrov-Galerkin [G. Strong and G. T. Fise, *An Analysis of the Finite Element Method* (Prentice Hall, London, 1973)] approximation to the operator $\mathcal{A}$. Since we shall have different representations of **A**, a double subscript formed from the letters S and D will indicate whether the left and right basis functions used for the calculation of the matrix elements are scaling functions ϕ (indicated by the subscript S) or wavelets ψ (indicated by the subscript D). Superscripts indicate in the usual way the level of resolution. Because of the finite support of all the basis functions, all matrices are banded. We have also assumed that enough derivatives of our wavelet family exist to make the differential operator well defined. This assumption will not hold for the Haar wavelet family introduced for pedagogic reasons, but it will hold, for instance, for interpolating wavelets of sufficiently high degree.

There are two possible ways to apply operators onto functions within wavelet theory. They are called the standard and nonstandard forms.

To derive these two forms, we always first assume that the operator $\mathcal{A}$ is defined with respect to a scaling-function representation [Eq. (16)]; that is,

$$(\mathbf{A}_{SS}^{k+1})_{i,j}=\int \tilde{\phi}_i^{k+1}(x)\mathcal{A}\phi_j^{k+1}(x)\ dx.$$

The matrix×vector multiplication then reads

$$\mathbf{c}_S^{k+1}=\mathbf{A}_{SS}^{k+1}\mathbf{b}_S^{k+1}, \tag{20}$$

where $\mathbf{c}_S^{k+1}$ are the scaling-function coefficients of the output vector **c** and $\mathbf{b}_S^{k+1}$ are the corresponding coefficients of the input vector **b**.

If we want to expand a function either in scaling functions or wavelets, at some point we have to perform integrations to calculate the coefficients.

Since we shall be interested in representations allowing varying resolution, this form is useless for us. Instead, we wish to derive the form of the operator with respect to a wavelet representation [Eq. (18)]. To obtain a wavelet representation, we have to apply a fast wavelet transformation on the input and output vectors and consequently on all the

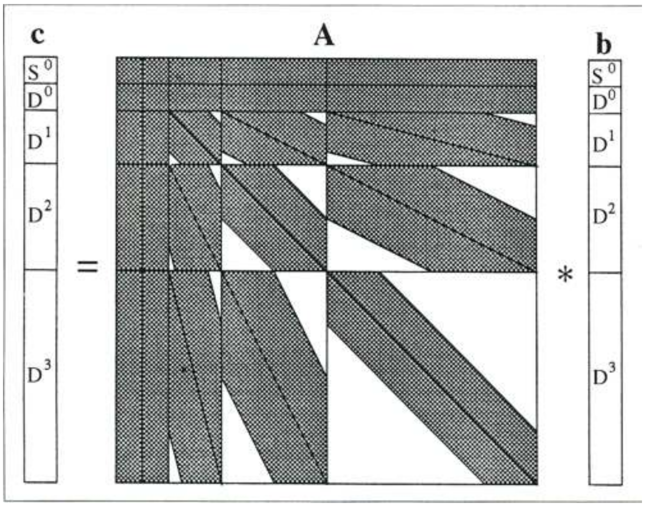

Figure 7. The structure of a matrix in the standard form. The parts containing nonzero entries are shaded.

rows and columns of the matrix. After one step of the wavelet transform, we obtain the following equations:

$$\mathbf{c}_S^k=\mathbf{A}_{SS}^k\mathbf{b}_S^k+\mathbf{A}_{SD}^k\mathbf{b}_D^k\,, \qquad (21)$$

$$\mathbf{c}_D^k=\mathbf{A}_{DS}^k\mathbf{b}_S^k+\mathbf{A}_{DD}^k\mathbf{b}_D^k\,, \qquad (22)$$

where

$$(\mathbf{A}_{SD}^{k+1})_{i,j}=\int\tilde{\phi}_i^{k+1}(x)\mathcal{A}\psi_j^{k+1}(x)\;dx,$$

$$(\mathbf{A}_{DS}^{k+1})_{i,j}=\int\tilde{\psi}_i^{k+1}(x)\mathcal{A}\phi_j^{k+1}(x)\;dx,$$

and

$$(\mathbf{A}_{DD}^{k+1})_{i,j}=\int\tilde{\psi}_i^{k+1}(x)\mathcal{A}\psi_j^{k+1}(x)\;dx.$$

Note that even though there are now two equations, the dimension of the total vectors **b** and **c** is of course the same in Eq. (20) as it is in Eqs. (21) and (22). Applying the further steps of the wavelet transformation to split up all the remaining $\mathbf{A}_{SS}$ parts recursively, we obtain the standard operator form that is visualized in Fig. 7.

Note that there is coupling between all resolution levels. Because of the coupling, it is neccesary to calculate many different types of matrix elements corresponding to all possible products of wavelets and scaling functions at different resolution levels and positions. This complicated structure makes the standard form inefficient for numerical purposes.

Fortunately a second form, the so-called nonstandard form,[7] gives an easier and more efficient representation of the matrix. To derive it, we again assume that initially all our quantities are given in a scaling-function representation [Eq. (20)]. We also do again the first transformation step visualized in Eqs. (21) and (22). To get the nonstandard form, we artificially enlarge the matrix by putting in five blocks of zeroes denoted by $\mathbf{0}^k$:

$$\mathbf{c1}_S^k=\mathbf{A}_{SS}^k\mathbf{b}_S^k+\mathbf{0}^k\mathbf{b}_S^k+\mathbf{0}^k\mathbf{b}_D^k\,, \qquad (23)$$

$$\mathbf{c2}_S^k=\mathbf{0}^k\mathbf{b}_S^k+\mathbf{0}^k\mathbf{b}_S^k+\mathbf{A}_{SD}^k\mathbf{b}_D^k\,, \qquad (24)$$

$$\mathbf{c}_D^k=\mathbf{0}^k\mathbf{b}_S^k+\mathbf{A}_{DS}^k\mathbf{b}_S^k+\mathbf{A}_{DD}^k\mathbf{b}_D^k\,. \qquad (25)$$

Both the input vector **b** and output vector **c** are now larger than necessary and therefore redundant. In the input vector $(\mathbf{b}_S^k,\mathbf{b}_S^k,\mathbf{b}_D^k)$, we have two copies of the $\mathbf{b}_S^k$ part; in the output vector we must add $\mathbf{c1}_S^k$ and $\mathbf{c2}_S^k$ to get back the correct $\mathbf{c}_S^k$ from Eq. (21). We can now recursively apply this two-step procedure, from Eq. (20) to Eqs. (21) and (22) and from these to Eqs. (23)–(25), on the remaining $\mathbf{A}_{SS}$ parts. This procedure produces the nonstandard form that is graphically visualized in Fig. 8.

There are no blocks in this matrix between the different levels, and so they have been completely decoupled. The coupling of different levels enters only through the wavelet transforms that must be performed at the beginning and end of the operator application to generate the redundant input vector and to reduce the redundant output vector to a nonredundant form. To generate the redundant input vector **b** of Fig. 8, we have to generate S^1 from S^0 and D^0, then S^2 from S^1 and D^1, and finally S^3 from S^2 and D^2. The reduction of the output vector **c** is done by splitting up S^3 into S^2 and D^2 and adding this result to the S^2 and D^2 parts, which are the output of the preceeding matrix×vector multiplication. Next this modified S^2 part is split into an S^1 and D^1 part, and again the result is added to the existing contributions. Finally the S^1 part is split and added to the existing S^0 and D^0 parts.

Note also from Fig. 8 that all the nonzero blocks of this nonstandard matrix representation are strictly banded. The application of this matrix to a vector therefore scales linearly.

The relative sizes of the different blocks of the matrix in Fig. 8 are for the case of uniform resolution in which all

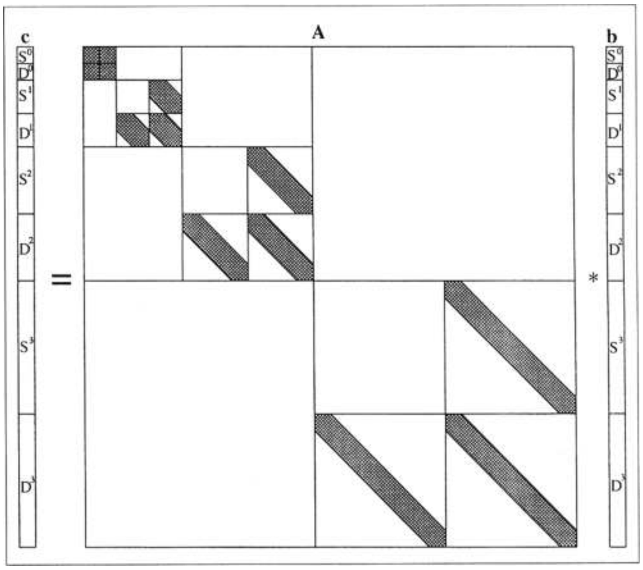

Figure 8. The structure of a matrix in the nonstandard form.

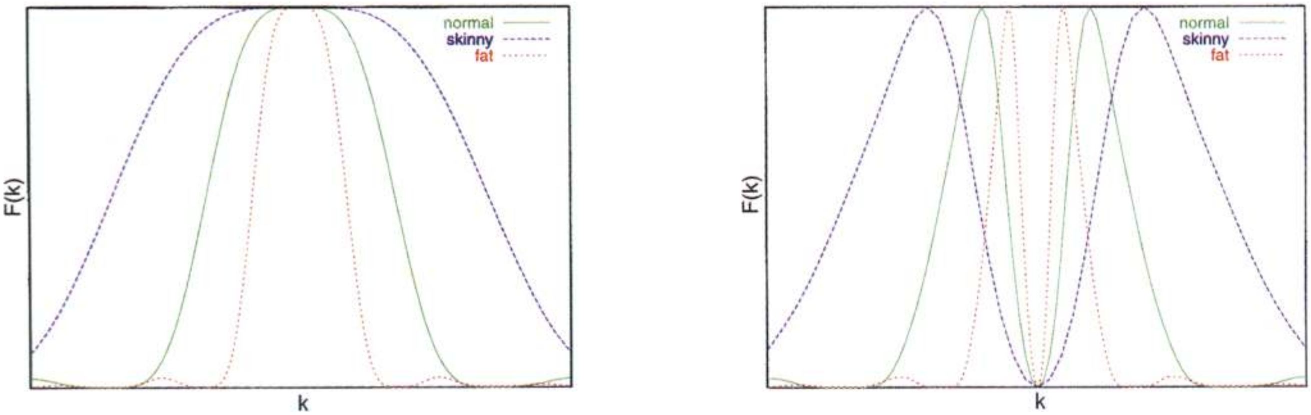

Figure 9. The Fourier spectrum of a fourth-order unlifted interpolating wavelet (left panel) and lifted wavelet (right panel). The spectrum is shown for three wavelets on neighboring resolution levels. There is reasonable frequency separation in the lifted but not in the unlifted case.

the possible D coefficients at the highest resolution level are nonzero. The nonstandard form, however, also allows matrix×vector multiplications to be done with strictly linear scaling in the varying resolution case in which only some of the D coefficients are nonzero. In this case the structure of the matrix in Fig. 8 remains the same; just its proportions change. This aspect of the matrix allowed us to achieve overall linear scaling in the multiscale application that is presented in the next section.

To set up an operator in the nonstandard form we only need a few basic integrals a_i. If the operator represents a differential operator for the second derivative, we need, for instance, only the basic integral

$$a_i=\int \tilde{\phi}(x)\frac{\partial^2}{\partial x^2}\phi(x-i)dx, \tag{26}$$

which can be calculated analytically.[4,8] All the matrix elements can then easily be obtained from a_i by scaling and by using the refinement relations (8).

The nonstandard operator form can be used not only for the application of differential operators but also for other operations. To transform, for instance, from one wavelet family ϕ to another wavelet family Φ, the basic integral becomes

$$a_i=\int \tilde{\Phi}(x)\phi(x-i)dx. \tag{27}$$

Another use is for scalar products, for which the fundamental integral is

$$a_i=\int \phi(x)\phi(x-i)dx. \tag{28}$$

Solving Poisson's equation for the uranium dimer

Biorthogonal wavelets form a natural basis for solving a differential equation in the Petrov–Galerkin sense. The Petrov-Galerkin method has two functional spaces. The first functional space is that of the basis functions and is used to represent the solution. The space of the test functions is used to multiply the differential equation from the left. Integration then yields a linear system of equations, which is the discretized version of the differential equation. In our case the expansion set are the scaling functions and wavelets, whereas the test set are their dual counterparts.

We consider the case of Poisson's equation

$$\nabla^2 v=-4\pi n. \tag{29}$$

We look for a wavelet representation $\mathbf{v}$ of the potential v, having available the corresponding wavelet representation $\mathbf{n}$, which is obtained from the charge density n on a nonuniform grid by the methods described in the section headed ''Expanding functions in a wavelet basis.'' Given an approximate solution vector $\tilde{\mathbf{v}}$, the iterative solution of Poisson's equation consists of three principal steps: calculating the residue vector, preconditioning, and updating the solution.

Calculation of the residue vector. The residue $\mathbf{r}$ is calculated from the definition, $\mathbf{r}=\mathbf{A}\tilde{\mathbf{v}}-\mathbf{n}$, where $\mathbf{A}$ is the Laplace operator with respect to the wavelet basis. Because of the advantages described in the previous section the nonstandard form of the operator, it is used in the calculation of $\mathbf{A}\tilde{\mathbf{v}}$.

Preconditioning. The condition number of the Laplace matrix worsens as high-resolution levels are added. The number of iterations needed to obtain convergence would dramatically increase if we used straightforward iterative methods. It is therefore absolutely necessary to use a preconditioned iterative method so as to obtain a condition number that is independent of the maximal resolution. Preconditioning requires finding an approximate inverse matrix of the Laplace matrix. If the Laplace matrix is strongly diagonally dominant, then the inverse of just the diagonal part (which is again diagonal) will be a good approximate inverse. Since the Laplacian is not diagonally dominant in the nonstandard form, we have to try the standard form for this step.

Whether the Laplace matrix is strongly diagonally dominant in the standard form depends on the kind of wavelet family that is used. In a plane-wave representation, the Laplace matrix is strictly diagonal. If therefore our wavelet family has good frequency-localization properties, the resulting matrix will be strongly diagonally dominant in the standard form. Unfortunately our favorite interpolating wavelets have poor frequency localization, making an iterative solution practically impossible. It is therefore necessary to do the preconditioning step within a related wavelet family, namely, the lifted interpolating wavelets,[4,9] which have much better frequency-localization properties, as shown in Fig. 9.

As discussed above, the transformation into another wavelet family [Eq. (27)] can also be done with the help of the nonstandard operator form. Using lifted interpolating wavelets, we were able to reduce the norm of the residue vector by one order of magnitude with three iterations independent of the maximal resolution.

Updating the solution. The preconditioned residue vector is then used to update the potential within a conjugate-gradient scheme. The process returns to the beginning of the iteration unless convergence has been achieved.

To demonstrate the power of the wavelet method, we solved the standard Poisson equation under the most difficult circumstances we could think of. We calculated the electrostatic potential of a three-dimensional uranium dimer.[10] In this example, we clearly find widely varying length scales. The valence electrons have an extension of 5 atomic units, and the 1*s* core electrons have an extension of 2/100 atomic units. The nucleus itself was represented by a charge distribution with an extension of 1/2000 atomic units. Overall, therefore, the length scales varied by four orders of magnitude. Two centers of increasing resolution (around each nucleus) were needed. In order to have quasiperfect natural boundary conditions, we embedded the molecule in a computational volume of side length 10^4 atomic units. Altogether this necessitated 22 levels of resolution. Even though the potential itself varies by many orders of magnitude, we were able to calculate the solution with typically 7 digits of accuracy. We believe that it would not be possible with any other method to solve this kind of benchmark problem.

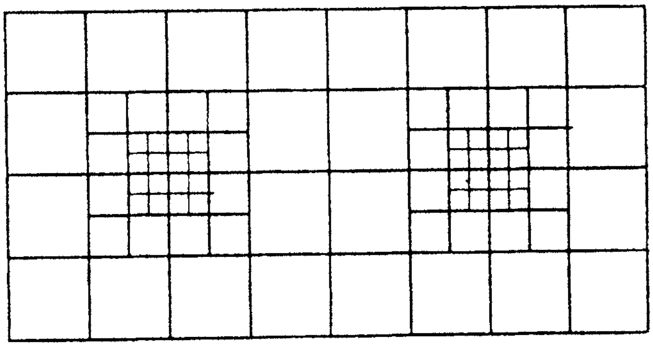

Figure 10. A grid with two centers of increased resolution around the two nuclei. Only 3 of the 22 resolution levels used in the calculation are shown in this projection on a plane.

Initially we had to find the wavelet expansion for a data set on a nonuniform real-space grid structure shown in Fig. 10, which represents the charge density. The resolution needed could be estimated in this example from the known extension and variation of the different atomic shells. Analogous to the one-dimensional case, this expansion can also easily be obtained for higher-dimensional interpolating wavelets, since all the dual functions are related to delta functions. In this case also the mapping from real-space to wavelet representation is invertible, and we could thus get back exactly the same real-space values if we evaluated the wavelet expansion on the grid points.

A similar approach to the solution of the three-dimensional Poisson equation has been proposed by Lippert *et al.*,[11] who, however, used unlifted interpolating wavelets despite their poor frequency-localization properties. In a similar spirit the one-dimensional heat and Burgers equations have been solved[12] as well as the two-dimensional Helmholtz equation.[13]

Conclusions

Wavelets are an elegant and powerful new mathematical theory that has a large impact on many areas. Used as a basis set, wavelets allow us to solve differential equations characterized by widely different length scales. Partial differential equations of this type can be found in many areas of physics and chemistry such as in quantum chemistry.[14]

This article is based on a tutorial-style book[4] describing in detail how to use wavelets for the solution of partial differential equations.

Acknowledgments

The authors thank Ralf Gruber and Jacques Rappaz for their encouragement and their many helpful suggestions on how to improve the presentation.

References

1. Y. Meyer, *Ondelettes et opérateurs* (Hermann, Paris, 1990).
2. J-P. Antoine, Ph. Antoine, and B. Piraux, in *Wavelets in Physics*, edited by J. C. van den Berg (Cambridge University Press, Cambridge, 1998); http://www.mathsoft.com/wavelets.html; http://www.amara.com/current/wavelet.html.
3. I. Daubechies, *Ten Lectures on Wavelets* (SIAM, Philadelphia, 1992).
4. S. Goedecker, *Wavelets and their Application for the Solution of Partial Differential Equations* (ISBN 2-88074-398-2) (Presses Polytechniques et Universitaires Romandes, Lausanne, Switzerland, 1998) (http://ppur.epfl.ch/books.html).
5. G. Deslauriers and S. Dubuc, Constr. Approx. **5**, 49 (1989); similar wavelets have been constructed by M. Teter (private communication).
6. W. Sweldens and R. Piessens, SIAM (Soc. Ind. Appl. Math.) J. Numer. Anal. **31**, 1240 (1994).
7. G. Beylkin, R. Coifman, and V. Rokhlin, Commun. Pure Appl. Math. **44**, 141 (1991).
8. G. Beylkin, SIAM (Soc. Ind. Appl. Math.) J. Numer. Anal. **6**, 1716 (1992).
9. W. Sweldens, Appl. Comput. Harmon. Anal. **3**, 186 (1996).
10. S. Goedecker and O. Ivanov, Solid State Commun. **105**, 665 (1998).
11. R. A. Lippert, T. Arias, and A. Edelman, J. Comput. Phys. **140**, 278 (1998).
12. G. Beylkin and J. Keiser, J. Comput. Phys. **132**, 233 (1997).
13. J. Froehlich and K. Schneider, J. Comput. Phys. **130**, 174 (1997).
14. T. Arias, Rev. Mod. Phys. (to be published).