January 2007

A Wavelet Optimised Method for Financial Derivatives

B. CARTON DE WIART [1,2] AND M.A.H. DEMPSTER,[1,3]
[1] CENTRE FOR FINANCIAL RESEARCH
JUDGE BUSINESS SCHOOL
UNIVERSITY OF CAMBRIDGE
EMAIL: MAHD2@CAM.AC.UK
[2] EQUITY DERIVATIVES, CITIGROUP
EMAIL: BENJAMIN.CARTON@CITIGROUP.COM
[3] CAMBRIDGE SYSTEM ASSOCIATES
EMAIL: MICHAEL.DEMPSTER@CAMBRIDGE-SYSTEMS.COM

We introduce a simple but efficient PDE method that uses interpolation wavelets for their advantages in compression and interpolation in order to define a sparse computational domain. It uses finite difference filters for approximate differentiation, which provide us with a simple and sparse stiffness matrix for the discrete system. Since the method only uses a nodal basis, the application of non-constant terms, boundary conditions and free-boundary conditions is straightforward. We give empirical results for financial products from the equity and fixed income markets in 1, 2 ant 3 dimensions and show a speed-up factor between 2 and 4 with no significant reduction of precision.

1 Introduction

Wavelets are functions that are used in representing data sets or other functions. The elements of a vector space can be represented in terms of a basis and wavelets form a basis for the Hilbert function space of square integrable functions. The most famous basis for this function space is that of Fourier, which represents a function in terms of sines and cosines, in other words, in terms of frequencies. Unfortunately, a function that has only finite support (*i.e.* is non-zero on a small part of its domain) may well have infinite support in the Fourier world. This has no consequence in the case of periodic functions, but it causes trouble for many other applications.

Wavelets use a different approach. The basis functions are well localised both in space and frequency. As a result, they can be used to identify regions where a function encounters high variation. Moreover, using a technique called *thresholding* we can then obtain a compressed version of a function or data set. The most successful application of wavelets is in image compression. The image is treated as a signal to which the wavelet transform is applied. Thresholding then identifies the area of low variation where it lowers the resolution. Wavelets are now the main feature of the JPEG2000 image coding system [8] serving Internet users everyday. But wavelets have other important uses – from speech recognition to *partial differential equations.*

In [14, 15], Dempster *et al.* designed a wavelet PDE method for financial derivatives. Their method solved the PDE using a Galerkin wavelet method in space and the method of lines for time evolution. However, the method can not be adapted to American and Bermudan options efficiently as the application of boundary conditions is difficult in the wavelet space.

We have designed a new method, inspired by the work of Jameson [19, 20, 21] and Walden [31, 32]. This method uses wavelets for what they do best - compress and interpolate - but resorts to finite differences for derivative approximation. This amounts to designing an optimised computational domain where finite differences can be applied. The new method is called *interpolating wavelet optimised finite differences* (IWOFD). The main advantage is that all the computations are done on a nodal basis and the application of non-constant terms, boundary conditions and free-boundaries is straightforward. This allows us to apply the method to a wide range of financial products and to use different stochastic processes to model the underlying asset.

The layout of the paper is as follows. In Section 2 we give a brief introduction to wavelet theory and give the main steps of the algorithm used to generate the sparse computational domain. In Section 3 we review comparable PDE methods and detail the IWOFD algorithm. Numerical results are given in Section 4 and Section 5 concludes.

2 Wavelet Theory

Multiresolution Analysis and Orthogonal Wavelet Basis

The fundamental idea behind wavelet analysis is that the space $L^2(\mathbb{R})$ of square integrable functions can be approximated as a nested hierarchy of subspaces. Formally, from, *e.g.* [10]

Definition 1

A multiresolution analysis (MRA) is a sequence of closed subspaces of $L^2(\mathbb{R})$ *such that:*

1. *The sequence is nested, i.e. for all* $j \in \mathbb{Z}$, $\mathbf{V}_j \subset \mathbf{V}_{j+1}$.
2. *The spaces are related to each other by dyadic scaling, i.e.*

$$f \in \mathbf{V}_j \Leftrightarrow f(2\cdot) \in \mathbf{V}_{j+1} \Leftrightarrow f(2^{-j}\cdot) \in \mathbf{V}_0.$$

3. *The union of the spaces is dense, i.e. for all* $f \in L^2(\mathbb{R})$, $\lim_{j\to+\infty} ||f - P^o_{\mathbf{V}_j} f||_{L^2} = 0$, *where* $P^o_{\mathbf{V}_j}$ *is an orthogonal projection onto* $\mathbf{V}_j$.
4. *The intersection of spaces is zero, i.e.* $\lim_{j\to-\infty} ||P^o_{\mathbf{V}_j} f||_{L^2} = 0$.
5. *There is a function* $\phi \in \mathbf{V}_0$ *such that the family* $\phi(x-k)$, $k \in \mathbb{Z}$ *is a Riesz basis*[1] *of* $\mathbf{V}_0$.

Since $\phi(x) \in \mathbf{V}_0$, $\phi(x/2) \in \mathbf{V}_0$ (by diadic scaling), we have $(h_k) \in \ell^2$ such that

$$\phi(x) = \sum_k h_k \phi(2x-k), \quad (h_k) \in \ell^2(\mathbb{Z}), \tag{1}$$

This functional equation is called the *dilation equation* and its solution the *mother scaling*

[1] A *Riesz basis* in a Hilbert space can be reduced by the Gramm-Schmidt process to an orthonormal basis which defines ℓ^2 as the sequence space of basis coefficients.

function. We term the *filter* $H = \{h_k\}_{k\in\mathbb{Z}}$ the *scaling* filter. Once a basis is found for $\mathbf{V}_0$, we may consider the functions

$$\phi_{j,k}(x) = \phi(2^j x - k), \quad j,k \in \mathbb{Z}, \tag{2}$$

as a basis for $\mathbf{V}_j$. We normalise ϕ so that $\langle \phi(x), \phi(x) \rangle = 1$. To obtain an *orthogonal* scaling function, we naturally impose orthogonality of the corresponding basis so that $\langle \phi(x), \phi(x - n) \rangle = \delta_{0n}$, where δ_{0n} is the Kronecker delta.

For every j, we define the *innovation space* $\mathbf{W}_j$ to be the orthogonal complement of $\mathbf{V}_j$ in $\mathbf{V}_{j+1}$

$$\mathbf{V}_j \oplus \mathbf{W}_j = \mathbf{V}_{j+1}. \tag{3}$$

From the definition of the multiresolution analysis, the innovation spaces satisfy

$$\bigoplus_{i=-\infty}^{\infty} \mathbf{W}_i = L^2(\mathbb{R}). \tag{4}$$

A basis for the *detail space* is given by Theorem 2, see *e.g.* [23], p.110.

Theorem 2
Let $\{\mathbf{V}_m\}$ *be a MRA generated by the orthogonal scaling function* $\phi \in \mathbf{V}_0$ *and define the mother wavelet function* $\psi \in \mathbf{V}_1$ *by*

$$\psi(x) \quad = \quad \sum_{m=-\infty}^{\infty} g_m \phi(2x - m) \tag{5}$$

with $g_k = (-1)^k h_{1-k}$, *then*

$$\{\psi_{j,k}(x) = \psi(2^j x - k) | k \in \mathbb{Z}\} \tag{6}$$

is an orthonormal basis for $\mathbf{W}_j$.

Since (6) defines a basis for $L^2(\mathbb{R})$, for any function $f \in L^2(\mathbb{R})$ there is a family of coefficients $\{d_{j,k}\}$ such that

$$f(x) = \sum_{j\in\mathbb{Z}} \sum_{k\in\mathbb{Z}} d_{j,k} \psi_{j,k}(x) \qquad d_{j,k} = \langle f, \psi_{j,k} \rangle. \tag{7}$$

Orthogonal Wavelet Transform
In practice, computers being finite object manipulators, we need finite sums. We fix two integer levels of coarseness J_1 and J_2 ($J_1 < J_2$) and describe the interplay between these. J_2 is the fine approximation space and we assume there is nothing finer than J_2. Then, by successive transforms, we go down to J_1 and approximate our function $f \in L^2(\mathbb{R})$ as

$$f(x) \approx P_{\mathbf{V}_{J_2}} f(x) = P_{\mathbf{V}_{J_1}} f(x) + \sum_{j=J_1}^{J_2-1} P_{\mathbf{W}_j} f(x). \tag{8}$$

First we need the projection coefficients of a function $f \in L^2(\mathbb{R})$ in $\mathbf{V}_{J_2}$ defined by $s_{J_2,k} := \langle f, \phi_{J_2,k} \rangle$. They can be found in various ways either by a first order approximation (substituting a constant function on the support of $\phi_{J_2,k}$) or a more refined approximation of the integral defining the inner product. Once we have the $s_{J_2,k}$ coefficients, we can then apply the dilation equation (1) to find the $s_{J_2-1,k}$ coefficients as follows

$$\begin{aligned} P_{\mathbf{V}_{J_2-1}} f(x) &= \sum_{k\in\mathbb{Z}} s_{J_2-1,k}\phi_{J_2-1,k}(x) \\ &= \sum_{k\in\mathbb{Z}} \langle f, \phi_{J_2-1,k}\rangle \phi_{J_2-1,k}(x) \\ &= \sum_{k\in\mathbb{Z}}\sum_{m\in\mathbb{Z}} h_m \langle f, \phi_{J_2,m+2k}\rangle \phi_{J_2-1,k}(x) \\ &= \sum_{k\in\mathbb{Z}} \left(\sum_{m\in\mathbb{Z}} h_m s_{J_2,m+2k} \right) \phi_{J_2-1,k}(x). \end{aligned} \tag{9}$$

We see, generalizing for any j, that we have

$$s_{j-1,k} = \sum_m h_m s_{j,m+2k}. \tag{10}$$

Similarly we have

$$d_{j-1,k} = \sum_m g_m s_{j,m+2k}. \tag{11}$$

The inverse transform can be found using the complementarity between $\mathbf{V}_{j-1}$ and $\mathbf{W}_{j-1}$. Using (3) we find the projection on $\mathbf{V}_j$ as

$$\begin{aligned} P_{\mathbf{V}_j} f(x) &= P_{\mathbf{V}_{j-1}} f(x) \oplus P_{\mathbf{W}_{j-1}} f(x) \\ \sum_{k\in\mathbb{Z}} s_{j,k}\phi_{j,k}(x) &= \sum_{l\in\mathbb{Z}} s_{j-1,l}\phi_{j-1,l}(x) + \sum_{m\in\mathbb{Z}} d_{j-1,m}\psi_{j-1,m}(x) \end{aligned} \tag{12}$$

and again, using the scaling equations and comparing coefficients, we obtain

$$s_{j,k} = \sum_{l\in\mathbb{Z}} h_{k-2l} s_{j-1,l} + \sum_{m\in\mathbb{Z}} g_{k-2m} d_{j-1,m}. \tag{13}$$

We can now write the two steps of the algorithm in terms of filter banks as illustrated in Figure 1. The left hand side of Figure 1 shows the forward transform from $\{s_j\}$ to $\{s_{j-1}, d_{j-1}\}$. The signal s_j enters the box, is duplicated and sent to two filters. Application of h followed by downsampling[2] gives $\{s_{j-1}\}$, application of g followed by downsampling gives $\{d_{j-1}\}$. The right hand side of Figure 1 shows the inverse wavelet transform, using $\{s_{j-1}, d_{j-1}\}$ and the mirror filter[3] to rebuild $\{s_j\}$.

The recursive application of (10) and (11) is known as the *cascade algorithm*. Starting with a signal s_{J_2}, we obtain the *scaling coefficients* $s_{J_2-1,k}$ and the *detail coefficients* $d_{J_2-1,k}$.

[2] *Downsampling* $\downarrow$ a function f is defined as $(\downarrow f)(x) := f(2x)$ and *upsampling* $\uparrow$ as $(\uparrow f)(x) := f(x/2)$.

[3] The *mirror filter* of a filter u is defined by $\check{u}(k) := u(-k)$

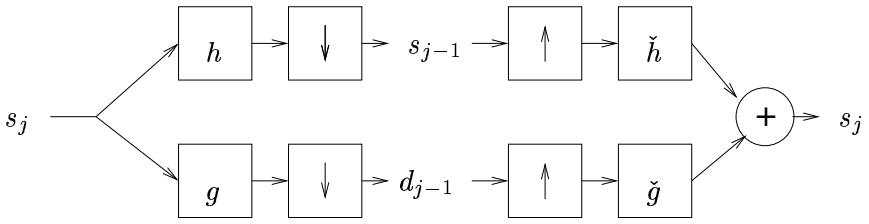

Figure 1: The filter bank algorithm for orthogonal wavelets

The detail coefficients will not be touched anymore. We apply the procedure recursively beginning with the scaling coefficients $s_{J_2-1,k}$ until the required resolution J_1 is obtained. This is illustrated in (14).

$$\begin{array}{ccccccccc} \mathbf{s_{J_2}} & \rightarrow & \mathbf{s_{J_2-1}} & \rightarrow & \mathbf{s_{J_2-2}} & \rightarrow & \ldots & \rightarrow & s_{J_1} \\ & \searrow & d_{J_2-1} & \searrow & d_{J_2-2} & \searrow & \ldots & \searrow & d_{J_1} \end{array} \tag{14}$$

All the coefficients in bold can be recovered using the fast wavelet transform algorithm and therefore need not be stored, while all the other coefficients must be kept. Unfortunately, orthogonality of $\mathbf{W}_j$ and $\mathbf{V}_j$ is a strong restriction. For instance, it can be shown[4], that if ϕ is symmetric and orthogonal, then the generating filter H can only have two nonzero components. Smoothness of the scaling and wavelet functions is also often an issue, with a trade off between smoothness and the length of the filter coefficients of H and G.

Biorthogonal Wavelets

Biorthogonal wavelets were introduced by Cohen *et al.* [11], we will only touch the surface of biorthogonal wavelets and refer to [7] for more detail. The main idea behind biorthogonal wavelets is to use a basis for the computation of the projection parameters different from that of the projection basis. We need the following.

Definition 3

A function $\tilde{\phi} \in L^2(\mathbb{R})$ defined by the dilation equation

$$\tilde{\phi}(x) = \sum_k \tilde{h}_k \tilde{\phi}(2x-k) \quad (\tilde{h}_k) \in \ell^2(\mathbb{Z}) \tag{15}$$

is dual to ϕ if it satisfies

$$\langle \phi(x-j), \tilde{\phi}(x-k) \rangle = \delta_{j,k} \quad j,k \in \mathbb{Z}. \tag{16}$$

From a dual pair of scaling functions we can define the projection by

$$P_{\mathbf{V}_j} f(x) = \sum_{k\in\mathbb{Z}} \langle f(u), \tilde{\phi}_{j,k}(u) \rangle \phi_{j,k}(x), \tag{17}$$

where $\tilde{\phi}$ does not have to be in $\mathbf{V}_0$ and is not uniquely determined for a given ϕ.

The dual space $\tilde{\mathbf{V}}_j$ is defined as the range of the dual projection given by

$$P_{\tilde{\mathbf{V}}_j} f(x) = \sum_{k\in\mathbb{Z}} \langle f(u), \phi_{j,k}(u) \rangle \tilde{\phi}_{j,k}(x). \tag{18}$$

[4]Bultheel, [5], p. 91.

We therefore have two paired hierarchies of approximation subspaces:

$$\{0\} \subset \cdots \subset \mathbf{V}_{j-1} \subset \mathbf{V}_j \subset \mathbf{V}_{j+1} \subset \cdots \subset L^2(\mathbb{R}) \tag{19}$$

$$\{0\} \subset \cdots \subset \tilde{\mathbf{V}}_{j-1} \subset \tilde{\mathbf{V}}_j \subset \tilde{\mathbf{V}}_{j+1} \subset \cdots \subset L^2(\mathbb{R}) \tag{20}$$

with the primal scaling functions $\phi_{j,k}$ being a basis for $\mathbf{V}_j$ and the dual scaling functions $\tilde{\phi}_{j,k}$ a basis for $\tilde{\mathbf{V}}_j$.

We then define two innovation spaces $\mathbf{W}_j$ and $\tilde{\mathbf{W}}_j$ as

$$\mathbf{V}_j \oplus \mathbf{W}_j = \mathbf{V}_{j+1} \qquad \tilde{\mathbf{V}}_j \oplus \tilde{\mathbf{W}}_j = \tilde{\mathbf{V}}_{j+1} \tag{21}$$

with cross orthogonality conditions

$$\tilde{\mathbf{V}}_j \perp \mathbf{W}_j \qquad \mathbf{V}_j \perp \tilde{\mathbf{W}}_j. \tag{22}$$

As with orthogonal wavelets, we can now define the projections of $f \in L^2(\mathbb{R})$ as

$$P_{\mathbf{W}_j} f(x) = \sum_{k \in \mathbb{Z}} \langle f(u), \tilde{\psi}_{j,k}(u) \rangle \psi_{j,k}(x) \tag{23}$$

$$P_{\tilde{\mathbf{W}}_j} f(x) = \sum_{k \in \mathbb{Z}} \langle f(u), \psi_{j,k}(u) \rangle \tilde{\psi}_{j,k}(x). \tag{24}$$

Only the first projection is really used. Taking the fixed resolution levels $J_1 < J_2$ as before, we can define the approximation of f with $s_{J_1,k} := \langle f(u), \tilde{\phi}_{J_1,k}(u) \rangle$ and $d_{j,k} := \langle f(u), \tilde{\psi}_{j,k}(u) \rangle$ as

$$P_{\mathbf{V}_{J_2}} f(x) = \sum_{k \in \mathbb{Z}} s_{J_1,k} \phi_{J_1,k}(x) + \sum_{j=J_1}^{J_2-1} \sum_{k \in \mathbb{Z}} d_{j,k} \psi_{j,k}(x). \tag{25}$$

The whole point of making things seemingly more complex here is to have an inner product $\langle f(u), \tilde{\phi}_{j,k}(u) \rangle$ which is easier to compute than $\langle f(u), \phi_{j,k}(u) \rangle$.

As with the orthogonal wavelet transform, we can write the forward and backward steps of the algorithms in terms of filter banks as illustrated in Figure 2 and see that the dual filters are used for the forward transform, whereas the mirrors of the primal filters are used for the backward transform. Exactly like the orthogonal case, once one forward transform has been applied to get $\{s_{j-1}, d_{j-1}\}$ from s_j, the transform can be applied again to $\{s_{j-1}\}$ in order to obtain $\{s_{j-2}, d_{j-2}\}$. For a fixed resolution level difference, both the forward and the inverse fast biorthogonal wavelet transforms are $O(J_2)$, *i.e.* they are *linear* algorithms.

Donoho's Interpolating Biorthogonal Wavelets

Donoho's biorthogonal wavelets [17] were designed to find a wavelet compression algorithm that would be highly parallelisable, where the computation of any new wavelet coefficient would not require the knowledge of other coefficients at the same level. The mother scaling

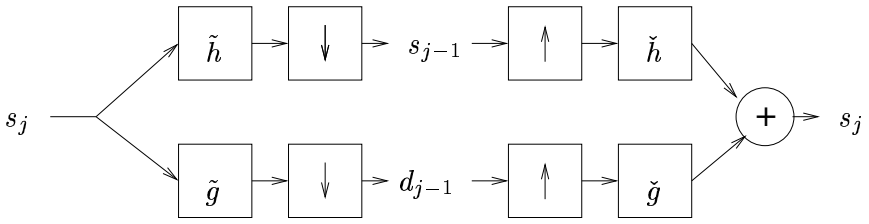

Figure 2: The filter bank algorithm for biorthogonal wavelets

function ϕ is the Deslauriers and Dubuc [16] interpolating function, with filter values given in Table 1. The other bases take the form

$$\begin{aligned} \phi_{j,k}(x) &= \phi(2^j x - k) && (26)\\ \psi_{j,k}(x) &= \phi(2^{j+1} x - 2k - 1) && (27)\\ \tilde{\phi}_{j,k}(x) &= \delta(2^j x - k) \qquad j,k \in Z && (28) \end{aligned}$$

where $\delta(.)$ is the Dirac delta function centered at zero and $x_{j,k} = k/2^j$ are the grid points at the given resolution. No analytical solution of the dual wavelet function $\tilde{\psi}$ is known. The dual filter $\tilde{H}$ is always composed of only one non-zero value, normalised to 1.

The fast wavelet transform algorithm works as in Figure 2, but with extra optimisation in that the filters G and $\tilde{H}$ have only one non-zero component each. Filter values for symmetric wavelets are given in Table 1. The dual filters are found using the flipping rule

$$g_k = (-1)^k \tilde{h}_{1-k} \qquad \tilde{g}_k = (-1)^k h_{1-k}. \tag{29}$$

N	H	$\tilde{H}$
2	$\frac{1}{2}(1, \underline{2}, 1)$	$(\underline{1})$
4	$\frac{1}{16}(-1, 0, 9, \underline{16}, 9, 0, -1)$	$(\underline{1})$
6	$\frac{1}{256}(3, 0, -25, 0, 150, \underline{256}, 150, 0, -25, 0, 3)$	$(\underline{1})$

Table 1: Filter parameters for Donoho's interpolating wavelets

Although interpolating biorthogonal wavelets are an extremely useful tool for data compression, they are not suited for most numerical methods. Technically, they are not a basis anymore since a translation and scaling of the Dirac deltas do not define a proper basis of $L^2(\mathbb{R})$. As a result $P_{\mathbf{W}_j}$ is not a bounded operator, see Cohen [10] p. 57, which affects many convergence results for Galerkin PDE methods.

On the other hand, we can apply any of the wavelet transform algorithms if they allow us to find a compact representation of a data set. The great advantage of Donho's interpolating wavelets is that the scaling coefficients can be very easily computed at all levels independently of other scaling coefficients. The extension of this theory to finite intervals can be found in [25].

Wavelets in Higher Dimensions

The wavelet theory presented so far can be extended to higher dimensions in many different ways, the most common being nested triangulations and tensor products. We will focus on tensor products and assume for simplicity that the resolution in both dimensions is the same. To keep the notation simple, we work with functions of two variables and orthonormal wavelets, the extension to higher dimensions and biorthogonal wavelets does not present any extra difficulties, see Cohen [10]. The two dimensional grid $\mathbf{V}_j$ is defined as

$$\mathbf{V}_j = \mathbf{V}_j^x \otimes \mathbf{V}_j^y. \tag{30}$$

Both $\mathbf{V}_j^x$ and $\mathbf{V}_j^y$ can be decomposed in the usual way

$$\mathbf{V}_j = (\mathbf{V}_{j-1}^x \oplus \mathbf{W}_{j-1}^x) \otimes (\mathbf{V}_{j-1}^y \oplus \mathbf{W}_{j-1}^y). \tag{31}$$

We can use the distributivity of the tensor product to define the two dimensional detail spaces

$$\begin{aligned} \mathbf{V}_j &= (\mathbf{V}_{j-1}^x \otimes \mathbf{V}_{j-1}^y) \oplus (\mathbf{W}_{j-1}^x \otimes \mathbf{V}_{j-1}^y) \oplus (\mathbf{V}_{j-1}^x \otimes \mathbf{W}_{j-1}^y) \oplus (\mathbf{W}_{j-1}^x \otimes \mathbf{W}_{j-1}^y) \\ &:= \mathbf{V}_{j-1} \oplus \mathbf{W}_{j-1}^a \oplus \mathbf{W}_{j-1}^b \oplus \mathbf{W}_{j-1}^c. \end{aligned} \tag{32}$$

We now have three detail spaces $\mathbf{W}_{j-1}^a, \mathbf{W}_{j-1}^b$ and $\mathbf{W}_{j-1}^c$ generated by three families of orthogonal bases $\{\psi_{j,k}^a, \psi_{j,k}^b, \psi_{j,k}^c\}$. The mother functions of the wavelets can be constructed as follows

$$\psi^a(x,y) := \psi(x)\phi(y) \tag{33}$$

$$\psi^b(x,y) := \phi(x)\psi(y) \tag{34}$$

$$\psi^c(x,y) := \psi(x)\psi(y). \tag{35}$$

Translation and dilation of the mother functions occurs in the usual way, *e.g.* for the first mother wavelet:

$$\psi_{j,k,l}^a = \psi^a(2^j x - k, 2^j y - l). \tag{36}$$

A two dimensional function can now be expressed as

$$\begin{aligned} P_{\mathbf{V}_{J_1}} f(x,y) = \sum_{k,l} s_{J_1,k,l}\phi_{J_1,k,l}(x,y) &+ \sum_{j=J_1}^{J_2-1} \sum_{k,l} d_{j,k,l}^a \psi_{j,k}^a(x,y) \\ &+ \sum_{j=J_1}^{J_2-1} \sum_{k,l} d_{j,k,l}^b \psi_{j,k,l}^b(x,y) \\ &+ \sum_{j=J_1}^{J_2-1} \sum_{k,l} d_{j,k,l}^c \psi_{j,k,l}^c(x,y). \end{aligned} \tag{37}$$

Although an extension of the wavelet transform to higher dimensions is relatively easy, methods to solve partial differential equations are often difficult to extend to higher dimensions due to the extra complexity of the geometry.

Thresholding and Sparse Grids

The first, and so far most successful, application of wavelets has been in data compression through *thresholding*. In short, one can delete all the detail coefficients $d_{k,j}$ below a given threshold in the detail space and compress the resulting data. If a large enough proportion of the coefficients are small, this leads to an effective data compression algorithm. In numerical analysis, the embodiment of data compression is a nonuniform grid. Fortunately, grid adaptation is a natural extension of wavelets.

Although $s_{j,k,l}, d^a_{j,k,l}, d^b_{j,k,l}$ and $d^c_{j,k,l}$ are coefficients of the basis functions, they are directly associated with the value of f at some given grid point. A result of the use of Donoho's interpolating wavelets is that

$$s_{j,k,l} = \langle f(x,y), \phi_{j,k,l}(x,y)\rangle = \langle f(x), \delta(2^j x - k)\delta(2^j y - l)\rangle = f(2^{-j}k, 2^{-j}l). \tag{38}$$

Therefore, the coefficients $s_{j,k,l}$ are actually values of f on the grid

$$\Omega^s_j := \{(2^{-j}k, 2^{-j}l) : k,l \in [0, 2^j]\}. \tag{39}$$

In the same vein, the coefficients $d^a_{j,k,l}$ can be associated with the grid

$$\Omega^a_j := \{(2^{-j}k, 2^{-j}(l+1)) : k \in [0, 2^j], l \in [0, 2^j - 1]\}, \tag{40}$$

While $d^a_{j,k,l}$ is *not* the value $f(2^{-j}k, 2^{-j}(l+1))$ of the function, it is the *detail coefficient* at that location. In other words, it is the difference between $f(2^{-j}k, 2^{-j}(l+1))$ and the interpolation of f at neighbouring points. Therefore, if $d^a_{j,k,l}$ is small, the function f can be expected to be smooth at $(x,y) = (2^{-j}k, 2^{-j}(l+1))$.

Similarly, we can define Ω^b_j and Ω^c_j associated with $d^b_{j,k,l}$ and $d^c_{j,k,l}$ as follows

$$\Omega^b_j := \{(2^{-j}(k+1), 2^{-j}l) : k \in [0, 2^j - 1]; l \in [0, 2^j]\} \tag{41}$$

$$\Omega^c_j := \{(2^{-j}(k+1), 2^{-j}(l+1)) : k \in [0, 2^j - 1]; l \in [0, 2^j - 1]\}. \tag{42}$$

Since we have

$$\mathbf{V}_j = \mathbf{V}_{j-1} \oplus \mathbf{W}^a_{j-1} \oplus \mathbf{W}^b_{j-1} \oplus \mathbf{W}^c_{j-1}, \tag{43}$$

the multiresolution analysis on the grid can be read as

$$\Omega^s_j = \Omega^s_{j-1} \cup \Omega^a_{j-1} \cup \Omega^b_{j-1} \cup \Omega^c_{j-1}. \tag{44}$$

It is clear that we can write the fine grid $\Omega^s_{J_2}$ as the union of the coarse grid $\Omega^s_{J_1}$ and the detail grids $(J_1 < J_2)$ as

$$\Omega^s_{J_2} = \Omega^s_{J_1} \cup \Big(\bigcup_{j=J_1}^{J_2-1} (\Omega^a_j \cup \Omega^b_j \cup \Omega^c_j)\Big). \tag{45}$$

This can be used to design a recursive algorithm which selects grid points in high gradient regions while deleting the others.

First we include in the grid all the coefficients in the coarse grid $\Omega^s_{J_1}$. We then look at the detail coefficients at level J_1, *i.e.* points in $\Omega^a_{J_1}, \Omega^b_{J_1}$ and $\Omega^c_{J_1}$. If the sum of the absolute value of the three detail coefficients with the same indices is greater than a given threshold value $\epsilon > 0$, we add the trio to the grid. We define the set of such points as

$$\begin{aligned}
\Upsilon^{0,a}_j &:= \{(2^{-j}k, 2^{-j}(l+1)) \in \Omega^a_j : \quad ||d^a_{j,k,l}|| + ||d^b_{j,k,l}|| + ||d^c_{j,k,l}|| > \epsilon\} \\
\Upsilon^{0,b}_j &:= \{(2^{-j}(k+1), 2^{-j}l) \in \Omega^b_j : \quad ||d^a_{j,k,l}|| + ||d^b_{j,k,l}|| + ||d^c_{j,k,l}|| > \epsilon\} \\
\Upsilon^{0,c}_j &:= \{(2^{-j}(k+1), 2^{-j}(l+1)) \in \Omega^c_j : \quad ||d^a_{j,k,l}|| + ||d^b_{j,k,l}|| + ||d^c_{j,k,l}|| > \epsilon\}
\end{aligned}$$

and their union as

$$\Upsilon^0_j := \Upsilon^{0,a}_j \cup \Upsilon^{0,b}_j \cup \Upsilon^{0,c}_j \tag{46}$$

which is the grid of relevant detail points at level j.

At this level, we also add M extra points in the immediate neighbourhood of the selected points in Υ^0_j. This will allow us to renew the grid less often, saving computational work. Following Vasiliev and Bowman [30] we call these latter points *type I* points. We define the set of such points as

$$\begin{aligned}
\Upsilon^1_j := \{(2^{-j}k, 2^{-j}l) \quad : \quad & \text{such that } \exists o, p \in [-M, ..., M] \\
& \text{with } op = 0 \text{ and } (2^{-j}(k+o), 2^{-j}(l+p)) \in \Upsilon^0_j\}.
\end{aligned}$$

We can now define all the points added at level j as

$$\Upsilon_j := \Upsilon^0_j \cup \Upsilon^1_j. \tag{47}$$

Adding extra coefficients at the edges of a sparse grid is often called *smear* in the wavelet literature.

Repeating the above procedure at each level $j \in [J_1, ..., J_2 - 1]$, we get the *sparse grid* at maximum resolution J_2 as

$$\Upsilon_{J_2} := \Omega^s_{J_1} \cup \Big(\bigcup_{j=J_1}^{J_2-1} \Upsilon_j \Big). \tag{48}$$

Figure 3 shows the sparse grid defined by a unitary impulse, with $J_2 := 6, J_1 := 4, \epsilon := 10^{-4}$. First the coarse grid Ω_{J_1} is set out, then the detail points with norm above ϵ are added, finally the neighbouring or *type I* points, here with $M = 2$ extra points, are included.

A useful capability would be to perform the wavelet transform on the sparse grid. To do so we need to ensure that all the coefficients needed for the transform belong to the grid. The easiest way to do this is to perform the transform on the sparse grid by red-marking all the points that are needed and adding them to the original sparse grid. A nice property of Donoho's interpolating wavelets is that all the points needed to reconstruct points in Ω^c_j belong to Ω^b_j and Ω^a_j. All the points needed to reconstruct points in Ω^b_j or Ω^a_j belong to Ω^s_{j-1}. One therefore avoids *contamination*, *i.e.* adding neighbouring points *ad infinitum*. In detail

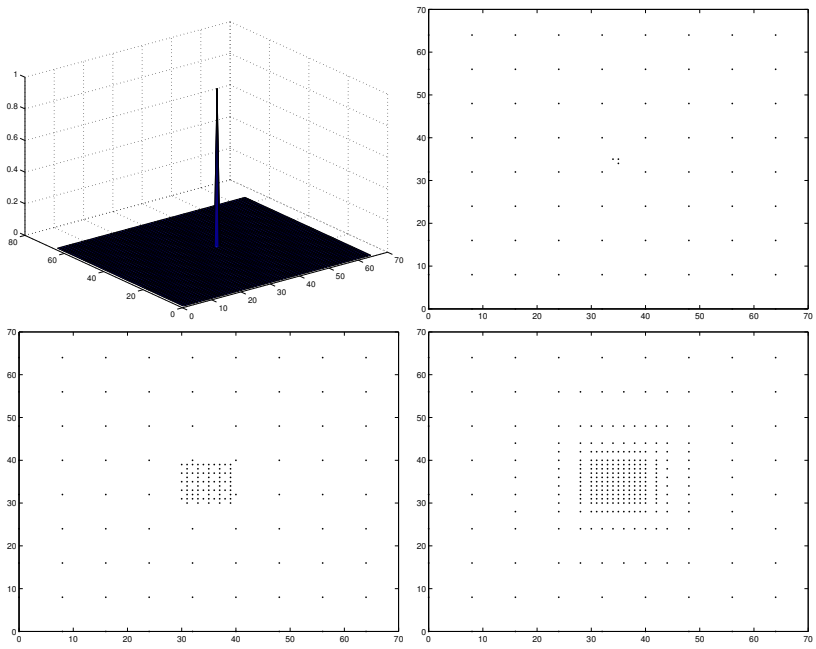

Figure 3: Unitary impulse (upper left), coarse grid with detail coefficients added (upper right), and coarse grid with *type I* points added (left). Sparse grid augmented with *type II* points for the wavelet transform (right)

the algorithm is as follows.

We start at the finest level of detail $J_2 - 1$.

We first inspect all the points in $\Upsilon_j \cap \Omega_j^c$ and red-mark those in $\Omega_j^a \cup \Omega_j^b$ which we need to perform the inverse transform; we call them Υ_j^2. The number of points added depends on the support of the scaling function. If the scaling function is of order N, we will add the N even indexed neighbouring points in each direction and we can write

$$\Upsilon_j^2 = \{(2^{-j}k, 2^{-j}l) \quad : \quad \text{such that } \exists o, p \in [-(N-1), ..., (N-1)] \\ \text{with } op = 0 \text{ and } (2^{-j}(k+o), 2^{-j}(l+p)) \in \Upsilon_j \cap \Omega_j^c\}.$$

We then inspect all the points in $(\Upsilon_j \cup \Upsilon_j^2) \cap (\Omega_j^a \cup \Omega_j^b)$ and red-mark the points in Ω_{j-1}^s we need and call them Υ_{j-1}^3. Once again, we can write

$$\Upsilon_{j-1}^3 = \{(2^{-(j-1)}k, 2^{-(j-1)}l) \quad : \quad \text{such that } \exists o, p \in [-(N-1), ..., (N-1)] \text{ with } op = 0 \\ \text{and } (2^{-j}(2k+o), 2^{-j}(2l+p)) \in (\Upsilon_j \cup \Upsilon_j^2) \cap (\Omega_j^a \cup \Omega_j^b)\}.$$

We update the original sparse grid with the detail coefficients necessary at level j to give

$$\Upsilon_j := \Upsilon_j \cup \Upsilon_j^2, \tag{49}$$

and the next grid with the scaling coefficients we know we will need at level $j-1$ to give similarly

$$\Upsilon_{j-1} := \Upsilon_{j-1} \cup \Upsilon_{j-1}^3. \tag{50}$$

We apply the same procedure recursively until we reach Ω_{J_1}. Nothing needs to be done at this level since Ω_{J_1} is included in the sparse grid and the algorithm ends.

We call the resulting sparse grid the *augmented sparse grid* and the points that have been added in this procedure *type II* points. Cohen [9] calls the addition of such points a *tree structure*. The final grid is denoted by $\Upsilon := \Upsilon_{J_2}$ and the number of nodes in Υ is denoted by $\mathcal{N}_\epsilon$.

Figure 3 shows the effect of this algorithm on the unitary impulse. We see that it has only local consequences. We can also observe the presence of *dangling points*, points that do not have the necessary surrounding points to apply a finite differences filter. The method to deal with such points will be explained later. We obtain a sparse grid which is refined in regions of high gradient, with a few extra points added next to the boundary of such regions, on which we can perform the forward and inverse wavelet transforms.

3 IWOFD A Wavelet Optimized PDE Method

Wavelets in numerical analysis show promising results in three domains: namely operator preconditioning, where discretised elliptic operators generate large stiffness matrices; adaptive approximation, where different parts of a function's domain show different degree of curvature; and sparse representation of large full matrices arising in the discretisation of integral operators. The first application can be linked to the *multigrid* approach, in which the stiffness matrix is successively smoothed, but not solved, at different levels of discretisation. The second domain is related to *mesh refinement* in finite element methods, although it works in the opposite direction. While a mesh refinement starts with a coarse mesh and adds points where they are needed, an adaptive approximation using wavelets starts with a dense mesh and removes useless nodes. The last application is related to *panel clustering* algorithms for integral equations.

Problems of Interest

Mathematical finance provides us with classical time dependent *initial*[5] *value* problems of *hyperbolic* type

$$\begin{cases} u_t = Pu & \forall (t,x) \in [0,T] \times \Omega \\ u(x,0) = \Lambda(x) & \forall x \in \Omega \end{cases} \tag{51}$$

where P is a *partial differential operator* defined on Ω. Once discretised it is called the *stiffness matrix*. In most real life problems, P is not constant in space and time (due to local volatility and non-constant interest rate). The solution vector u tends to possess a smooth gradient in most parts of the domain with sharp changes in localised parts. We will use a wavelet method to discretise the partial differential operator P, which will leave us with an ordinary differential equation that we will tackle using the *method of lines*. In some cases,

[5]The original problem works backwards in time, so we have made a change of variable to make it forwards.

when the option allows *early exercise*, the problems become a *free boundary* problem with complementarity conditions to be applied at all times. The extra complexity added by the free boundary condition requires special algorithms which most wavelet methods cannot incorporate efficiently.

Related Methods

In most wavelet PDE methods in numerical analysis one looks for a solution of the type $u_h \approx u \in V$ with $u_h := \sum_{i=1}^{n} U_i \chi_i$, where the χ_i are basis functions $\{\phi_{J_2,k}, \psi_{j,k}\}$ of V, by solving the weak form

$$\begin{cases} \langle \frac{\partial u_h}{\partial t} - P u_h, \chi' \rangle = 0 & \forall \chi' \in V' \quad \forall (t,x) \in [0,T] \times \Omega \\ u_h(x,0) = \Lambda_h(x) & \forall x \in \Omega \end{cases} \tag{52}$$

and the χ' are known as *test functions*.

Wavelet Galerkin methods [1, 15, 25] solve equation (52) using the basis functions of V as test functions. The most striking results are when solving an integro-partial differential equation (IPDE). The integral term results in a full operator and therefore a full stiffness matrix. It can easily be shown that this matrix can be compressed with great efficiency speeding up all the linear algebra involved in inverting the stiffness matrix by a significant factor; see Matache *et al.* [24]. Another advantage of Galerkin methods lies in the properties of the stiffness matrix. Thanks to the wavelet construction, one can bound the *condition number* (the product of the norm of a matrix and its inverse) of the resulting stiffness matrix; see, *e.g.*, [10]. The consequence is that the numerical methods used to invert the stiffness matrix converge more rapidly, independent of the step size in the discretisation. Another advantage is that multilevel methods naturally lead to multigrid preconditioning, where one successively smooths but does not completely solve the PDE at different levels (discretisation steps); see [13].

The main disadvantage is that computing the stiffness matrix is an expensive task, which is bearable if one only needs to perform it once, *e.g.* for a PDE with constant coefficients, but extremely expensive if it needs to be done more often. Another disadvantage is that even if the basis functions are highly localised, the stiffness matrix is much less sparse than it would be with a plain finite differences method, due to the fact that cross products between bases at different levels are not zero. A last, and non-negligible inconvenience in our case, is that free-boundaries are extremely expensive to apply. One cannot compare functions in the wavelet basis. In other words, if we know the coefficient values for u and Λ, there is no way to compare them unless we invert them back to nodal values. Therefore, the solution needs to be continuously transformed back and forth to test the free boundary conditions.

Collocation methods [6, 29] generally involve operations on nodal values using Dirac functions centred at the collocation points in the original domain as test functions when solving (52). A great advantage of this method is that it naturally reverts to nodal values at every time step, making the application of boundary conditions, non-constant coefficients and free boundaries relatively easy. Adaptivity also comes naturally to those methods. This is done by thresholding the transformed solution to know where to increase the number of collocation points

and thereby increase precision. On the other hand, a disadvantage is that one needs to transform back and forth between nodal values and wavelet coefficients everytime the differential operator is applied. This overhead is expensive if the support of the wavelets is large and unnecessary if the collocation points of importance do not move. Another related method is the *filter bank* method [31, 32] that uses the wavelets (or filter banks, a more general family of transforms) to define the sparse domain and then applies a finite differences filter at each level.

Wavelet Optimised Finite Differences Methods

Since interpolating wavelets were designed in order to reproduce a polynomial perfectly up to a given order it seems natural to use polynomials for differentiation. Finite differences methods are constructed from an underlying interpolation polynomial that is differentiated to obtain the appropriate filter (or from a Taylor expansion of the function, which gives the same result). The combination of the two is finite differences on a non uniform grid generated by a wavelet "shock detector".

Jameson introduced the *wavelet optimised finite difference method* in [20, 19, 22]. WOFD uses wavelets for what they do well – compress a signal – but uses finite difference filters for differentiation. The idea when solving an evolution equation is simple: compute the wavelet transform of the initial conditions, threshold to define an optimal grid and then compute the derivative using finite differences. Since the grid is not regular, one needs to design a local differentiation operator that takes into account the situation of each node, but it is a price worth paying if the number of nodes can be reduced significantly.

A great advantage of this method is that the domain does not need to be updated every time one computes the derivative of the function, which happens frequently when one tries to invert the stiffness matrix using iterative methods. Furthermore, the domain does not need to be updated at every time step. As with collocation methods, all the computations are done with nodal values and, since we only use the wavelet transform every so often to update the grid, we therefore have no problem with boundary conditions, non-constant coefficients and free boundary conditions.

Unfortunately, WOFD uses Daubechies wavelets thats have a strong interrelation between subspace levels. As a result, contamination is observed and the grid fills in much faster than necessary. Furthermore, for some irregularities, the grid fills completely even though the irregularity is highly localised. This does not occur with Donoho's interpolating wavelets (although these introduce other problems such as dangling points).

Finite Differences on Irregular Grids

The easiest way to look at finite differences on a non uniform grid is to locally interpolate the function by a polynomial using a Lagrange formula and differentiate the polynomial to find the weights of the finite difference filter. We approximate the function f of a real argument using the points $x_0, ..., x_n$, which do not have to be evenly spaced, with the formula

$$p_n(x) = \sum_{k=0}^{n} f(x_k) L_k(x) \in \mathcal{P}^n, \tag{53}$$

where the $L_j(x)$ are the well known Lagrange interpolation polynomials

$$L_j(x) = \frac{\prod_{k=0,k\neq j}^{n}(x - x_k)}{\prod_{k=0,k\neq j}^{n}(x_j - x_k)}. \tag{54}$$

As a result, $L_j(x_k) = \delta_{jk}$, which implies $p_n(x_k) = f(x_k)$. We can then find the finite differences approximation of the derivative of f at point x_i using $f(x_0), ..., f(x_n)$ by differentiating the relevant interpolation polynomial as

$$f'(x_i) \approx p'_n(x_i) = \sum_{k=0}^{n} f(x_k)L'_k(x_i) \in \mathcal{P}^{n-1}. \tag{55}$$

For the case of a second order interpolation and the second derivative, differentiation gives

$$p''_2(x_i) = \sum_{k=i-1}^{i+1} f(x_k)L''_k \tag{56}$$

with $L''_k = \frac{2}{\prod_{k=0,k\neq i}^{n}(x_j - x_k)}$. If the points are evenly spaced with distance h, we find $[L''_{i-1}, L''_i, L''_{i+1}] = \frac{1}{h^2}[1, -2, 1]$, which is the classical finite differences filter. These concepts are extended to partial derivatives of a function of a vector argument in a natural way.

Dangling points

The application of finite differences is extremely easy in one dimension, since the points $x_0, ..., x_n$, even if unevenly spaced, are always ordered and easy to find. When the dimension increases, such a situation cannot be assumed since the uneven grid generated by the thresholding of the wavelet transform will create so-called dangling points. It is therefore not convenient to apply a finite differences operator on these points.
We define the set of *regular points* Ξ on the grid Γ as

$$\Xi := \{x_{i,j} \in \Gamma : \exists k, l, m, n \in \mathbb{N} \text{ s.t. } x_{i+k,j},\ x_{i-l,j},\ x_{i,j+m},\ x_{i,j-n} \in \Gamma\}. \tag{57}$$

If $x_{i,j} \in \Gamma \setminus \Xi$ is a *dangling point. Interpolating wavelet optimised finite differences* can use the convenient properties of the wavelet transform to interpolate these points instead of updating them. Since the proportion of such points is very small, and since they are located at the boundaries of the regions containing large gradients (in other words, far away from the "interesting" regions) this will not tend to affect convergence. The main advantage is that it allows us to easily design the discretisation of partial differential operators.

Method of Lines

Since we deal with evolution equations, we have two parts to the problem, the first problem is the partial differential operator P. Most wavelet methods discretise the operator P to provide us with a stiffness matrix A. We are then left with the following system of ODEs

$$\begin{cases} u_t = Au & \forall t \in [0, T] \\ u(0) = \Lambda, \end{cases} \tag{58}$$

where $A \in \mathbb{R}^{\mathcal{N}_\epsilon(t)\times\mathcal{N}_\epsilon(t)}$ is a matrix, $u \in \mathbb{R}^{\mathcal{N}_\epsilon(t)}$ the solution vector, $\Lambda \in \mathbb{R}^{\mathcal{N}_\epsilon(t)}$ the appropriately discretised initial condition and $\mathcal{N}_\epsilon(t)$ the grid size at time t. For clarity, we assume

we have inverted the time and solve the equation forwards and that the boundary conditions have been included into the stiffness matrix. We have implemented several standard ODE solvers such as Crank-Nicolson (with different solvers for the linear algebra), backward differentiation and Dufort-Frankel (see Section 4).

IWOFD Algorithm

We now have all the tools to define the *interpolating wavelet optimised finite difference* (IWOFD) algorithm. IWOFD can be seen as a version of the WOFD algorithm that uses Donoho's interpolating wavelet for grid selection and the interpolating wavelets again for interpolation of the function values at the dangling points. The main steps are given in Algorithm 1.

Algorithm 1 IWOFD

```
1:  t = 0
2:  Δt = T/N
3:  u = Λ
4:  for i = 0 to N − 1 do
5:    if i%RENEW_GRID = 0 then
6:      Decompress u
7:      Interpolate missing values
8:      Compute wavelet transform
9:      Define sparse Grid Γ
10:     Identify dangling points Γ \ Ξ
11:     Compress u
12:     Compute stiffness matrix A on Ξ
13:   end if
14:   Solve u_t = Au, t ∈ [iΔt, (i + 1)Δt] with the method of lines
15: end for
16: Decompress u
17: Interpolate missing values on u
```

The first step is to initialise and create the operator. We transform the (initial) solution into wavelet space and threshold, adding *type I* and *type II* neighbouring points to obtain a grid on which we can perform the forward and inverse wavelet transforms. We define the computational domain Γ and identify any dangling points. We can then compress the (original, non-transformed) solution vector by keeping values only on the optimised computational domain. We then compute a stiffness matrix for the regular points in Ξ. The application of the complete discretised operator is simply the application of the stiffness matrix on the points in Ξ and the application of the inverse wavelet transform for the points in $\Gamma \setminus \Xi$. We therefore have a stiffness matrix A, even though we do not access it directly. Once we have the stiffness matrix, time evolution can be applied with the method of lines using different numerical schemes. It is important to note that the stiffness matrix A can change with time, it is only its computational domain which is fixed between updates.

The computational domain is updated every few steps, this variable is controlled by "RE-

NEW_GRID". To update the domain, we decompress the solution vector u, interpolate the solution to the finest grid and apply the same steps as before to design the operator.

There is a trade off between the value of "RENEW_GRID" and the number of Type II neighbouring points we add. The more Type II points we add at the edge of the region of computational interest, the bigger our computational domain is, but the less often we need to renew the grid. Optimal values for such parameters must be found by trial and error at this point.

4 Numerical Results

We now give empirical results for the interpolating wavelet optimised finite differences method. Since IWOFD is, at its core, a finite difference method with an optimiser, it is natural to use a basic finite difference algorithm as a benchmark. It is hoped that IWOFD will provide the same accuracy with reduced computation time.

The IWOFD method was always implemented with three resolution levels, *i.e.* going from the highest resolution J_2 to the lowest resolution J_1 with $J_2 - J_1 = 2$. Donoho's biorthogonal interpolating wavelets with a cubic ($N = 4$) interpolation scheme were used in all experiments. Run times on a Pentium Xeon 3.4 GHz machine are given in hundredths of seconds. When a running time of 0 is given, it means the code ran in less than a hundredth of a second, when "XX" is given, it means that the program failed to find a solution, usually because the experiment took too long or sometimes because the time evolution algorithm did not converge.

Equity Derivatives

The Black-Scholes model [2] gives the price of European option[6] on a stock x with volatility σ as the solution to the following PDE

$$\begin{cases} \frac{\partial V}{\partial t}(x,t) + \frac{1}{2}\sigma^2 x^2 \frac{\partial^2 V}{\partial x^2}(x,t) + rx\frac{\partial V}{\partial x}(x,t) - rV(x,t) = 0 & \forall (t,x) \in [0,T] \times \mathbb{R}_+ \\ u(x,T) = \psi(x) & \forall x \in \mathbb{R}_+ . \end{cases} \tag{59}$$

We have implemented a direct tridiagonal solver which applies Gaussian elimination directly to a tridiagonal matrix. We have also implemented two iterative solvers, successive over-relaxation (SOR) with over-relaxation parameter ω fixed at 1.2 and a conjugate gradient squared (CGSq) method for non symmetrical matrices. The last solver is a backward differentiation solver for stiff problems taken from the *Lawrence Livermore National Laboratory* package *Slatec* [18].

The option valued is a plain vanilla at-the-money European call option with the following parameters:

[6] A *European call option*, with maturity T and exercise price K, is a security which pays $\psi(\mathbf{S}(T)) = (\mathbf{S}(T) - K)^+ := \max(\mathbf{S}(T) - K, 0)$ at the maturity date T

Stock price (S): 10 *Exercise price (K)*: 10 *Interest rate (r)*: 5%
Volatility (σ): 20% *Time to maturity (T)*: 1 Year.

The exact solution, found by direct integration is 1.04505. Results using IWOFD and finite differences are given in Table 2. The number of space discretisation points is 2^N and TT is the number of time discretisation steps. The time steps given do not apply to the backward differentiation solver as the method is adaptive and chooses its own time step. The value below the solution is the relative error at the exercise price. This can be seen as a kind of $||.||_\infty$ norm since it is where the error can be expected to be the greatest. For the IWOFD solver, we have used the thresholding value of $1e-06 \times \Delta S$, where ΔS is the space step. The number of type I points kept is proportional to the logarithm of the number of discretisation points, fixed to $2N$. We have recomputed the grid every 100 time steps.

We see that the fastest and most reliable solver is the tridiagonal solver. In this case, the improvements due to the use of the wavelet optimiser is negligible. For the iterative solver, the speed up factor is between 2 and 3. This is better than for the direct solver since the discretised operator has to be applied several times in order to invert the stiffness matrix. As a result, the thresholding is more effective. The speed up factor for the backward differentiation solver is around 4.

	CN Tridiag		CN SOR, $\omega = 1.2$		CN CGSq		Backward Diff	
	FDIFF	IWOFD	FDIFF	IWOFD	FDIFF	IWOFD	FDIFF	IWOFD
N= 5	1.03531	1.02121	1.03528	1.02119	1.03531	1.02121	1.03521	1.03523
TT =200	(9e-03)	(2e-02)	(9e-03)	(2e-02)	(9e-03)	(2e-02)	(9e-03)	(9e-03)
	T=0	T=0	T=1	T= 0	T=1	T=1	T=1	T=1
N= 6	1.04272	1.04271	1.04271	1.04271	1.04272	1.04271	1.04261	1.04264
TT =200	(2e-03)	(2e-03)	(2e-03)	(2e-03)	(2e-03)	(2e-03)	(2e-03)	(2e-03)
	T=0	T=0	T=1	T=1	T=1	T=1	T=5	T=3
N= 7	1.04449	1.04449	1.04449	1.04449	1.04449	1.04449	1.04442	1.04445
TT =400	(5e-04)	(5e-03)	(5e-04)	(5e-04)	(5e-04)	(5e-04)	(6e-04)	(6e-04)
	T= 4	T= 1	T=2	T=3	T=3	T=3	T=13	T=6
N= 8	1.04493	1.04493	1.04493	1.04493	1.04493	1.04493	1.04487	1.0449
TT =800	(1e-04)	(1e-04)	(1e-04)	(1e-04)	(1e-04)	(1e-04)	(2e-04)	(1e-04)
	T= 4	T= 5	T=9	T=9	T=13	T=9	T=35	T=13
N= 9	1.04504	1.04504	1.04503	1.04503	1.04504	1.04504	1.04501	1.04502
TT =800	(9e-06)	(9e-06)	(2e-05)	(2e-05)	(9e-06)	(9e-06)	(4e-05)	(3e-05)
	T=9	T= 7	T=40	T=23	T=47	T=16	T=269	T=67

Table 2: Black-Scholes vanilla call $V_{exact} = 1.04505$

Fixed Income Derivatives

We now value a Bermudan swaption under a three factor Gaussian short rate model. A *swap* is an agreement between two parties to exchange payments at fixed dates for a specified

amount of time, typically, one party pays a fixed amount while the other pays a *floating* rate function of the market at payment time. A *Bermudan swaption* is an option on a swap that can be exercised at a number of specified dates, entitling the owner to enter into a swap at a pre-specified rate. The three factor Gaussian short rate model assumes that the short-rate is the sum of a deterministic function $s(t)$ and a mean-zero stochastic process:

$$\boldsymbol{\tau}(t) = \mathbf{X}_1(t) + \mathbf{X}_2(t) + \mathbf{X}_3(t), \tag{60}$$

where the $\mathbf{X}_i$ are defined by the following stochastic differential equations (SDE).

$$d\mathbf{X}_i(t) = -\lambda_i \mathbf{X}_i(t)\ dt + \sigma_i d\mathbf{W}_i(t), \tag{61}$$

The $\mathbb{Q}$-Brownian motion $\mathbf{W}_i(t)$ are correlated such that $E[d\mathbf{W}_i(t)d\mathbf{W}_j(t)] = \rho_{ij}\ dt$. The function $s(t)$ is chosen so that the model fits the time-t^0 term structure. The bond prices under this model have been derived by Thompson [27] by direct integration of the bond equation. The pricing equation is a multi-dimensional version of (59) and can be found in [7]. The deal is a 5 year semi-annual settlement Bermudan swaption with notional 100\$, fixed rate 5% and the market parameters for the deal the following:

- *Flat term structure of 5%*
- Parameter values: $\lambda_1 = 0.01,\ \lambda_2 = 0.7,\ \lambda_3 = 3.0,$
  $\sigma_1 = 0.006,\ \sigma_2 = 0.012,\ \sigma_3 = 0.005,\ \rho_{12} = -0.8,\ \rho_{22} = 0.0.$

Table 3 gives the numerical results. The exact solution used is a Richardson extrapolation of all the results and was found to be within the interval of confidence of a simulation method developed by Thompson [28] similar to the techniques presented in [12] and [26]. The grid was revised at every settlement date during the time evolution. We see that the speed up factor is above 2 for the Dufort-Frankel solver. Results for the Crank-Nicolson method with SOR and biconjugate gradient stabilised have not been given as both methods took too long to converge at high resolution.

BGM Model

The next product is valued using the Brace-Gatarek-Musiela model [4]. It is a three year annual fixed-for-floating Bermudan swaption with notional 100\$ and strike price 5%. The holder can exercise the swaption at times $T_0 = 0$, T_1 or T_2 and enter a swap with maturity T_3. This is a two dimensional problem, with underlying variables L_1 and L_2, the LIBOR rates expiring at time T_2 and T_3 respectively. Data was taken from Blackham [3], provided by Dresdner Kleinwort Wasserstein.

- Parameter values: $L_0 = 0.02433306,\ L_1 = 0.03281384,\ L_2 = 0.03931690,$
  $\sigma_{L_1} = 24.73\%,\ \sigma_{L_2} = 22.45\%,\ \rho_{L_1,L_2} = e^{-0.1}$

For the IWOFD solver, we have used a thresholding value of $1e-06 \times \min(\Delta L_1, \Delta L_2)$, where ΔL_i is the space step for rate i. The number of type I points kept is proportional to the logarithm of the number of discretisation points, fixed to N. We have revised the grid every 200 time steps.

Table 4 gives the numerical results. Successive over relaxation and biconjugate gradient

	Dufort-Frankel	
	FDIFF	IWOFD
N=M=5,K=4	0.935071	0.935071
TT=100	(2.8e-02)	(2.8e-03)
	RT= 281	RT= 285
N=M=6,K=5	0.910827	0.910827
TT=100	(1.8e-03)	(1.8e-03)
	RT=2545	RT= 2392
N=M=7,K=5	0.907663	0.907663
TT=200	(1.7e-03)	(1.7e-03)
	RT= 17451	RT= 11358
N=M=8,K=5	0.907908	0.907906
TT=400	(1.4e-03)	(1.4e-03)
	RT= 119543	RT= 54245

Table 3: Gaussian model Bermudan fixed-for-floating swaption $V_{ext} = 0.911$

stabilised were used as time solvers. The reference solution was found using Richardson extrapolation. The speed up factor for SOR was 2 and BCGStab did not converge for the finest resolution with finite differences.

Figure 4 shows the payoff of the swaption at time $t = 0$ and an example of the grid used by the IWOFD solver.

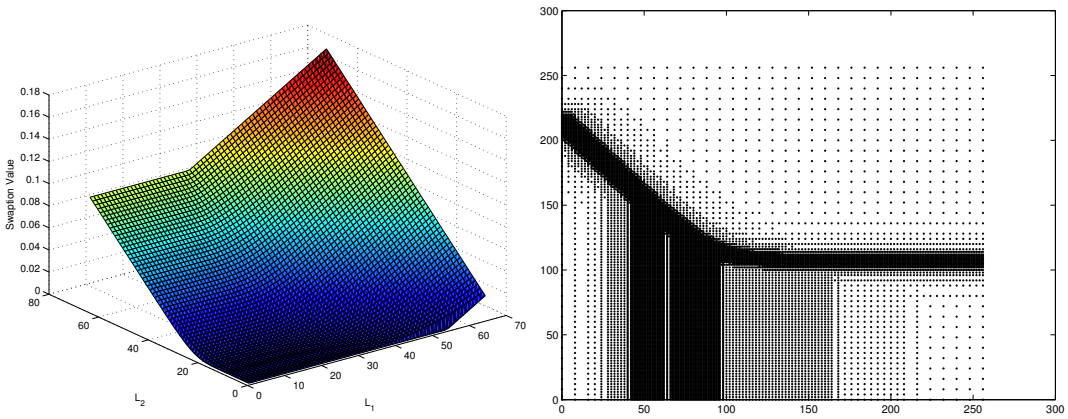

Figure 4: BGM model Bermudan swaption: payoff at time $t = 0$ (left), localisation of grid points at time $t = T_2$ (right)

5 Conclusions

In this paper, we have shown that an accurate, efficient and flexible adaptive method for contingent claim valuation could be designed using wavelets. The inherent problem with nu-

	CN SOR, $\omega = 1.2$		CN BCGStab	
	FDIFF	IWOFD	FDIFF	IWOFD
N=5 TT=200	0.112164 (3.0e-02) RT=6	0.112164 (3.0e-02) RT=8	RT=18	RT=18
N=6 TT=200	0.113605 (1.7e-02) RT=40	0.113605 (1.7e-02) RT=35	RT=91	RT=94
N=7 TT=400	0.114402 (1.0e-02) RT=272	0.114402 (1.0e-02) RT=269	RT=1094	RT=897
N=8 TT=800	0.115468 (1.1e-03) RT=2219	0.115468 (1.1e-03) RT=1847	RT=11803	RT=8027
N=9 TT=800	0.115653 (4.2e-04) RT=16235	0.115651 (4.0e-04) RT=8057	RT=XX	RT=38762

Table 4: BGM model fixed for floating Bermudan swaption $V_{ext} = 0.115604$

merical methods in mathematical finance is that the variety of problems is extremely vast. Firstly, different models, according to different markets or underlying variables, imply different kinds of partial differential equations which cannot all be transformed to the simplistic heat equation. Secondly, different derivative products imply different boundary conditions. The boundary conditions resulting from some problems, such as American or Bermudan options can be extremely hard to fit into a numerical scheme.

This wide variety of problems is the main reason why methods that can handle all kinds of PDEs and boundary conditions easily, such as finite differences, are so popular. The method described in this paper is based on finite differences and benefits from its extreme flexibility. It also makes extensive use of signal compression theory, more precisely wavelets, to design an optimal computational domain on which the PDE can be solved. The method has been applied to different practical problems, showing a speed-up factor between 2 and 4.

References

[1] G. Beylkin, R. Coifman, and V. Rokhlin, *Fast wavelet transforms and numerical algorithms I*, Communications in Pure and Applied Mathematics, 44 (1991), pp. 141–183.

[2] F. Black and M. Scholes, *The pricing of options and corporate liabilities*, The Journal of Political Economy, 81 (1973), pp. 635–654.

[3] J. Blackham, *Sparse grid solutions to the libor market model*, master's thesis, Magdalen College, University of Oxford, 2004.

[4] A. Brace, D. Gatarek, and M. Musiela, *The market model of interest rate dynamics*, Mathematical

Finance, 7 (1997), pp. 127–155.

[5] A. Bultheel, *Wavelets with applications in signal and image processing.* Lecture Notes, University of Louvain, `http://www.cs.kuleuven.be/~ade/WWW/WAVE/contents.html`, 2002.

[6] W. Cai and W. Zhang, *An adaptive spline wavelet adi (sw-adi) method for two-dimensional reaction-diffusion equations*, Journal of Computational Physics, 139 (1998), pp. 92–126.

[7] B. Carton de Wiart, *Wavelet Optimised PDE Methods for Financial Derivatives*, PhD thesis, Centre for Financial Research, Judge Institute of Management and St John's College, University of Cambridge, 2005.

[8] M. Charrier, D. Cruz, and M. Larsson, *JPEG2000, the next millennium compression standard for still images*, in Proceedings of the IEEE ICMCS'99, vol. 1, 1999, pp. 131–132.

[9] A. Cohen, *Multiscale adaptive processing for evolution equations*, in proceedings of ENUMATH conference (Ischia 2001), W. Hackbusch and G. Wittum, eds., 2002, pp. 99–119.

[10] A. Cohen, *Numerical Analysis of Wavelet Methods*, vol. 32 of Studies in Mathematics and Its Applications, North-Holland, Amsterdam, 2003.

[11] A. Cohen, I. Daubechies, and J. Feauveau, *Biorthogonal bases of compactly supported wavelets*, Communications on Pure and Applied Mathematics, 45 (1992), pp. 485–560.

[12] M. Curran, *Beyond average intelligence*, Risk, 5 (1992), pp. 60–67.

[13] S. Dahlke and A. Kunoth, *Biorthogonal wavelets and multigrid*, in Proceedings of the 9th GAMM-Seminar "Adaptive Methods: Algorithms, Theory and Applications", W. Hackbusch and G. Wittum, eds., Wiesbaden, 1994, Vieweg, pp. 99–119.

[14] M. Dempster and A. Eswaran, *Wavelet based PDE valuation of derivatives*, in Proceedings of the Third European Congress of Mathematics, C. Casacuberta *et al.*, ed., Progress in Mathematics, Basel, 2001, Birkhauser, pp. 347–365.

[15] M. Dempster, A. Eswaran, and D. Richards, *Wavelet methods in PDE valuation of financial derivatives*, in Proceedings of the Second International Conference of Intelligent Data Engineering and Automated Learning (IDEAL 2000), K. Leung, L. W. Chang, and H. Meng, eds., Berlin, 2000, Springer, pp. 215–238.

[16] G. Deslauriers and S. Dubuc, *Symmetric iterative interpolation process*, Constrained Approximation, 5 (1987), pp. 49–68.

[17] D. Donoho, *Interpolating wavelet transforms.* Presented at the NATO Advanced Study Institute conference, Ciocco, Italy., 1992.

[18] A. Hindmarsh, *Slatec common mathematical library.* `http://www.netlib.org/slatec`.

[19] L. Jameson, *On the Daubechies-based wavelet differentiation matrix.* Technical report, Institute for Computer Applications in Science and Engineering, NASA Langley, VA., 1996.

[20] ———, *On the wavelet optimized finite difference method.* Technical report, Institute for Computer Applications in Science and Engineering, NASA Langley, VA, 1996.

[21] L. Jameson, *A wavelet-optimized, very high order adaptive grid and order numerical method*, SIAM Journal on Scientific Computing, 19 (1998), pp. 1980–2013.

[22] L. Jameson and T. Miyama, *Wavelet analysis and ocean modeling: A dynamically adaptive numerical method "wofd-aho"*, Monthly Weather Review, 128 (2000), pp. 1536–1548.

[23] A. Louis, P. Maass, and A. Rieder, *Wavelets: Theory and applications*, Wiley, New-York, 1997.

[24] A.-M. Matache, C. Schwab, and T. Wihler, *Fast numerical solution of parabolic integro-differential equations with applications in finance*, SIAM Journal on Scientific Computing, 27 (2005), pp. 369–393.

[25] R. Prosser and R. Cant, *A wavelet-based method for the efficient simulation of combustion*, Journal of Computational Physics, 147 (1998), pp. 337–361.

[26] L. Rogers and Z. Shi, *The value of an Asian option*, Journal of Applied Probability, 32 (1995), pp. 1077–1088.

[27] G. Thompson, *Citibank project: The gaussian three-factor short-rate model.* Technical report, Centre for Financial Research, University of Cambridge, 2002.

[28] ———, *Pricing bermudan swaptions in three-factor gaussian interest-rate models using simulation.* Technical report, Centre for Financial Research, University of Cambridge, 2002.

[29] O. Vasilyev and S. Paolucci, *A fast adaptive wavelet collocation algorithm for multi-dimensional PDEs*, Journal of Computational Physics, 138 (1997), pp. 16–56.

[30] O. V. Vasilyev and C. Bowman, *Second generation wavelet collocation method for the solution of partial differential equations*, Journal of Computational Physics, 165 (2000), pp. 660–693.

[31] J. Walden, *Filter bank methods for hyperbolic pdes*, SIAM Journal on Numerical Analysis, 36 (1999), pp. 1183–1233.

[32] ——, *A general adaptive solver for hyperbolic pdes based on filter bank subdivisions*, Applied Numerical Mathematics, 33 (2000), pp. 317–325.