UCLA
COMPUTATIONAL AND APPLIED MATHEMATICS

Computation of Shift Operators in Orthonormal Compactly Supported Wavelet Bases

Vincente F. Candela

September 1991

CAM Report 91-19

Department of Mathematics
University of California, Los Angeles
Los Angeles, CA. 90024-1555

COMPUTATION OF SHIFT OPERATORS IN ORTHONORMAL COMPACTLY SUPPORTED WAVELET BASES

VICENTE F. CANDELA

September 12, 1991

Author's Current Address:
Vicente F. Candela
Departamento de Matemática Aplicada
Universitat de València
C/ Dr. Moliner, 50
46100-Burjassot (Valencia) SPAIN
e-mail: candela@evalun11.bitnet
fax: (346)3864356

Computation of shift operators in orthonormal compactly supported wavelet bases

Vicente F. Candela *

Departamento de Matemática Aplicada
Universitat de València

September 12, 1991

Abstract

In this report some algorithms to translate signals using orthonormal compactly supported wavelet bases are introduced. We obtain the approximation to the shift operator in these bases, and we study the performance of the different wavelets through examples.

1 Introduction

In recent years, *wavelet bases* have been used in a wide variety of topics in both pure and applied mathematics. The concept of wavelet itself is vague enough to be satisfied by a great number of classes of functions. Otherwise, it provides a multiscale structure which allows fast algorithms similar to those used in Fourier analysis.

In [4], Daubechies introduced a class of *compactly supported orthogonal wavelet bases* in $\mathcal{L}^2(R)$, the space of squared integrable functions. While the orthogonality of these bases produces some nice properties similar to those given by trigonometric bases, the compactness of the wavelet support avoids some of the problems inherent to Fourier series. One of the main problems of the class of wavelets in [4] is its behaviour with translations. Unlike the Fourier case, in which the shift operator is represented by a diagonal matrix, orthogonal wavelet bases do not have a simple way to represent translations.

*Research supported by a Grant from Conselleria de Cultura, Educaciò i Ciència de la Generalitat Valenciana, and partly supported by a DARPA Grant in the ACM Program.

This problem was partly solved by Beylkin in [1]. He found exact and explicit representations of the differentiation and the shift operators for integer shiftings in these wavelets bases. If we focus only on the translation of functions, these results are especially useful when all the wavelet values that we need are known. Sometimes, however, we can need a large number of wavelet values in order to achieve a high accuracy, or we do not have all the values we need. Usually, in order to get values of the wavelets, we have a great increase of the computational cost (these values are obtained either by a recursive algorithm or by evaluation of Fourier transforms). Furthermore, the storage of these values can be impractical if we require high accuracy. So, the representation of the shift operator for noninteger translations is not a purely academic problem.

The purpose of this report is to obtain these representations and to provide some simple algorithms in order to evaluate them, using Beylkin's mentioned results. From these representations it is a direct and easy application to get the nonstandard form of these operators (such as it is defined in [2]).

This report is organized as follows. In section §2, the class of wavelets in [4] is introduced; in that section, we also assert the notation and assumptions used along the report. We refer the reader to [4] or [8] in order to get more information about the properties, applications and algorithms related to wavelets. In §3 we get the representation of the shift operator, in two different ways, one exact up to the resolution we are dealing with for dyadic shifts, and the other one approximated for any kind of shifts. These approximations can be accurate either up to the roundoff error, or, in a faster way, up to the truncation error, with a very reasonable increase of computational cost. Finally, in §4, we implement this algorithm for some particular examples and we show some numerical results.

2 Representation of functions in a compactly supported wavelet basis

In [5], wavelets were introduced related to the solution of a two-scale difference equation, called *scaling function*. We denote by $\varphi(x)$ such scaling function, and we assume it satisfies

$$\varphi(x) = \sqrt{2} \sum_{k=0}^{n} h_k \varphi(2x - k) \tag{1}$$

where $\{h_k\}_{k=0}^n$ are real numbers. The scaling function and its properties are completely characterized by these numbers.

In [4], necessary and sufficient conditions on the constants $\{h_k\}_{k=0}^n$ in order to get a scaling function *orthogonal* to its integer translates were found. Orthogonality and a finite number of nonzero coefficients h_k are sufficient, though not necessary, conditions in order to guarantee that the support of the scaling function is compact, and $\varphi(x)$ is zero outside $[0, n]$.

The wavelet basis is obtained dilating and translating a *wavelet mother* $\psi(x)$, which is related to the scaling function according to the following equations:

$$\psi(x) = \sqrt{2} \sum_{k=0}^{n} g_k \varphi(2x - k) \tag{2}$$

$$g_k = (-1)^k h_{n-k} \tag{3}$$

Let us denote

$$\varphi_k^j(x) = 2^{-j/2} \varphi(2^{-j}x - k) \tag{4}$$

$$\psi_k^j(x) = 2^{-j/2} \psi(2^{-j}x - k) \tag{5}$$

We shall assume that the wavelet mother generates an orthonormal wavelet basis, that is, $\{\psi_j^k(x)\}_{j,k}$ is an orthonormal basis in $\mathcal{L}^2(R)$ with the usual norm and inner product. Furthermore, we shall assume that the first p moments of the *wavelet mother* are zero,

$$\int_{-\infty}^{+\infty} x^m \psi(x) dx = 0, for \quad m = 0, 1, \cdots, p-1 \tag{6}$$

Necessary and sufficient conditions in order to verify those assumptions can be found in [4].

A wavelet basis induces a multiresolution analysis (MRA) ([7]) in $\mathcal{L}^2(R)$. Thus, if we call W_j and V_j the subspaces generated by $\{\psi_k^j(x)\}_k$ and $\{\varphi_k^j(x)\}_k$ respectively, then

$$\bigoplus_{j=-\infty}^{+\infty} W_j = \bigoplus_{-\infty}^{+\infty} V_j = \mathcal{L}^2(R) \tag{7}$$

$$V_j \bigoplus W_j = V_{j-1} \tag{8}$$

and we can represent any function $f(x)$ in $\mathcal{L}^2(R)$ in each of both bases:

$$f(x) = \sum_{j,k} d_k^j \psi_k^j(x) \tag{9}$$

$$= \sum_{j,k} s_k^j \varphi_k^j(x) \tag{10}$$

where, due to the orthonormality,

$$s_k^j = \int_{-\infty}^{+\infty} f(x)\varphi_k^j(x)dx \tag{11}$$

$$d_k^j = \int_{-\infty}^{+\infty} f(x)\psi_k^j(x)dx \tag{12}$$

Usually the representation of the function $f(x)$ is given in a finite number of scales, for example, between the finest scale $j = n_1$ and the coarsest one $j = n_2$. In such a case, we have an approximation of the original function ([8]):

$$f(x) = \sum_k s_k^{n_2}\varphi_k^{n_2}(x) + \sum_{j=n_1}^{n_2}\sum_k d_k^j \psi_k^j(x) + O(h^p) \tag{13}$$

where $h = 2^{n_1-1}$ and p is the number of zero moments of the wavelet (6).

This last expression is equivalent to

$$f(x) = \sum_k s_k^{n_1-1}\varphi_k^{n_1-1}(x) + O(h^p) \tag{14}$$

We can assume the finest scale $n_1 = 1$, and the function $f(x) \in V_0$, that is

$$f(x) = \sum_k s_k^0 \varphi_k^0(x) \tag{15}$$

We shall justify in §3 that this is not a tight restriction. For simplicity, we will suppress the scale index, 0, in both the scaling function and the coefficients.

3 Shift operators in a wavelet basis

3.1 Standard and non-standard form of a matrix

In [2], Beylkin, Coifman and Rokhlin introduce the concept of nonstandard form of a matrix in a multiresolution analysis. Roughly speaking, the nonstandard form of a matrix is the representation of such a matrix in all

different scales. It has the advantage that most dense $(N \times N)$ matrices can be compressed, and the product of that matrix times a vector can be done in a fast way, just in $O(N \log_2 N)$ scalar multiplications. It is immediate to get the non-standard form of a matrix once we have it in the finest scale with respect to the scaling function (in our case, in V_0). In fact, both representations are equivalent. We refer the reader to [2]. In [1] some of the algorithms to get the nonstandard form of a matrix from the finest representation can also be found.

So, our aim is to obtain the finest scale representation of the shift operator. We consider different kind of shifts. In the next subsection, we get the exact matrix associated with dyadic shifts, and, next, we obtain approximations for any shift.

We shall denote by $V^j f$ the projection of the function $f(x)$ on the space V_j, and $f_\lambda(x) = f(x - \lambda)$, λ a real number.

As it was said in §2, we will assume that $f(x) \in V_0$. If the finest scale is different from zero, all our results are valid with some small normalizing changes. All the details concerning the *finer than zero* scales are obviously lost in our representation, because we limit our resolution to that scale. So, when we qualify our representation as *exact*, this exactness is referred to the resolution in the space V_0 and not to $\mathcal{L}^2(R)$.

3.2 The shift operator

The problem we attempt is the following:

We assume

$$f(x) = \sum_k s_k \varphi_k(x) \tag{16}$$

and we want to know the coefficients $\tilde{s}_k$ corresponding to the function $f_\lambda(x)$.

First, we observe that $f(x) \in V_0$ does not imply that $f_\lambda(x)$ is in V_0. In general, this is only true if λ is an integer. We assume that we are limited to the *zero-scale*, and, thus, we accept the approximation in that scale. Our problem, formally, can be expressed in this terms:

Problem

Given $\{s_k\}_k$, *the scaling coefficients of* $f(x)$,
find $\{\tilde{s}_k\}_k$, *the scaling coefficients of* $V^0 f_\lambda(x)$.

As the scaling basis is orthonormal, the coefficients $\tilde{s}_k$ must satisfy:

$$\tilde{s}_k = \int_{-\infty}^{+\infty} f_\lambda(x)\varphi_k(x)dx \tag{17}$$

On the other hand,

$$f_\lambda(x) = f(x-\lambda) = \sum_k s_k \varphi_k(x-\lambda) \tag{18}$$

From (17) and (18),

$$\tilde{s}_k = \sum_l s_l \int_{-\infty}^{+\infty} \varphi_l(x-\lambda)\varphi_k(x)dx = \sum_l s_l \int_{-\infty}^{+\infty} \varphi(x-\lambda)\varphi_{k-l}(x)dx$$

If we call,

$$r_j = \int_{-\infty}^{+\infty} \varphi(x-\lambda)\varphi_j(x)dx \tag{19}$$

Then

$$\tilde{s}_k = \sum_l s_l r_{k-l} \tag{20}$$

All we have to do is find $\{r_j\}_j$. The rest of this section is aimed to get those coefficients. In general, they can be obtained using any numerical quadrature, but these methods usually need a large amount of storage of the values of $\varphi(x)$ (as it was said in §1, these values cannot be evaluated directly without a great amount of computational work) and an increasing number of multiplications in order to evaluate the integral if we want to get high accuracy. The scheme we propose allows us to get $\{r_j\}_j$ according to an iterative algorithm that requires less than a fixed number of operations, depending only on the roundoff error, and not on the truncation one. Another advantage is that we obtain the coefficients directly from the dilation equation, and, so, we do not need to keep any value of $\varphi(x)$, saving most of the storage we should need otherwise.

First, we remark that, if the scaling function satisfies (1), and it generates an orthonormal basis, then its support is contained in $[0, n]$. Therefore, there are at most $2n-1$ nonzero coefficients r_k.

We distinguish two different cases:

3.2.1 λ is a dyadic number

Let us assume that there exist two integers, j and m, such that

$$\lambda = \frac{j}{2^m} \tag{21}$$

If we denote by

$$r_k^l = \int_{-\infty}^{+\infty} \varphi(x - \frac{j}{2^l})\varphi_k(x)dx \tag{22}$$

then, $r_k = r_k^m$, and

$$r_k^0 = \int_{-\infty}^{+\infty} \varphi(x-j)\varphi(x)dx = \begin{cases} 1, & \text{if } k = j \\ 0, & \text{otherwise} \end{cases} \tag{23}$$

Using (1),

$$\begin{aligned} r_k^{l+1} &= \int_{-\infty}^{+\infty} \varphi(x - \frac{j}{2^{l+1}})\varphi_k(x)dx \\ &= 2\sum_{p=0}^{n}\sum_{q=0}^{n} h_p h_q \int_{-\infty}^{+\infty} \varphi(2x - \frac{j}{2^l} - p)\varphi(2x - 2k - q)dx \\ &= \sum_{p=0}^{n}\sum_{q=0}^{n} h_p h_q \int_{-\infty}^{+\infty} \varphi(x - \frac{j}{2^l})\varphi_{2k+q-p}(x)dx \end{aligned} \tag{24}$$

Changing the order of the sums, and denoting $a_t = \sum_{q=0}^{n-t} h_q h_{q+t}$, if $t \geq 0$, and $a_t = \sum_{q=t}^{n} h_q h_{q+t}$ if $t \leq -1$, finally, we have

$$r_k^{l+1} = \sum_{t=-n}^{n} a_t r_{2k+t}^l \tag{25}$$

The parameters a_t depend only on the number of vanishing moments of the wavelets ([1]), and they are zero when t is even, except for $t = 0$, where $a_0 = 1$. Moreover, $a_t = a_{-t}$.

We get then the following recurrent relation:

Algorithm 1

For $l = 0$, and for all k

$$r_k^0 = \begin{cases} 1, & \text{if } k = j \\ 0, & \text{otherwise} \end{cases} \tag{26}$$

For $l = 0, \cdots, m-1$
  for all k

$$r_k^{l+1} = r_{2k}^l + \sum_{t=0}^{\frac{n-1}{2}} a_{2t+1}(r_{2(k+t)+1}^l + r_{2(k-t)-1}^l)$$

For all k

$$r_k = r_k^m$$

Remark A similar algorithm can be used to evaluate the translation of the derivatives of wavelets. In [1], it is proved that if

$$r_k^{(l)} = \int \varphi_k(x) \frac{d^l}{dx^l} \varphi(x) dx \tag{27}$$

exist for all integer k, then they are the solution of this algebraic linear system:

$$r_k^{(l)} = 2_l\{r_{2k}^{(l)} + \sum_{t=1}^{\frac{n-1}{2}} a_{2t+1}(r_{2k-2t-1}^{(l)} + r_{2k+2t+1}^{(l)})\}$$

$$\sum_k k^l r_k^{(l)} = (-1)^l l! \tag{28}$$

Furthermore, because of (1), the derivatives of the scaling functions, $\varphi^{(l)}(x) = \frac{d^l}{dx^l}\varphi(x)$, verify a two scale difference equation similar to (1):

$$\varphi^{(l)}(x) = \sqrt{2} \sum_{k=0}^{n} h_k^{(l)} \varphi^{(l)}(2x - k) \tag{29}$$

where, for all $k = 0, \cdots, n$, and for l such that $\varphi^{(l)}(x)$ exists,

$$h_k^{(l)} = 2^l h_k \tag{30}$$

Denoting, in a similar way as we did in *algorithm 1*, $a_t^{(l)} = \sum_{q=0}^{n-t} h_q^{(l)} h_{q+t}^{(l)}$, if $t \geq 0$, and $a_t^{(l)} = \sum_{q=t}^{n} h_q^{(l)} h_{q+t}^{(l)}$ if $t \leq -1$, then it is clear that

$$a_t^{(l)} = 2^l a_t \tag{31}$$

Finally, we can get the values of

$$\tilde{r}_k^{(l)} = \int \varphi_k(x) \frac{d^l}{dx^l} \varphi(x - \lambda) dx \tag{32}$$

substituting in *algorithm 1* the initial conditions (26) by the values $r_k^{(l)}$ (that is, $r_k^{(l)0} = r_k^{(l)}$), and using the corresponding values (those with the superscript (l)) throughout the rest of the algorithm.

3.2.2 λ any real number

Although any real number can be approximated by dyadic numbers, the *algorithm 1* can become practically complicated when we want an accuracy equivalent to the roundoff error. The problem is not the number of iterations (in most machines, double precision is reached with 2^{-50}, and that means that we do not need to iterate the process more than fifty times to have an approximation to the shift operator by any given real number in double precision), but the value of j when it is large. In that case, the auxiliary vector $\{r_k^l\}_k$ must have a large dimension even though only a small number of its components ($2n-1$ for each iteration l) are nonzero. We present now a modification of *algorithm 1*, in which that problem is avoided. This is more adequate in general for any λ, and it is easy to implement provided λ is given in the binary system.

First, we observe that we can restrict ourselves to the case when $|\lambda| < 1$, because a shift by $\lambda - j_0$ for j_0 an integer is equivalent to a shift in the coefficients. If

$$r_k = \int_{-\infty}^{+\infty} \varphi(x - \lambda) \varphi_k(x) dx \tag{33}$$

$$\tilde{r}_k = \int_{-\infty}^{+\infty} \varphi(x - (\lambda - j_0)) \varphi_k(x) dx \tag{34}$$

Then,

$$\tilde{r}_k = \int_{-\infty}^{+\infty} \varphi(x - \lambda) \varphi_{k-j_0}(x) dx = r_{k-j_0} \tag{35}$$

and all the results are valid, except for an eventual shifting in the indexes of the r_k.

So, we consider the binary expansion of λ,

$$\lambda = \sum_{l=1}^{m} \frac{p_l}{2^l} \tag{36}$$

where $p_l = 0$ or 1.

We define

$$\omega_k^l = \int \varphi(x - \sum_{s=l}^{m} \frac{p_s}{2^{s-l}})\varphi_k(x)dx \tag{37}$$

Thus,

$$\omega_k^m = \int \varphi(x - p_m)\varphi_k(x)dx = \begin{cases} 1, & \text{if } k = p_m \\ 0, & \text{otherwise} \end{cases} \tag{38}$$

And, similarly to *algorithm 1* we can get $r_k = \omega_k^0$ recursively, using (1)

$$\begin{aligned} \omega_k^{l-1} &= \int \varphi(x - \sum_{s=l-1}^{m} \frac{p_s}{2^{s-l+1}})\varphi_k(x)dx \\ &= \int \varphi(x - \sum_{s=l}^{m} \frac{p_s}{2^{s-l+1}})\varphi_{k-p_{l-1}}(x)dx \\ &= \sum_{t=-n}^{n} a_t \int \varphi(x - \sum_{s=l}^{m} \frac{p_s}{2^{s-l}})\varphi_{2k-2p_{l-1}+t}(x)dx \\ &= \sum_{t=-n}^{n} a_t \omega_{2k-2p_{l-1}+t}^l \end{aligned} \tag{39}$$

according to the same properties for a_t as in *algorithm 1*,

Algorithm 2

For all k,

$$\omega_k^m = \int \varphi(x - p_m)\varphi_k(x)dx = \begin{cases} 1, & \text{if } k = p_m \\ 0, & \text{otherwise} \end{cases}$$

For $l = m, \cdots, 1$
For all k,

$$\omega_k^{l-1} = \sum_{t=0}^{\frac{n-1}{2}} a_{2t+1}(\omega_{2(k-p_{l-1}+t)+1}^l + \omega_{2(k-p_{l-1}-t)-1}^l)$$

For all k,

$$r_k = \omega_k^0$$

3.2.3 Approximation of the shift operator

In the last paragraph we got an algorithm to evaluate any shift operator. We notice that the number of iterations we have to evaluate in that algorithm depends only on the roundoff error (for example, in most machines, *algorithm 2* is exact up to double precision in at most fifty iterations, as much as *binary decimals* has any real number in double precision). However, we do not usually require an accuracy up to the roundoff error, because the data are given with some truncation errors, larger than the roundoff ones. As a consequence, we shall limit the number of iterations according to the data. Anyway, if we have to evaluate a great number of shifts, *algorithm 2* does not provide a simple way to evaluate them (each shift needs the same number of ellementary operations, and all the process can be very slow. Let us remark, though, that in this algorithm the iterations consist of sums and multiplications, and the computational cost can be comparable, sometimes, to the evaluation of the shift in Fourier series, because in this one exponential evaluations are needed).

In this paragraph, we will assume that the wavelet is n_0 times differentiable, and its n_0 derivative is bounded. For example, the wavelets listed in [1] verify those assumptions.

We present now an approximation to the shift operator, which is obtained with less multiplications than in *algorithm 2* (the number of multiplications in the following algorithm is equal to n_0). This approximation allows to get immediate representations when we need to change the shifting often in a *small range* of real λ, or for *any* lambda, if we can store some amount of data and we are dealing with a certain truncation error. The price to pay is less accuracy in the algorithm.

Let us assume our truncation error is $O((\frac{1}{2^m})^{n_0})$.

Using the remark in §3.2.1, if $|\lambda| \leq \frac{1}{2^m}$, we can approximate r_k, using Taylor's theorem,

$$r_k = \int \varphi(x-\lambda)\varphi_k(x)dx = \sum_{i=0}^{n_0} \frac{\lambda^i}{i!} \int \frac{d^i}{dx^i}\varphi(x)\varphi_k(x)dx + O((\frac{1}{2^m})^{n_0})$$

$$= \sum_{i=0}^{n_0} \frac{\lambda^i}{i!} r_k^{(i)} + O((\frac{1}{2^m})^{n_0}) \quad (40)$$

If $|\lambda|$ is greater than $\frac{1}{2^m}$, then we can use a mixed algorithm. And we have:

Algorithm 3

$$\lambda = \frac{j}{2^m} + \tilde{\lambda}, where \quad |\tilde{\lambda}| < \frac{1}{2^m}$$

For $l = 0, \cdots, n_0$, *and for all* k,
Evaluate, using algorithm 1 or algorithm 2,

$$\tilde{r}_k^{(l)} = \int \frac{d^l}{dx^l}\varphi(x - \frac{j}{2^m})\varphi_k(x)dx$$

$$r_k = \sum_{l=0}^{n_0} \frac{\tilde{\lambda}^l}{l!}\tilde{r}_k^l$$

Remark 1 When m is small, the values $\tilde{r}_k^{(l)}$ may be stored in order to compute r_k immediately for all λ. This storage is less than the one needed to save the values of r_k for many different λ. In practice, *algorithm 3* provides high accuracy for m less than five or six, and the storage in this case is reduced to thirty-two or sixty-four different values for each $\tilde{r}_k^{(l)}$ in order to get immediate values r_k for *every* λ. This is not possible if the r_k are evaluated directly using (19).

Remark 2 Notice the theoretical similarity between this last representation and the shift representation in the Fourier transform. In fact, one approach to the shifting operator in Fourier transform could be done using *algorithm 3*, and the differentiability of the exponential (and, thus, with an infinite series instead of a finite sum in (41)).

Remark 3 *Algorithm 3* is obtained from Beylkin's results in [1], for $m = 0$. In such a case, the accuracy of the algorithm depends on the step size in the finest grid. The greater m is, the greater accuracy we get in *algorithm 3*. In fact, the choice of m makes it possible to get any required approximations even if the step size in the finest grid is large.

4 Numerical results

In this section, we shall present some of the results obtained after implementing the above algorithms.

In [2], there are some simple quadratures to get the scaling coefficients s_k^j of a function $f(x)$ as samples of another function $F(x) = \sum_{i=0}^{N} c_i f(x - t_i)$, where c_i and t_i are fixed real constants depending only on the wavelet.

Due to the linearity of the scaling coefficients, (11), a translation of $f(x)$ produces the same translation of $F(x)$ and *viceversa*. From now on, we are going to consider the shift of $F(x)$ instead of the original function, $f(x)$, in order to study the performance of the methods presented here. Indeed, all the properties inherent to the wavelet basis (such as the decay of the coefficients in the coarsest scales when there is regularity), and the order of accuracy of the algorithms are kept if instead of getting the scaling coefficients of the function $f(x)$, we consider samples of $F(x)$. In these examples, the functions we mention are referred to $F(x)$. The corresponding $f(x)$ exists in these examples, although in general it may not exist. (Nevertheless, the order of accuracy of the quadratures in [2] can decay wherever $f(x)$ is not smooth).

The wavelet bases we will use are Daubechies ones constructed in [4], for $M = 2, 3, 4, 5, 6$ vanishing moments and support in $[0, n]$, with $n = 3, 5, 7, 9, 11$ respectively.

The functions to be tested are periodic with period 2, and the evaluations are done in the interval $[-1, 1]$. Among the examples we have run for different wavelet bases and various shift sizes, we have chosen the following ones, that we consider as representatives.

4.1 Example 1

In order to compare the translations with the different wavelet bases, we start testing the smooth function $\sin \pi x$.

We try the above mentioned wavelets, applying *algorithm 1* and we evaluate the numerical order of the error. Although the order of the quadrature in [2] is the number of vanishing moments, the order of the translation, actually, is the length of the support. So, in order to draw adequate conclusions, we assume the coefficients s_k^0 are exact and not just approximations.

In *tables 1-3*, we list the absolute errors and the numerical orders of the translations for different wavelets, λ, and number of points, applying *algorithm 1*; h is the length of the intervals in the finest scale. The numerical order is evaluated according to h. We denote by an $*$ those errors close to the double precision. That is the reason why we do not consider the numerical orders of the wavelet with support $n = 11$.

In *tables 4, 5* we show the results obtained for the same function using

algorithm 3 for the wavelet with support $n = 5$, up to the second derivative. In this case, the error depends on the shifting λ, as we pointed out before (the numerical order with respect to λ is 2., as it can be expected). Some accuracy is lost in this algorithm as a consequence of using an approximated representation and not an exact one. The low computational cost of this method, and its adaptability to any λ, are remarkable. Furthermore, the order of accuracy of this method is approximately the number of zero moments of the wavelet, and this is the accuracy in which the scaling coefficients are often given. So, the order of *algorithm 3* is usually the truncation order.

4.2 Example 2

We study the resolution of our methods after testing the shift on the oscillatory function $\sin(16\pi x)$. In *figure 1* we can see the results obtained for the wavelets with support in $[0, 11]$, applying *algorithm 1*, after 100 periods (25600 iterations). In all the figures we show, the exact translation is displayed through a solid line and the obtained approximation is marked with small circles.

Tables 6 and *7* list the absolute errors and numerical orders applying *algorithm 1*. The wavelet $n = 3$ does not approximate the function with the number of points we consider here. Unlike *example 1*, the order depends on λ, but this dependence is not monotonic, in spite of the appearance of these tables.

4.3 Example 3

This example appears in [9]. We test the function

$$f(x) = \exp(-300x^2)$$

in order to consider the behaviour of the wavelets in the maximum, $x = 0$. From our numerical experiments, we see that this extremum is well preserved by the higher order wavelets. The difference between the L_∞ and the L_1 errors is due to the extremum in this function. *Figures 2* and *3*, show the results with the low order wavelet, $N = 3$ (one period) and the higher order $n = 11$ (one hundred periods).

4.4 Concluding remarks

In this report, we have obtained a representation of the shift operator, for noninteger shifts. Our representation is exact up to the finest scale when

the shift is a dyadic number, and we can approximate it up to the roundoff error (or the truncation one) in any other case.

The order of the translation is at least the same as the number of zero moments of the wavelet, and, in some cases, it reaches the length of the support. Therefore, the methods described are at least as accurate as most of the quadratures used in order to evaluate the scaling or the wavelet expansions of the function.

The greater the support of the wavelet mother (or the scaling function), the greater accuracy we get, and the better these methods behave, as it was expected. In particular, the wavelet supported in $[0, 11]$, provides high resolution at local extrema. These properties can justify the use of wavelets and the methods above mentioned in order to translate functions. Once we get the nonstandard form of these representations ([2]), the wavelet coefficients in the coarsest scales decay faster if the support of the wavelet is greater. As a consequence, the algorithms to evaluate the operator in nonstandard form need less computational work than other similar methods (compared, for instance, to trigonometric basis). This allows the use of high order methods with a moderate increase of computational work.

Finally, we pay attention to the minimum storage required. All the necessary storage to apply these algorithms is provided by the coefficients a_k. The proposed methods do not require actual values of the wavelet, unlike most of the methods related to wavelet bases. So, the values of the wavelet are only necessary in order to get the coefficients s_k, if we do not use the quadratures in [2].

Acknowledgements

The author thanks S. Osher for suggesting the problem and a lot of valuables ideas about it; T. Mathew and S. Zhong for many fruitful discussions. Last, and maybe least, he would like to thank A. Marquina for providing some suggestions throughout the writing of this report, a couple of them really useful. Perhaps he will thank him in a future, if he uses them...

References

[1] G. Beylkin, *On Representation of Operators in Compactly Supported Wavelet Bases*, preprint

[2] G. Beylkin, R. Coifman and V. Rokhlin, *Fast Wavelet Transform and Numerical Algorithms I.*, Yale University Technical Report, (1989)

[3] J.M. Combes, A. Grossmann, Ph. Tchamitchian (Eds.) *Wavelets. Time-Frequency Methods and Phase Space (Proceedings of the International Conference, Marseille, France, December 14-18, 1987)*, 2nd. ed., Springer-Verlag, Berlin, (1990)

[4] I. Daubechies *Orthonormal Bases of Compactly Supported Wavelets*, Comm. Pure, Applied Math., **XLI**, (1988)

[5] I. Daubechies and J. Lagarias, *Two-scale Difference Equations, I. Existence and global regularity of solutions*, preprint

[6] I. Daubechies and J. Lagarias, *Two-scale Difference Equations, II. Local regularity, Infinite Products of Matrices and Fractals*, preprint

[7] S. Mallat, *Multiresolution approximations and wavelet orthonormal bases of* $\mathcal{L}^2(R)$, Trans. Amer. Math. Soc., **Vol. 315**, 69-87, (1989)

[8] G. Strang, *Wavelets and Dilation Equations: A Brief Introduction*, SIAM Review, Vol. **31**, 4, pp. 614-627, (1989)

[9] S. Zalesak, *A Preliminary Comparison of Modern Shock-Capturing Schemes: Linear Advection* in *Advances in Computer Methods for Partial Differential Equations, VI*, edited by R. Vichnevetsky and R. Stepleman (IMACS, New Brunswick, NJ, 1987)

h		1/16	1/32	1/64
$n=3$	L_1	$0.282 \cdot 10^{-2}$	$0.354 \cdot 10^{-3}$	$0.443 \cdot 10^{-4}$
	L_∞	$0.222 \cdot 10^{-2}$	$0.278 \cdot 10^{-3}$	$0.348 \cdot 10^{-4}$
$n=5$	L_1	$0.226 \cdot 10^{-4}$	$0.711 \cdot 10^{-6}$	$0.223 \cdot 10^{-7}$
	L_∞	$0.178 \cdot 10^{-4}$	$0.559 \cdot 10^{-6}$	$0.175 \cdot 10^{-7}$
$n=7$	L_1	$0.190 \cdot 10^{-6}$	$0.150 \cdot 10^{-8}$	$0.117 \cdot 10^{-10}$
	L_∞	$0.150 \cdot 10^{-6}$	$0.118 \cdot 10^{-8}$	$0.921 \cdot 10^{-11}$
$n=9$	L_1	$0.164 \cdot 10^{-8}$	$0.325 \cdot 10^{-11}$	*
	L_∞	$0.129 \cdot 10^{-8}$	$0.255 \cdot 10^{-11}$	*
$n=11$	L_1	$0.145 \cdot 10^{-10}$	*	*
	L_∞	$0.114 \cdot 10^{-10}$	*	*

Table 1: Absolute errors for *example 1*; n=length of the support of the wavelet; $\lambda = 1/2$

h		1/16	1/32	1/64
$n=3$	L_1	$0.118 \cdot 10^{-2}$	$0.141 \cdot 10^{-3}$	$0.173 \cdot 10^{-4}$
	L_∞	$0.913 \cdot 10^{-3}$	$0.110 \cdot 10^{-3}$	$0.136 \cdot 10^{-4}$
$n=5$	L_1	$0.470 \cdot 10^{-5}$	$0.163 \cdot 10^{-6}$	$0.485 \cdot 10^{-8}$
	L_∞	$0.616 \cdot 10^{-5}$	$0.127 \cdot 10^{-6}$	$0.380 \cdot 10^{-8}$
$n=7$	L_1	$0.484 \cdot 10^{-7}$	$0.295 \cdot 10^{-9}$	$0.214 \cdot 10^{-11}$
	L_∞	$0.369 \cdot 10^{-7}$	$0.229 \cdot 10^{-9}$	$0.168 \cdot 10^{-11}$
$n=9$	L_1	$0.430 \cdot 10^{-9}$	*	*
	L_∞	$0.324 \cdot 10^{-9}$	*	*
$n=11$	L_1	$0.368 \cdot 10^{-11}$	*	*
	L_∞	$0.276 \cdot 10^{-11}$	*	*

Table 2: Absolute errors for *example 1*; n=length of the support of the wavelet; $\lambda = 1/32$

	λ	L_1	L_∞	L_2
$n=3$	1/2	2.995	2.995	2.995
	1/32	3.079	3.064	3.069
$n=5$	1/2	4.993	4.993	4.993
	1/32	5.232	5.214	5.225
$n=7$	1/2	6.990	6.990	6.990
	1/32	7.341	7.333	7.343
$n=9$	1/2	8.987	8.987	8.987
	1/32	9.304	9.286	9.288

Table 3: Numerical orders for *example 1*; h=1/16

h		1/16	1/32	1/64
$\lambda = 1/2$	L_1	$0.135 \cdot 10^{-1}$	$0.329 \cdot 10^{-2}$	$0.813 \cdot 10^{-3}$
	L_∞	$0.101 \cdot 10^{-1}$	$0.252 \cdot 10^{-2}$	$0.631 \cdot 10^{-3}$
$\lambda = 1/32$	L_1	$0.493 \cdot 10^{-4}$	$0.128 \cdot 10^{-4}$	$0.317 \cdot 10^{-5}$
	L_∞	$0.367 \cdot 10^{-4}$	$0.981 \cdot 10^{-5}$	$0.246 \cdot 10^{-5}$

Table 4: Absolute errors for *example 1*, using *algorithm 3*; $n = 5$; $\tilde{\lambda} = \lambda$

	L_1	L_∞	L_2
$\lambda = 1/2$	2.002	1.998	2.000
$\lambda = 1/32$	1.886	1.882	1.882

Table 5: Numerical orders for *example 1*, using *algorithm 3*; $n = 5$; $\tilde{\lambda} = \lambda$

λ		1/2	1/32
$n=5$	L_1	0.282	0.218
	L_∞	234	0.154
$n=7$	L_1	$0.407 \cdot 10^{-1}$	$0.309 \cdot 10^{-1}$
	L_∞	$0.337 \cdot 10^{-1}$	$0.227 \cdot 10^{-1}$
$n=9$	L_1	$0.546 \cdot 10^{-2}$	$0.439 \cdot 10^{-2}$
	L_∞	$0.452 \cdot 10^{-2}$	$0.329 \cdot 10^{-2}$
$n=11$	L_1	$0.736 \cdot 10^{-3}$	$0.631 \cdot 10^{-3}$
	L_∞	$0.609 \cdot 10^{-3}$	$0.477 \cdot 10^{-3}$

Table 6: Absolute errors for *example 2*; n=length of the support of the wavelet; $h = 1/64$

	λ	L_1	L_∞	L_2
$n=5$	1/2	1.764	1.764	1.764
	1/32	1.762	2.271	2.127
$n=7$	1/2	4.768	4.768	4.768
	1/32	5.886	6.148	6.081
$n=9$	1/2	7.326	7.326	7.326
	1/32	8.817	8.285	8.587
$n=11$	1/2	9.533	9.533	9.533
	1/32	10.744	10.385	10.542

Table 7: Numerical orders for *example 2*; h=1/32

h		1/16	1/32	1/64
$n=3$	L_1	$0.918 \cdot 10^{-1}$	$0.586 \cdot 10^{-1}$	$0.207 \cdot 10^{-1}$
	L_∞	0.591	0.378	0.162
$n=5$	L_1	$0.815 \cdot 10^{-1}$	$0.300 \cdot 10^{-1}$	$0.382 \cdot 10^{-2}$
	L_∞	0.449	0.193	$0.323 \cdot 10^{-1}$
$n=7$	L_1	$0.770 \cdot 10^{-1}$	$0.168 \cdot 10^{-1}$	$0.854 \cdot 10^{-3}$
	L_∞	0.364	0.110	$0.734 \cdot 10^{-2}$
$n=9$	L_1	$0.672 \cdot 10^{-1}$	$0.981 \cdot 10^{-2}$	$0.220 \cdot 10^{-3}$
	L_∞	0.309	$0.683 \cdot 10^{-1}$	$0.195 \cdot 10^{-2}$
$n=11$	L_1	$0.611 \cdot 10^{-1}$	$0.817 \cdot 10^{-2}$	$0.670 \cdot 10^{-4}$
	L_∞	0.271	$0.458 \cdot 10^{-1}$	$0.597 \cdot 10^{-3}$

Table 8: Absolute errors for *example 3*; n=length of the support of the wavelet; $\lambda = 1/2$

h		1/16	1/32	1/64
$n=3$	L_1	0.130	$0.674 \cdot 10^{-1}$	$0.164 \cdot 10^{-1}$
	L_∞	0.574	0.337	0.114
$n=5$	L_1	0.148	$0.497 \cdot 10^{-1}$	$0.357 \cdot 10^{-2}$
	L_∞	0.438	0.200	$0.245 \cdot 10^{-1}$
$n=7$	L_1	0.150	$0.359 \cdot 10^{-1}$	$0.921 \cdot 10^{-3}$
	L_∞	0.392	0.121	$0.696 \cdot 10^{-2}$
$n=9$	L_1	0.129	$0.266 \cdot 10^{-1}$	$0.260 \cdot 10^{-3}$
	L_∞	0.311	$0.699 \cdot 10^{-1}$	$0.194 \cdot 10^{-2}$
$n=11$	L_1	0.125	$0.188 \cdot 10^{-1}$	$0.888 \cdot 10^{-4}$
	L_∞	0.306	$0.477 \cdot 10^{-1}$	$0.641 \cdot 10^{-3}$

Table 9: Absolute errors for *example 3*; n=length of the support of the wavelet; $\lambda = 1/32$

	λ	L_1	L_∞	L_2
$n=3$	1/2	1.350	0.813	1.066
	1/32	2.063	1.467	1.603
$n=5$	1/2	3.053	2.389	2.633
	1/32	3.760	3.253	3.048
$n=7$	1/2	4.312	3.818	4.070
	1/32	5.274	4.172	4.619
$n=9$	1/2	5.449	5.094	5.355
	1/32	6.726	5.206	5.822
$n=11$	1/2	6.920	6.246	6.480
	1/32	7.624	6.356	6.898

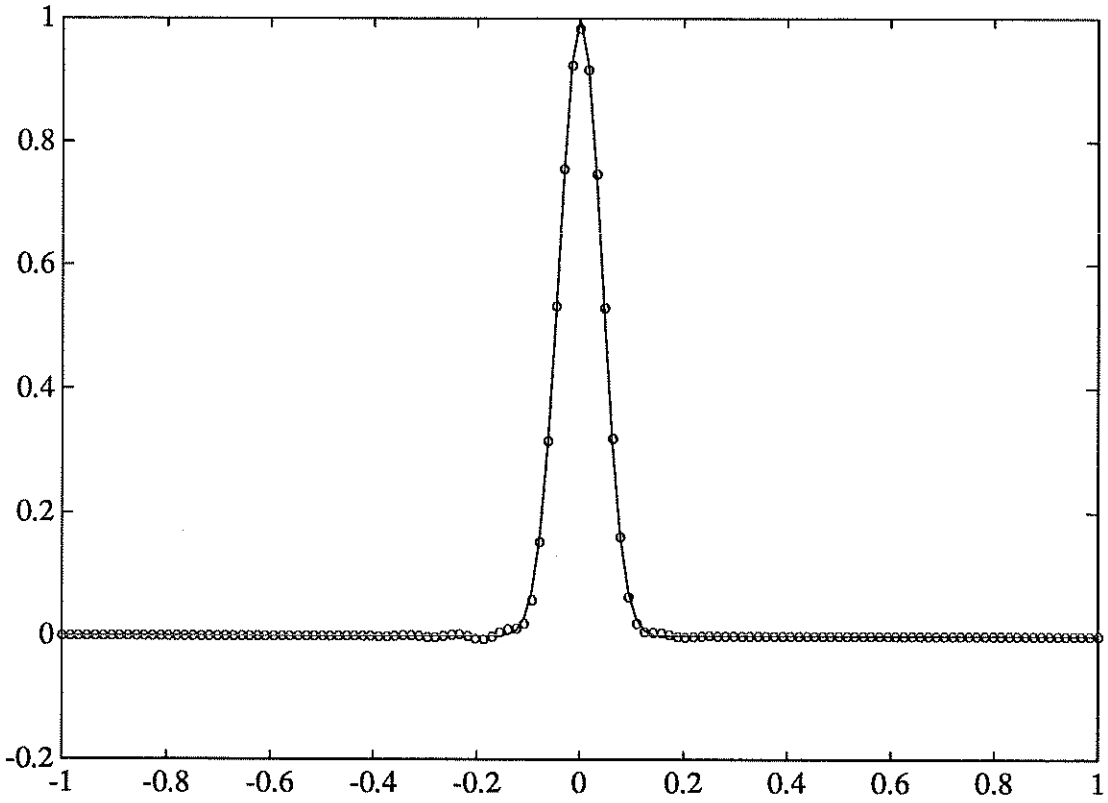
Table 10: Numerical orders for *example 3*; h=1/32

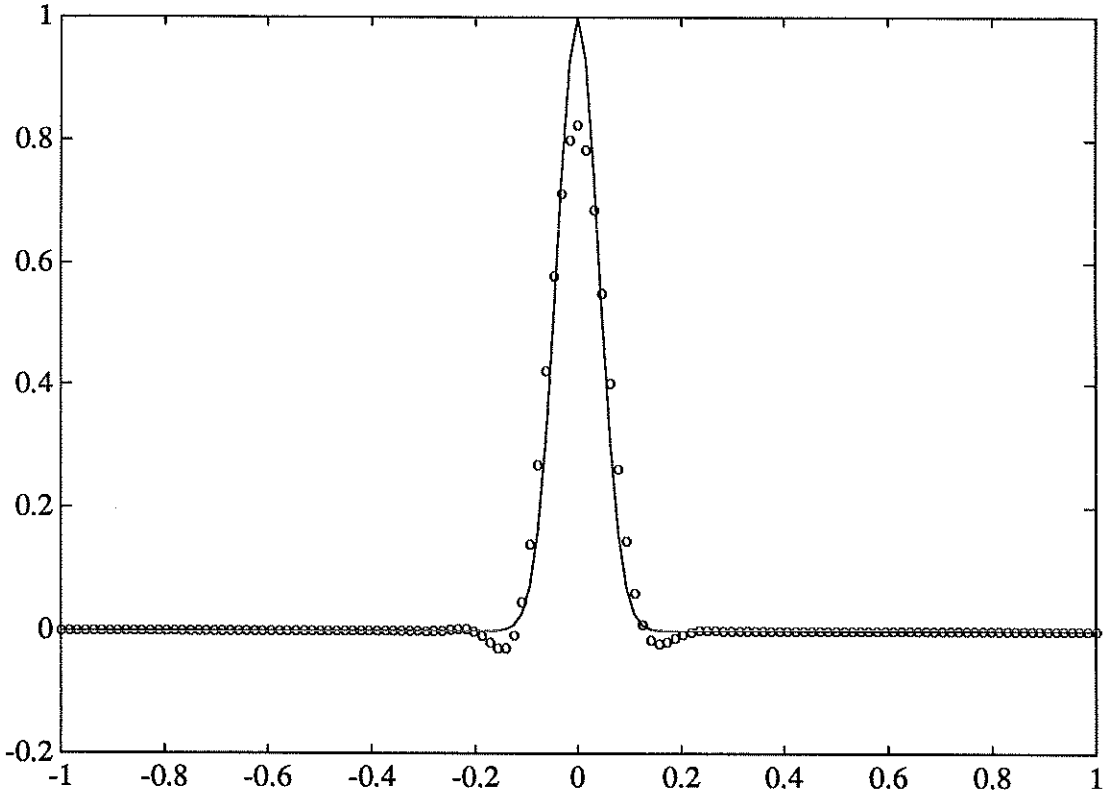
Figure Captions

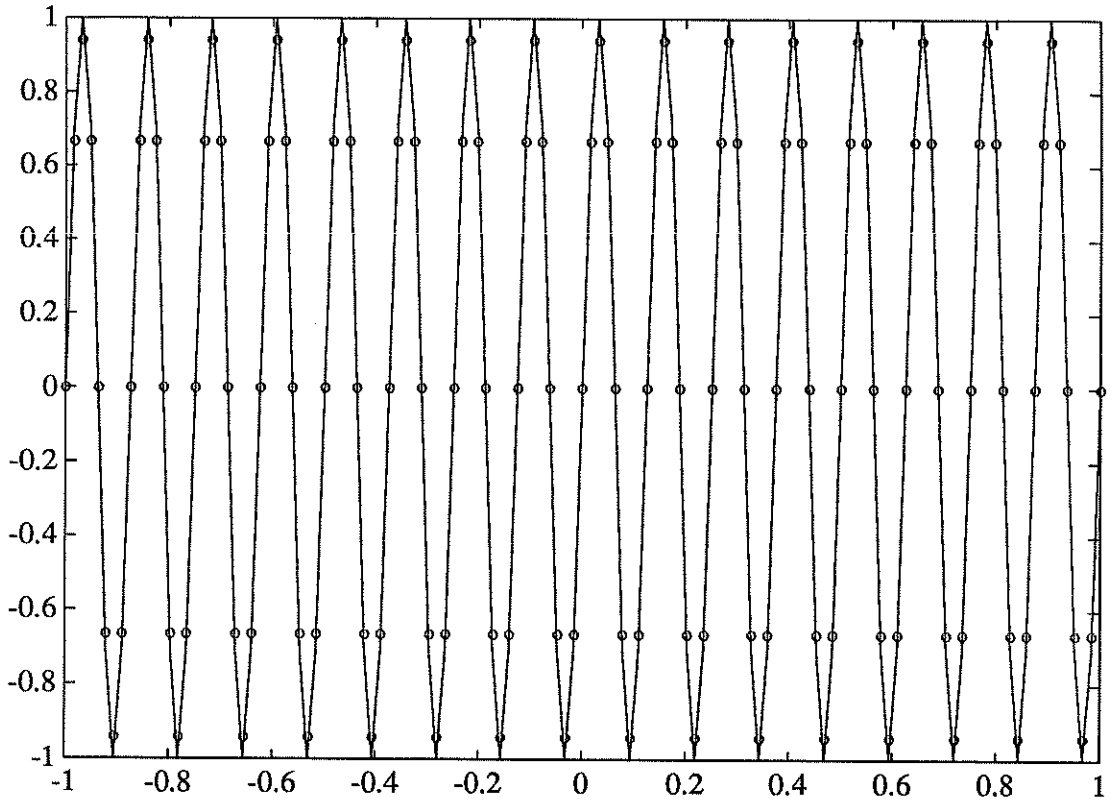
Figure 1 Example 2: $\lambda = \frac{1}{2}; n = 11; h = \frac{1}{64}$; 100 periods (25600 iterations)

Figure 2 Example 3: $\lambda = \frac{1}{4}; n = 3; h = \frac{1}{64}$; 1 period (512 iterations)

Figure 3 Example 3: $\lambda = \frac{1}{4}; n = 11; h = \frac{1}{64}$; 100 periods (51200 iterations)


1
0.8
0.6
0.4
0.2
0
-0.2
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1


1
0.8
0.6
0.4
0.2
0
-0.2
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1