UCLA
COMPUTATIONAL AND APPLIED MATHEMATICS

Fast Multiresolution Algorithms for Solving Linear Equations: A Comparative Study

Francesc Arandiga
Vicente F. Candela
Rosa Donat

December 1992

CAM Report 92-52

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

FAST MULTIRESOLUTION ALGORITHMS FOR SOLVING LINEAR EQUATIONS : A COMPARATIVE STUDY.

FRANCESC ARANDIGA[1]
VICENTE F. CANDELA[2]
ROSA DONAT[3]

DEPARTAMENT DE MATEMÀTICA APLICADA
UNIVERSITAT DE VALÈNCIA

1. Introduction. Fourier transforms are good tools to analyze those operators with coefficients constant in space (or in time, according to the usual signal processing notation). From a numerical point of view, these operators (or their discretizations) are represented by Toeplitz matrices (that is, matrices with constant diagonals), and it is well known that the Fourier transform of many matrices in this class (the special ones called *circulant* matrices, that have some sort of periodicity. For example, the matrices coming from convolution operators) are diagonal matrices. Every Toeplitz matrix can be related to a circulant one and, thus, the Fourier transform simplifies the algorithms involving Toeplitz matrices, such as matrix-vector or matrix-matrix multiplications or inversion of matrices. In this context, we can consider the Fourier transform as a way to obtain a sparse matrix (in fact, a very sparse matrix, because it is diagonal) from a possible dense one.

However, Fourier analysis is not so adequate when the operator does not behave in the same way all over the space. It seems evident that the main remark about Fourier transforms, that is, all circulant matrices of the same order have the same eigenvectors, does not apply here. Matrices representing these operators are no longer Toeplitz and, in the same way as Fourier analysis cannot locate spatial variations, its discrete version cannot diagonalize the matrix (actually, the matrix may even not be diagonalizable). Each matrix needs a different treatment in order to be diagonalized, if possible. It is interesting, though, to have a general analysis which can be applied to a larger class of matrices for practical purposes.

The key point is to realize that the problem of getting a sparse matrix from any given one is similar to that of data compression in signal (or image) processing. In this framework, the natural idea is to analyze the operator in different scales. Sparsity is obtained by neglecting those coefficients corresponding to scales and locations where the operator has very small variation. If the operator is smooth enough, except perhaps in a finite number of points, the corresponding matrix becomes sparse (if the matrix is $N \times N$, it has at most $O(N \log N)$ significant coefficients). The multiscale analysis we are concerned from now on is the *multiresolution* analysis. We refer the reader to [9] for details.

Over the years, some *local* transforms (in contrast to Fourier, which is *global*) have been proposed. Some of them, like the *windowed* or the *Gabor* transform are closely related to Fourier. Nevertheless, the wavelet transform, as stated by Meyer and Grossmann among others, has some advantages with respect to the aforementioned ones and it is more appropiate for frequency analysis (the *windows* are adapted to the frequencies: they are wider as the fequency is lower).

[1] Research supported by a University of Valencia Grant and by ONR Grant N00014-91-J-1034
[2] Research supported by a Grant from DGICYT PS90-0265 and by ONR Grant N00014-91-J-1034
[3] Research supported by a University of Valencia Grant and a grant from DGICYT PS90-0265

In [7], Harten presents a new multiresolution set-up. The basic ideas about wavelets are used in a simple way in order to relate the different scales, but the main difference with the wavelet transform is that the resolution in each scale is obtained by means of interpolatory schemes (point value or cell-average). The better the approximation resulting from the interpolatory schemes, the greater the compression obtained by Harten's multiresolution methods. A two dimensional version of Harten's transform appears in [1]. There, a matrix transform (called *standard multiresolution form* of the matrix because of its obvious parallelism with the *standard wavelet form*) is introduced.

The aim of this paper is to show that the *standard multiresolution form* behaves in most cases in the same way as the standard wavelet form. The structure of these matrices, the compression and the speed of the algorithms, for instance, are similar in both cases. It seems to us, however, that Harten's multiresolution is more natural than the usual wavelet transform in most numerical problems. Interpolatory schemes are required in these problems, and interpolatory multiresolution is an obvious frame to get their multiresolution analysis.

In the next paragraph, we outline the main ideas about interpolatory multiresolution and its related standard matrix form. In §3., we will develop some algorithms, initially appeared in [6], to solve hyperbolic and parabolic partial differential equations and integral equations. Finally, in §4., those algorithms will be applied to some specific problems, and we will draw some conclusions in §5.

2. Multiresolution analysis. We review here some of the results in [7, 8].

From now on, we are going to consider $f(x)$ a periodic function in $[0,1]$ with $N_0 = 2^{n_0}$ subintervals $[x_{j-1}^0, x_j^0]$, $x_j^0 = j \cdot h_0$, $h_0 = \frac{1}{N_0}$, $j = 1, \ldots, N_0$, the set of nested grids $\{x_j^k\}_{j=1}^k$, $x_j^k = 2^k, x_j^0$ $k = 1, \ldots, L < n_0$ and a wavelet function $\varphi(x)$, $\varphi_j^k(x) = 2^{-k}\varphi(2^{-k}x - j)$.

According to the usual notation in wavelet theory,

$$s_j^k = \int f(x)\varphi_j^k(x)dx. \tag{1}$$

The wavelet satisfies a dilation equation,

$$\varphi(x) = 2\sum_j \alpha_j \varphi(2x - j). \tag{2}$$

Thus, once we have the knowledge of the finest scale $\{s_j^0\}$, it is easy to get the information of the coarser grids, $\{s_j^k\}$.

We assume that, for each scale, we have a reconstruction procedure that predicts $f(x)$ from the values $\{s_j^k\}$, which we denote by $R_k(x)$. We impose that the reconstruction is conservative in the sense that:

$$\int R_k(x)\varphi_j^k(x)dx = \int f(x)\varphi_j^k(x)dx = s_j^k \tag{3}$$

If $R_k(x)$ is a good approximation to $f(x)$ for every scale k, the differences

$$Q_k(x) = R_{k-1}(x) - R_k(x), \quad k = 1, \ldots, L \tag{4}$$

must be small.

From the knowledge of the coarsest scale and the differences $Q_k(x)$, we can reconstruct the function in the finest grid:

$$R_0(x) = R_L(x) + \sum_{k=1}^{L} Q_k(x) \tag{5}$$

From a discrete point of view, this means that we can recover the information $\{s_j^0\}$ in the finest grid from the coarsest one $\{s_j^L\}$, and the values $\{d_j^k\}$, where

$$d_j^k = \int Q_k(x)\varphi_j^{k-1}(x)dx = s_j^{k-1} - \int R_k(x)\varphi_j^{k-1}(x)dx \tag{6}$$

The more accurate $R_k(x)$ is, the smaller the absolute values of $\{d_j^k\}$ are. If we truncate to zero the coefficients d_j^k with absolute values below a given thresholding, we get data compression.

As we have just seen this process depends not only on the wavelet but also on the reconstruction procedure. The freedom to choose this reconstruction is the main difference with traditional wavelet methods.

In the case of point value interpolation, the wavelet function $\varphi(x)$ is the Dirac distribution, which verifies the following dilation equation:

$$\delta(x) = 2\delta(2x) \tag{7}$$

In this context, (1) and (3) are equivalent to:

$$s_j^k = \int f(x)\frac{1}{2^k h_0}\delta\left(\frac{2^{-k}x - j}{h_0}\right)dx = f(x_j^k) \tag{8}$$

$$R_k(x_j^k) = f(x_j^k), \quad j = 1, \ldots, N_k \tag{9}$$

We denote $f_j^k = s_j^k$, and $I_k(x) = R_k(x)$ in order to remark the interpolation process. In this case, the reconstruction is just an interpolation of the functions in the points $\{x_j^k\}$.

The previous comments lead us to the following algorithm:

Algorithm 1

Given $\{c_j\}_{j=1}^{N_0}$, and an interpolatory scheme $I_k(x; f)$ for all $k = 1, \ldots, L$ such that $I_k(x_j^k; s_j^k) = s_j^k$, for all $j = 1, \ldots, N_L$, we set:

$$f_j^0 = c_j, \quad 1 \leq j \leq N_0 \tag{10}$$

$$
(11)\qquad \begin{cases} \text{Do for } k = 1, \ldots, L \\ f_j^k = f_{2j}^{k-1}, \quad 1 \le j \le N_k \\ d_j^k = f_{2j-1}^{k-1} - I_k(x_{2j-1}^{k-1}; f_i^k), \quad 1 \le j \le N_k \\ \text{End} \end{cases}
$$

$$
(12)\qquad c^{MR} = \{(d^1, \ldots, d^L), f^L\}
$$

is the multiresolution representation of c.

We can recover c from its multiresolution by means of the following algorithm:

Algorithm 2

$$
(13)\qquad \begin{cases} \text{Do for } k = L, \ldots, 1 \\ f_{2j}^{k-1} = f_j^k, \quad 1 \le j \le N_{k-1} \\ f_{2j-1}^{k-1} = d_j^k + I_k(x_{2j-1}^{k-1}; f_i^k), \quad 1 \le j \le N_k \\ \text{End} \end{cases}
$$

$$
(14)\qquad c = f^0
$$

These multiresolution transformations are similar to Mallat's pyramidal scheme. Given $\{f_j^k\}_{j=1}^{N_k}$ for any $k = 1, \cdots, L$ we can get the samples and the differences in the level $k+1$,

$$
(15)\qquad \{d_j^{k+1}\} = G\{f_j^k\}
$$

$$
(16)\qquad \{f_j^{k+1}\} = H\{f_j^k\}
$$

A similar multiresolution analysis can be done for cell-averages. This case comes by taking $\varphi(x)$ to be the box function, that is:

$$
(17)\qquad \varphi(x) = \begin{cases} 1, & -1 \le x < 0 \\ 0, & \text{otherwise} \end{cases}
$$

$\varphi(x)$ verifies the following dilation equation:

(18) $$\varphi(x) = \varphi(2x) + \varphi(2x-1)$$

The multiresolution analysis

(19) $$\{\{\bar{f}_j^k\}_{j=1}^{N_k}\}_{k=0}^{L}$$

is given by

(20) $$s_j^k = \int f(x)\varphi_j^k(x)dx = \frac{1}{h_k}\int_{x_{j-1}^k}^{x_j^k} f(x)dx$$

We denote $\bar{f}_j^k = s_j^k$, and we call it the cell-average of $f(x)$ in $I_j^k = [x_{j-1}^k, x_j^k]$. As a consequence of 18, these cell-averages are related, as we can see:

(21) $$\bar{f}_j^{k+1} = \frac{1}{2}(\bar{f}_{2j-1}^k + \bar{f}_{2j}^k)$$

If we denote by $A(I)$ the cell-averaging operator,

(22) $$A(I)f = \frac{1}{|I|}\int_I f(x)dx$$

where $|I|$ is the diameter of the *"cell" I*.

In order to apply multiresolution analysis, the reconstruction procedure, $R(x;\bar{f}^k)$, must verify

(23) $$A(I_j^k)R(.;\bar{f}^k) = \bar{f}_j^k$$

The following algorithms are the corresponding to *algorithm 1* and *algorithm 2* for cell-averages.

Algorithm 3

Given $\{c_j\}_{j=1}^{N_0}$, and a cell-average reconstruction scheme $R_k(x;\bar{f})$ for all $k = 1,\ldots,L$ such that $R_k(x_j^k; s_j^k) = s_j^k$, for all $j = 1,\ldots,N_L$, we set:

(24) $$\bar{f}_j^0 = c_j, \quad 1 \le j \le N_0$$

(25)
$$\begin{cases} \text{Do for } k = 1,\ldots,L \\ \bar{f}_j^k = \frac{1}{2}(\bar{f}_{2j-1}^{k-1} + \bar{f}_{2j}^{k-1}), \quad 1 \le j \le N_k \\ d_j^k = f_{2j-1}^{k-1} - A(I_{2j-1}^{k-1})R(.;\bar{f}^k), 1 \le j \le N_k \\ \text{End} \end{cases}$$

$$c^{MR} = \{(d^1, \ldots, d^L), \bar{f}^L\} \tag{26}$$

is the multiresolution representation of c.

The reconstruction algorithm is

Algorithm 4

$$\begin{cases} \text{Do for } k = L, \ldots, 1 \\ \bar{f}^{k-1}_{2j-1} = A(I^{k-1}_{2j-1})R(.; \bar{f}^k) + d^k_j, \quad 1 \le j \le N^k \\ \bar{f}^{k-1}_{2j} = 2\bar{f}^k_j - \bar{f}^{k-1}_{2j-1}, \quad 1 \le j \le N_k \\ \text{End} \end{cases} \tag{27}$$

$$c = \bar{f}^0 \tag{28}$$

The simplest case is when the point value or cell-average interpolation is centered and symmetric around the node. Then, the operators I_k and A can be evaluated by the following relations:

$$I_k(x^{k-1}_{2j-1}; f^k_i) = \sum_{l=1}^{s} \beta_l (f^k_{j+l-1} + f^k_{j-l}) \tag{29}$$

where

$$\begin{cases} r = 2 \Rightarrow \beta_1 = -\frac{1}{2} \\ r = 4 \Rightarrow \beta_1 = \frac{9}{16}, \beta_2 = -\frac{1}{16} \\ r = 6 \Rightarrow \beta_1 = \frac{150}{256}, \beta_2 = -\frac{25}{256}, \beta_3 = \frac{3}{256} \end{cases} \tag{30}$$

The order of the interpolation is $s = 2r$.

$$\begin{aligned} A(I^{k-1}_{2j-1})R(.; \bar{f}^k) &= \bar{f}^k_j + \textstyle\sum_{l=1}^{s} \gamma_l (\bar{f}^k_{j+l} - \bar{f}^k_{j-l}) \\ A(I^{k-1}_{2j})R(.; \bar{f}^k) &= \bar{f}^k_j - \textstyle\sum_{l=1}^{s} \gamma_l (\bar{f}^k_{j+l} - \bar{f}^k_{j-l}) \end{aligned} \tag{31}$$

where

$$\begin{cases} r = 3 \Rightarrow \gamma_1 = -\frac{1}{8} \\ r = 5 \Rightarrow \gamma_1 = -\frac{22}{128}, \gamma_2 = -\frac{3}{128} \end{cases} \tag{32}$$

The order of the interpolation is $s = 2r + 1$.

We denote by v^{MR} the multiresolution form of the vector v either in point value interpolation or in cell-average according to the context.

A two dimensional version of these algorithms can be found in [1]. Given a matrix A, we look for its standard form, that is the matrix A^s such that

$$(Af)^{MR} = A^s f^{MR}, \text{ for every vector } f \tag{33}$$

Introducing the invertible operator M such that $f^{MR} = Mf$ for every vector f, the standard form of the matrix is obtained by the following change of basis:

$$A^s = MAM^{-1} \tag{34}$$

Algorithms 1 and 3 represent the operator M in point value and cell-average interpolation, respectively, and algorithms 2 and 4 are the inverses of M in both cases. Therefore, (34) is equivalent, in point value (respectively, in cell average) to apply algorithm 1 (resp. algorithm 2) to the rows of A and the transposed of algorithm 3 (resp. algorithm 4) to the new columns (or *vice versa*). It is remarkable the fact that, unlike the wavelet standard form, this multiresolution standard form is not symmetric, and, so, we do not apply the same algorithm to the rows and to the columns of the matrix. We develop here the transformation (34), when the interpolation is centered around the node we want to interpolate and we assume periodicity. Some changes must be introduced if any of these conditions fails.

In the case of point value interpolation, we have

Algorithm 5

For $j = 1, \cdots, N$

$$f_j(i) = a_{i,j}, \quad 1 \le i \le N \tag{35}$$

$$b_{*,j} = f_j^{MR} \tag{36}$$

End
For $i = 1, \cdots, N$

$$g_i^0(j) = b_{i,j}, \quad 1 \le j \le N \tag{37}$$

$$\begin{cases} \text{Do for } k = 1, \ldots, L \\ s_i^k(j) = g_i^{k-1}(2j-1), \quad 1 \le j \le N_k \\ g_i^k(j) = g_i^{k-1}(2j) + I_k(x_{2j}^{k-1}, s_i^k), \quad 1 \le j \le N_k \end{cases} \tag{38}$$

End

(39) $$c_{i,*} = \{(s_j^1)_{j=1}^{N_1}, \ldots, (s_j^L)_{j=1}^{N_L}, (g_j^L)_{j=1}^{N_L}\}$$

(40) $$A^s = (c_{i,j})_{i,j=1}^{N}$$

In the case of cell-average, the algorithm is the following:

Algorithm 6

For $j = 1, \cdots, N$

(41) $$f_j(i) = a_{i,j}, \quad 1 \leq i \leq N$$

(42) $$b_{*,j} = f_j^{MR}$$

End
For $i = 1, \cdots, N$

(43) $$g_i^0(j) = b_{i,j}, \quad 1 \leq j \leq N$$

(44)
$$\begin{cases} \text{Do for } k = 1, \ldots, L \\ s_i^k(j) = g_i^{k-1}(2j-1) - g_i^{k-1}(2j), \quad 1 \leq j \leq N_k \\ g_i^k(j) = g_i^{k-1}(2j-1) + g_i^{k-1}(2j) - A(I_{2j-1}^{k-1})R(.; s^k) + s_i^k(j) \quad 1 \leq j \leq N_k \end{cases}$$

End

(45) $$c_{i,*} = \{(s_j^1)_{j=1}^{N_1}, \ldots, (s_j^L)_{j=1}^{N_L}, (g_j^L)_{j=1}^{N_L}\}$$

(46) $$A^s = (c_{i,j})_{i,j=1}^{N}$$

In a similar way as the one dimensional case, when the interpolatory scheme is good enough, most of the coefficients in the standard form are small, and we can get data compression by discarding those coefficients with absolute value below a given threshold.

In the next sections we are going to display some examples where we use the standard multiresolution form, in order to show the liability of the previous algorithms.

3. The basic Algorithm. We shall apply compression algorithms to problems that, after the discretization process is complete, have the following form:

$$u^{n+1} = Au^n + f \tag{47}$$

with u^0 and $f \in R^N$ given vectors. This formula admits a closed form solution given by

$$u^n = A^n u^0 + \sum_{j=0}^{n-1} A^j f \tag{48}$$

which can be used to compute the solution $A^n u^0$, for $f = 0$ in $\log n$ steps, ($n = 2^m, m$ integer; here and throughout $\log n = \log_2 n$) by repeated squaring of A: A, A^2,A^4, A^8, ..., A^{2^m} (*see*[6]).

The later squarings would involve almost dense matrices even if we started with a very sparse A (as it is the case for discretizations of partial differential equations) and the algorithm, as stated, is useless. However, for an appropriate representation of A in a multiresolution basis, all the powers A^j may be approximated by sparse matrices. Note that all powers are equally sparse since the kernel they are approximating satisfies uniform estimates ([6]). Then, the algorithm of repeated squaring should be advantageous.

To compute the closed form solution (48), we consider the following algorithm (see [6]):

$$\begin{aligned} B &:= MAM^{-1} \\ C &:= I \end{aligned}$$

$$\left.\begin{aligned} C &:= TRUNC(C + BC, \epsilon) \\ B &:= TRUNC(BB, \epsilon) \end{aligned}\right\} \text{ repeat m times} \tag{49}$$

$$u^n := M^{-1}(BMu^0 + CMf)$$

The matrix M corresponds to a fast multiresolution transform and the truncation operator sets elements in a matrix to zero if their absolute value is below the given threshold ϵ

$$\tilde{A} = TRUNC(A, \epsilon) := \left\{ \begin{array}{lcll} \tilde{a}_{ij} & = & a_{ij} & |a_{ij}| \geq \epsilon \\ \tilde{a}_{ij} & = & 0 & |a_{ij}| < \epsilon \end{array} \right. \tag{50}$$

For $\epsilon = 0$, (49) is equivalent to (48) because

$$(I + B)(I + B^2)\cdots(I + B^{2^{m-1}}) = \sum_{j=0}^{2^m - 1} B^j. \tag{51}$$

They are not equivalent for $\epsilon > 0$, but by choosing ϵ small enough the result of (49) can be arbitrarily close to (48) (see [6]).

In this paper the matrix M corresponds to the multiresolution analysis based on fourth order compactly supported wavelets, fifth order cell average interpolation and sixth order point value interpolation. In order to keep symmetry, we have to choose different orders of approximation in both interpolations. Though the results depend on these orders, this dependence is not as strong as to lead to different conclusions.

We shall apply the algorithm to compute approximate solutions to hyperbolic and parabolic problems in one space dimension, and to some integral equations.

4. Numerical experiments.

4.1. Hyperbolic problems. Consider the following scalar hyperbolic problem:

$$\partial_t u + a(x)\partial_x u = f(x) \tag{52}$$

$$u(x,0) = u_0(x) \tag{53}$$

with periodic boundary conditions and the following choices:

$$a(x) = 0.5 + 0.115\sin(4\pi x) \tag{54}$$

$$f(x) = \cos(4\pi x) \tag{55}$$

$$u_0(x) = \sin(4\pi x) \tag{56}$$

The basic finite difference schemes for the discretization of the PDE are obtained as follows (see [6]): In each interval

$$I_{j-\frac{1}{2}} = \{x \ : \ (j-1)h \le x \le jh\} \tag{57}$$

we construct a polynomial of degree k that interpolates the two points (x_{j-1}, u^n_{j-1}), (x_j, u^n_j) and $k-1$ of its neighbors. If k is even these interpolation points go from $x_{j-\frac{k}{2}}$ to $x_{j+\frac{k}{2}}$. If k is odd, they go from $x_{j-\frac{k-1}{2}}$ to $x_{j+\frac{k-1}{2}}$. The procedure gives a reconstruction function, $R^{n,k}$, which is a polynomial of degree k in each $I_{j-\frac{1}{2}}$ and is continuous but generally not differentiable at the boundary points x_{j-1} and x_j.

To approximate the solution of (52) at the grid points (x_j, t^{n+1}) we solve the equation "exactly" with initial data

$$u_h(x,t^n) = R^{n,k}(x) \tag{58}$$

form $t^n \le t \le t^{n+1}$, evaluate the solution at (x_j, t^{n+1}), and set $u^{n+1}_j = u_h(x_j, t^{n+1})$. We require the usual Courant condition $\Delta t|a(x)| < h$.

In the special case when $a(x) = a$, constant, $f = 0$ and $k = 1$ we recover the first order accurate upwind difference scheme. For $k = 2$ we get the classical Lax-Wendroff second order accurate three point scheme. For k=3,4,5 the schemes are known to be L_2 stable.

For variable coefficients the approximation becomes

$$u_h(x_j, t^{n+1}) = R^{n,k}(x_j(t^n)) + \int_{t^n}^{t^{n+1}} f(x_j(t^{n+1}-s))ds \tag{59}$$

where $x_j(t)$ solves

$$\frac{dx_j}{dt} = a(x_j), \qquad t^n \le t \le t^{n+1} \tag{60}$$

$$x_j(t^{n+1}) = x_j \tag{61}$$

As in [6], a fourth order Runge-Kutta method is used to integrate the O.D.E. and Simpson's rule is used to evaluate the integral. The result of this approximation to the right side of (59) is defined to be u^{n+1}_j.

We should remark here that, because of the spatial dependence of the coefficients of the differential operator, the fast Fourier transform is not an efficient algorithm for these type of problems.

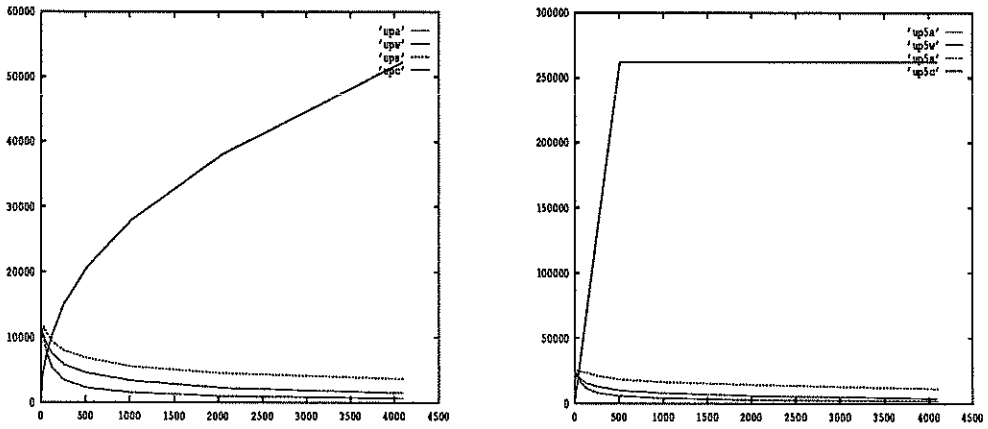

FIG. 1. *upwind, N=256, N=512*

r	$\|\|u-u_w\|\|_\infty$	$\|\|u-u_s\|\|_\infty$	$\|\|u-u_c\|\|_\infty$	$\|\|u-u_d\|\|_\infty$
1	.0402	.0478	.0442	.0711
2	.0052	.0101	.0124	.00079
3	.0067	.0105	.0124	.00079

TABLE 1
Hyperbolic equation, L_∞ error, t=1, N=512

r	$\|\|u-u_w\|\|_1$	$\|\|u-u_s\|\|_1$	$\|\|u-u_c\|\|_1$	$\|\|u-u_d\|\|_1$
1	.0173	.0239	.0254	.0430
2	.0024	.0050	.0072	.00032
3	.0023	.0053	.0072	.0000063

TABLE 2
Hyperbolic equation, L_1 error, t=1, N=512

Numerical results are compiled in tables 1 and 2 and in figures 1, 2, 3, 4 and 5. The tables show numerical errors with respect to the exact solution. Here $\Delta t/h = 1$. In each table, u represents the exact solution and u_w, u_s, u_c the solutions produced by the multiresolution frameworks based on wavelets, point-value interpolation and cell-average reconstruction.

Although the solution obtained with the wavelet algorithm is more accurate in the L_∞ norm (in this particular case), the overall quality of all the approximations (i.e. the L_1 norm) is about the same.

Figures 4 and 5 show, for comparison purposes, the standard forms of the matrices A^8 corresponding to the third order method. We can observe the same finger-like structure in each case.

Figures 1,2 and 3 show the growth pattern of the number of nonzero elements in $MA^{2^k}M^{-1}$. Again, we observe the same behavior for each one of the multiresolution transforms.

4.2. Parabolic problems. We choose the following parabolic problem:

$$\partial_t u = \partial_x(a(x)\partial_x u) + f(x) \tag{62}$$

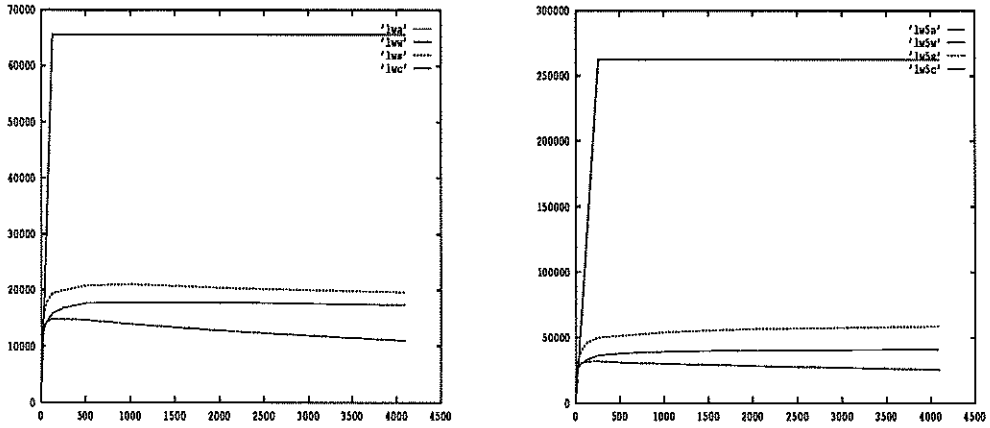

FIG. 2. *Second order, N=256, N=512*

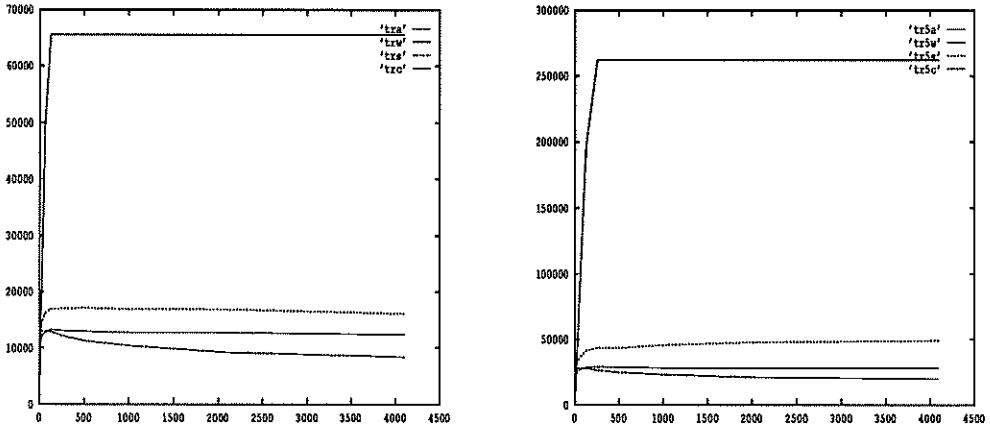

FIG. 3. *Third order, N=256, N=512*

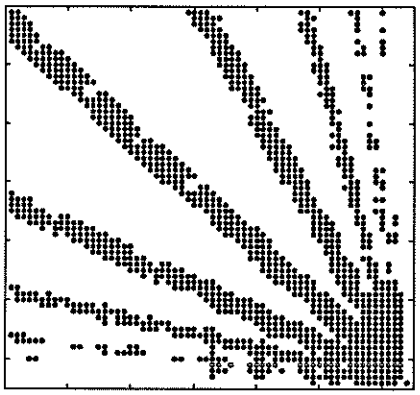

FIG. 4. *Third order, wavelets*

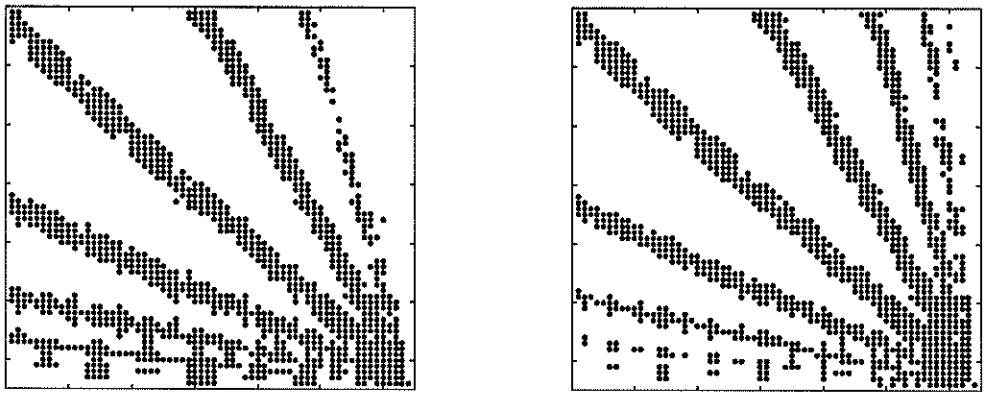

FIG. 5. *Third order, pointvalues and cell-averages*

norm	t	2^m	$\|\|u-u_w\|\|$	$\|\|u-u_s\|\|$	$\|\|u-u_c\|\|$	$\|\|u-u_d\|\|$
∞	.00049	512	.0819	.0971	.0857	.000075
1	.00049	512	.0395	.0467	.0233	.00004179

TABLE 3

L_∞ and L_1 errors, N=512, parabolic equation $f(x)=0$

norm	t	2^m	$\|\|u-u_w\|\|_1$	$\|\|u-u_s\|\|_1$	$\|\|u-u_c\|\|_1$	$\|\|u-u_d\|\|_1$
∞	.0039	4096	.29	.53	.20	.00029
1	.0039	4096	.16	.23	.089	.00015

TABLE 4

L_∞ and L_1 errors, N=512, parabolic equation $f(x)=0$

norm	t	2^m	$\|\|u-u_w\|\|$	$\|\|u-u_s\|\|$	$\|\|u-u_c\|\|$	$\|\|u-u_d\|\|$
∞	.00049	512	.040	.038	.0857	.000163
1	.00049	512	.0200	.012	.0125	.0000440

TABLE 5

L_∞ and L_1 errors, N=512, parabolic equation $f(x)\neq 0$

norm	t	2^m	$\|\|u-u_w\|\|$	$\|\|u-u_s\|\|$	$\|\|u-u_c\|\|$	$\|\|u-u_d\|\|$
∞	.0039	4096	.21	.22	.13	.00033
1	.0039	4096	.08	.069	.049	.00015

TABLE 6

L_∞ and L_1 errors, N=512, parabolic equation $f(x)\neq 0$

$$u(x,0) = u_0(x) \tag{63}$$

with periodic boundary conditions and the following choices:

$$a(x) = 0.5 + 0.25\sin(2\pi x) \tag{64}$$

$$f(x) = 0 \quad \text{and} \tag{65}$$

$$f(x) = -\pi^2\cos(2\pi x)^2 + \pi^2(a(x))\sin(2\pi x) \tag{66}$$

$$u_0(x) = \sin(4\pi x) \tag{67}$$

The discrete setting is the same as in the hyperbolic problem. Here we will use the simple central difference scheme

$$u_j^{n+1} = u_j^n + \frac{\Delta t}{h^2}\Delta_-(a(x_j)\Delta_+ u_j) + \Delta t f(x_j) \tag{68}$$

where

$$\Delta_\pm u_j = \pm(u_{j\pm1} - u_j) \tag{69}$$

with $\Delta t/h^2 = 0.25$.

Numerical results for the case $f(x)=0$ are compiled in tables 3 and 4. We observe that the errors in the L_1 norm are comparable for all the three cases at hand. The same conclusions are drawn from tables 5 and 6.

Figure 6 shows the growth of nonzero elements in each of the standard forms and figures (7) and 8 the pattern of significant elements of MA^8M^{-1} . In each case, the behavior is the same.

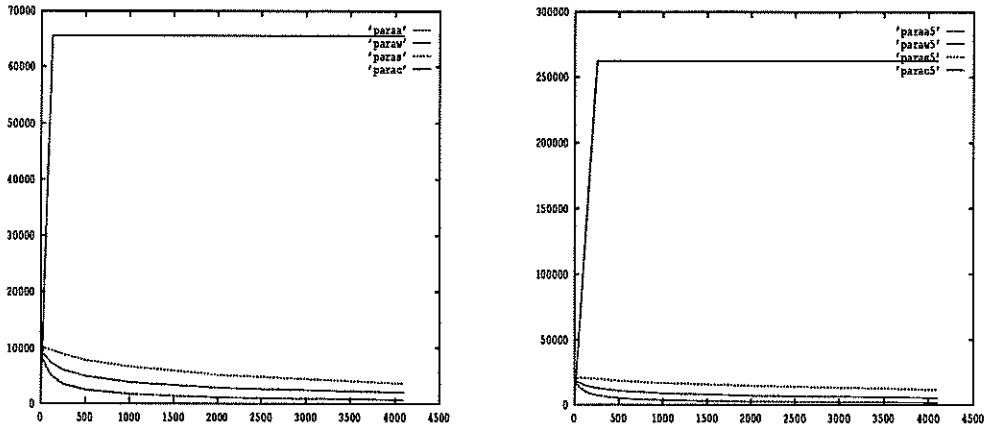

FIG. 6. *Parabolic Equation, N=256, N=512*

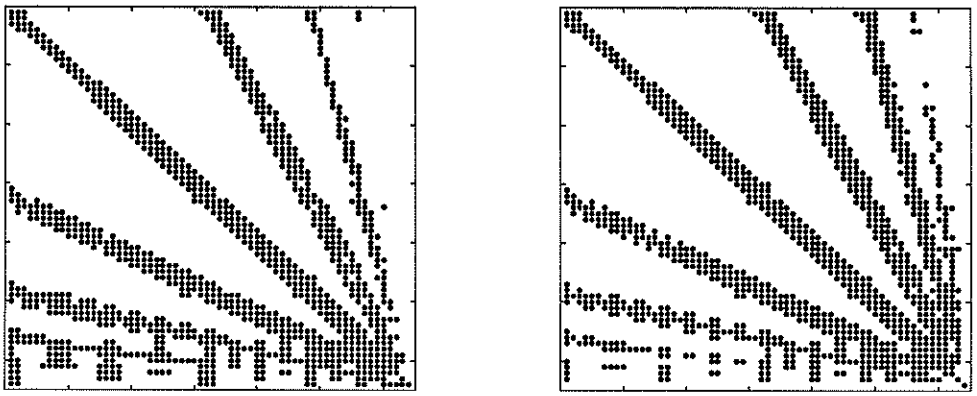

FIG. 7. *Parabolic Equation, pointvalues and cellaverages*

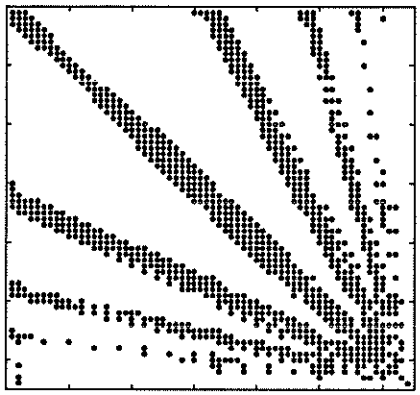

FIG. 8. *Parabolic Equation, wavelets*

4.3. Integral Equations. Our goal now is to compute approximations to the solution of integral equations. We shall start with the easiest type:

$$u(x) = \int_a^b K(x,y)v(y)dy \tag{70}$$

If $K(x,y)$ is a smooth kernel, there is an obvious way to discretize the equation:

$$u(x_i) = \int_a^b K(x_i,y)v(y)dy \tag{71}$$

$$= \sum_{j=1}^{N} a_{ij}v(x_j) + O(h^p) \tag{72}$$

where p is the order of the integration and $h = \max\{|x_{i+1} - x_i|,\ i = 1,\ldots,N-1\}$. Then

$$\bar{u} = A\bar{v}, \tag{73}$$

where

$$\bar{u} = (u(x_1), u(x_2), \ldots, u(x_N)) + O(h^p) \tag{74}$$

$$\bar{v} = (v(x_1), v(x_2), \ldots, v(x_N)) \tag{75}$$

$$A = (a_{ij}) \tag{76}$$

This matrix-vector multiplication can be computed by first computing the standard form of the matrix, then compressing the vector $\bar{v}$ and then performing the multiplication. There are other ways to perform the discretization procedure, however we must remark that our main objective is not the study of the different discretizations, but the comparative behavior of the different multiresolution transforms applied to these problems. From our point of view, any consistent discretization of an integral equation leads to a linear system of the form

$$u = Av \tag{77}$$

where A is, usually, a dense matrix. This system is equivalent to

$$Mu = MAM^{-1}Mv \tag{78}$$

for any invertible matrix M. When M corresponds to a fast multiresolution transform, the matrix $MAM^{-1} = A^s$ (see (34)) has only $O(NlogN)$ elements and the vector Mv has only $O(logN)$ nonzero entries. The main advantage is that, once MAM^{-1} has been computed, one has a fast algorithm to compute integral transforms (with the same kernel) of different functions.

We have performed numerical tests on three integral equations studied in [5]. We start with

$$\text{Example 1:} \qquad u(x) = \int_0^1 cos(x-y)sin^2(y)dy \tag{79}$$

comments on tables etc.

When the integral kernel $K(x,y)$ is not smooth, one has to be a little more careful with the discretization procedure. We discretize following Atkinson [2], and perform several tests on the equation

$$\text{Example 2} \qquad u(x) = \int_0^1 log|x-y|(y-y^2)dy \tag{80}$$

This discretization turns out to be such that the error is of order $= O(1/N^2)$.

Numerical results for these two cases are compiled in tables 7 to 18. Here we also study the behavior of the approximation for different tolerances.

The original design of the multiresolution transforms assumes periodicity of the vectors to be compressed. If this is not the case, the compressed version of a vector still has relatively many non-zero components, that account for its lack of periodicity. Harten's multiresolution set-up can be adapted to work with non-periodic boundaries. In the tables, u_{bs} and u_{bc} represent the solutions obtained with the algorithms based on pointvalue interpolation and cell average reconstruction modified so that no periodicity is assumed. Thus, the number of nonzero elements in the compressed vector is considerably reduced. These modifications reduce the order of the interpolation at the boundaries and, as a consequence, their behavior is a bit different.However, they produce good quality approximations for low values of ϵ, the tolerance.

These problems do not fit into the general framework of section 3. They are easier and do not involve any iterative process. Fredholm integral equations of the second kind however, are usually approximated by an iterative procedure that follows the framework of (47). We shall apply these compression techniques to the following particular case:

$$\text{Example 3:} \qquad u(x) = f(x) + \frac{1}{4}\int_0^1 \log|x-y|u(y)dy \tag{81}$$

where $f(x)$ is chosen so that $u(y) = y(1-y)$.

Now, u^n is an approximation to the solution of the integral equation that should improve with n. Moreover, the iterates are closer to the limit if the initial vector is close to the limit. These two observation set the basis for a very easy acceleration procedure with no additional cost. Note that

$$u^n = A^n u^0 + \sum_{j=1}^{n-1} A^j f,$$

N	$\|\|u-u_w\|\|$	$\|\|u-u_c\|\|$	$\|\|u-u_p\|\|$	$\|\|u-u_{bp}\|\|$	$\|\|u-u_{bc}\|\|$
128	$2.1 \cdot 10^{-5}$	$2.5 \cdot 10^{-5}$	$6.1 \cdot 10^{-6}$	$5.6 \cdot 10^{-5}$	$1.6 \cdot 10^{-4}$
256	$6.0 \cdot 10^{-6}$	$2.5 \cdot 10^{-5}$	$6.1 \cdot 10^{-6}$	$5.9 \cdot 10^{-5}$	$3.4 \cdot 10^{-4}$
512	$2.4 \cdot 10^{-6}$	$2.5 \cdot 10^{-5}$	$6.0 \cdot 10^{-6}$	$6.0 \cdot 10^{-5}$	$7.8 \cdot 10^{-4}$

TABLE 7

Integral Equations. Example 1. L_1 errors tol= 10^{-4}

	W	C	P	BP	BC	N
nzv	40	24	13	15	30	128
nzm	861	3022	3577	1144	1200	128
nzv	48	24	13	15	30	256
nzm	1349	6930	8439	2284	2274	256
nzv	56	24	13	15	30	512
nzm	1475	9666	19268	4016	2288	512

TABLE 8

Integral Equations. Example 1. tol= 10^{-4}

N	$\|\|u-u_w\|\|$	$\|\|u-u_c\|\|$	$\|\|u-u_p\|\|$	$\|\|u-u_{bp}\|\|$	$\|\|u-u_{bc}\|\|$
128	$2.7 \cdot 10^{-5}$	$2.9 \cdot 10^{-5}$	$2.0 \cdot 10^{-5}$	$1.8 \cdot 10^{-4}$	$3.5 \cdot 10^{-4}$
256	$2.4 \cdot 10^{-5}$	$2.6 \cdot 10^{-5}$	$1.9 \cdot 10^{-5}$	$1.9 \cdot 10^{-4}$	$7.7 \cdot 10^{-4}$
512	$2.4 \cdot 10^{-5}$	$2.6 \cdot 10^{-5}$	$2.4 \cdot 10^{-5}$	$1.9 \cdot 10^{-4}$	$1.5 \cdot 10^{-3}$

TABLE 9

Integral Equations. Example 1. L_1 errors tol= 10^{-3}

	W	C	P	BP	BC	N
nzv	28	16	9	12	18	128
nzm	714	2500	3551	1009	783	128
nzv	28	16	9	12	18	256
nzm	1054	5004	8363	2013	1393	256
nzv	30	16	9	12	18	512
nzm	1475	9666	19268	4016	2288	512

TABLE 10

Integral Equations. Example 1. tol= 10^{-3}

N	$\|\|u-u_w\|\|$	$\|\|u-u_c\|\|$	$\|\|u-u_p\|\|$	$\|\|u-u_{bp}\|\|$	$\|\|u-u_{bc}\|\|$
128	$3.3 \cdot 10^{-4}$	$2.0 \cdot 10^{-4}$	$3.0 \cdot 10^{-4}$	$3.2 \cdot 10^{-3}$	$1.6 \cdot 10^{-2}$
256	$2.8 \cdot 10^{-4}$	$1.6 \cdot 10^{-4}$	$3.1 \cdot 10^{-4}$	$3.2 \cdot 10^{-3}$	$3.6 \cdot 10^{-2}$
512	$2.6 \cdot 10^{-4}$	$2.5 \cdot 10^{-3}$	$3.1 \cdot 10^{-4}$	$3.3 \cdot 10^{-3}$	$9.3 \cdot 10^{-2}$

TABLE 11

Integral Equations. Example 1. L_1 errors tol= 10^{-2}

	W	C	P	BP	BC	N
nzv	16	8	5	7	11	128
nzm	565	1154	3231	765	797	128
nzv	16	8	5	7	11	256
nzm	831	1940	7554	1520	534	256
nzv	16	7	5	7	10	512
nzm	1183	2922	17314	3024	796	512

TABLE 12

Integral Equations. Example 1. tol= 10^{-2}

N	$\|\|u-u_w\|\|$	$\|\|u-u_c\|\|$	$\|\|u-u_p\|\|$	$\|\|u-u_{bp}\|\|$	$\|\|u-u_{bc}\|\|$
128	$3.5 \cdot 10^{-5}$	$5.8 \cdot 10^{-5}$	$9.3 \cdot 10^{-5}$	$7.2 \cdot 10^{-5}$	$1.1 \cdot 10^{-4}$
256	$6.8 \cdot 10^{-5}$	$2.3 \cdot 10^{-5}$	$1.7 \cdot 10^{-4}$	$1.4 \cdot 10^{-4}$	$3.2 \cdot 10^{-3}$

TABLE 13

Integral Equations. Example 2. L_1 errors tol= 10^{-4}

	W	C	P	BP	BC	N
nzv	25	24	23	3	4	128
nzm	3030	3946	4964	3940	3623	128
nzv	28	27	27	3	4	256
nzm	5177	6764	10926	9201	6805	256

TABLE 14

Integral Equations. Example 2. tol= 10^{-4}

N	$\|\|u-u_w\|\|$	$\|\|u-u_c\|\|$	$\|\|u-u_p\|\|$	$\|\|u-u_{bp}\|\|$	$\|\|u-u_{bc}\|\|$
128	$5.4 \cdot 10^{-4}$	$2.8 \cdot 10^{-3}$	$1.7 \cdot 10^{-3}$	$2.1 \cdot 10^{-3}$	$8.6 \cdot 10^{-3}$
256	$5.7 \cdot 10^{-4}$	$5.6 \cdot 10^{-3}$	$5.8 \cdot 10^{-3}$	$5.0 \cdot 10^{-3}$	$4.4 \cdot 10^{-2}$

TABLE 15

Integral Equations. Example 2. L_1 errors tol= 10^{-3}

	W	C	P	BP	BC	N
nzv	20	19	19	3	4	128
nzm	1329	1545	3130	2730	1591	128
nzv	23	20	21	3	3	256
nzm	1547	2189	4703	4322	2466	256

TABLE 16

Integral Equations. Example 2. tol= 10^{-2}

N	$\|\|u-u_w\|\|$	$\|\|u-u_c\|\|$	$\|\|u-u_p\|\|$	$\|\|u-u_{bp}\|\|$	$\|\|u-u_{bc}\|\|$
128	$3.6 \cdot 10^{-3}$	$5.5 \cdot 10^{-2}$	$8.6 \cdot 10^{-2}$	$1.0 \cdot 10^{-1}$	$5.9 \cdot 10^{-2}$
256	$6.0 \cdot 10^{-3}$	$2.6 \cdot 10^{-1}$	$1.9 \cdot 10^{-1}$	$2.1 \cdot 10^{-2}$	$3.1 \cdot 10^{-1}$

TABLE 17

Integral Equations. Example 2. L_1 errors tol= 10^{-2}

	W	C	P	BP	BC	N
nzv	9	7	7	3	3	128
nzm	247	21	539	690	112	128
nzv	11	7	7	3	3	256
nzm	225	0	359	108	663	256

TABLE 18

Integral Equations. Example 2. tol= 10^{-3}

initial vec.	iter. algorith 1	iter. algorith 2	$\|\cdot\|_1$ $-error$	$\|\cdot\|_\infty$ $-error$
f	1	0	$0.9741 \cdot 10^{-2}$	$0.1165 \cdot 10^{-1}$
	2	0	$0.1422 \cdot 10^{-2}$	$0.1699 \cdot 10^{-2}$
	3	0	$0.2859 \cdot 10^{-4}$	$0.8056 \cdot 10^{-4}$
	4	0	$0.9551 \cdot 10^{-5}$	$0.6040 \cdot 10^{-4}$
	1	3	$0.3001 \cdot 10^{-4}$	$0.9725 \cdot 10^{-4}$
	2	2	$0.2744 \cdot 10^{-5}$	$0.4142 \cdot 10^{-5}$
	3	1	$0.9537 \cdot 10^{-5}$	$0.6044 \cdot 10^{-5}$
1	1	0	$0.1168 \cdot 10^{-0}$	$0.1356 \cdot 10^{-0}$
	2	0	$0.1702 \cdot 10^{-1}$	$0.2019 \cdot 10^{-1}$
	3	0	$0.3612 \cdot 10^{-3}$	$0.4372 \cdot 10^{-3}$
	4	0	$0.9537 \cdot 10^{-5}$	$0.6040 \cdot 10^{-4}$
	1	3	$0.3659 \cdot 10^{-3}$	$0.4361 \cdot 10^{-3}$
	2	2	$0.8925 \cdot 10^{-5}$	$0.6438 \cdot 10^{-4}$
	3	1	$0.9493 \cdot 10^{-5}$	$0.6054 \cdot 10^{-4}$

TABLE 19
Integral ec. Example 3 ; N= 128 tol= 10^{-5}, u_3

and the second term on the right hand side of the equation above does not depend on the initial vector chosen for the iterative procedure. Thus, we can start by choosing any arbitrary vector u^0, say $(1, 1, \ldots, 1)$ and compute the n-th iterate with $n = 2^m$ by using algorithm (49) m times. Obviously $b = u^n$ is a much better choice than u^0. We can now compute again the n-th iterate with initial vector b without repeating the algorithm m times as before. In fact the cost of this step is just the cost of multiplying a compressed vector by the compressed from of a matrix which has been computed already.

Numerical results are compiled in Tables 19 and 20. Table 19 uses the multiresolution transform associated to Harten's interpolatory set-up. Table 20 the wavelet multiresolution transform. Algorithm 1 refers to (49) algorithm 2 refers to the acceleration procedure described above. We would like to remark that 2 iterations of algorithm 1 (that involve 2 matrix multiplications) followed by two iterations of algorithm 2 (the acceleration procedure, that only involves matrix-vector multiplications of a compressed matrix by a compressed vector) gives the same accuracy as 4 iterations of the basic algorithm (49) (that, of course would involve 4 matrix multiplications, even though these matrices are sparse).

5. Conclusions. Along the previous examples, it seems clear that the behavior of both the traditional wavelet analysis and the interpolatory multiresolution (in its point value and cell-average versions) is similar. The significant coefficients in the standard matrices are distributed in the well known finger-shape, where the *fingers* appear in those locations and frequencies where the analysis finds some sort of singularity.

Moreover, the standard form of matrices representing smooth operators, or operators with a finite number of singularities, is sparse. This sparsity depends on the operator and the three multiresolution analysis we have studied produce comparable results.

We do not want, however, to to get into too much detail about comparing the particular results of all three standard forms. As noticed, the compression rates depend on the order of the approximation and, due to the symmetry, the reconstructions

initial vec.	iter. algorith 1	iter. algorith2	$\|\cdot\|_1$ $-error$	$\|\cdot\|_\infty$ $-error$
f	1	0	$0.9740 \cdot 10^{-2}$	$0.1166 \cdot 10^{-1}$
	2	0	$0.1426 \cdot 10^{-2}$	$0.1716 \cdot 10^{-2}$
	3	0	$0.3308 \cdot 10^{-4}$	$0.6226 \cdot 10^{-4}$
	4	0	$0.8318 \cdot 10^{-5}$	$0.3110 \cdot 10^{-4}$
	1	3	$0.3314 \cdot 10^{-4}$	$0.6253 \cdot 10^{-4}$
	2	2	$0.8326 \cdot 10^{-5}$	$0.2983 \cdot 10^{-4}$
	3	1	$0.8329 \cdot 10^{-5}$	$0.3105 \cdot 10^{-4}$
1	1	0	$0.1167 \cdot 10^{-0}$	$0.1356 \cdot 10^{-0}$
	2	0	$0.1701 \cdot 10^{-1}$	$0.2020 \cdot 10^{-1}$
	3	0	$0.3659 \cdot 10^{-3}$	$0.4522 \cdot 10^{-3}$
	4	0	$0.8318 \cdot 10^{-5}$	$0.3110 \cdot 10^{-4}$
	1	3	$0.3659 \cdot 10^{-3}$	$0.4555 \cdot 10^{-3}$
	2	2	$0.1180 \cdot 10^{-4}$	$0.3031 \cdot 10^{-4}$
	3	1	$0.8362 \cdot 10^{-5}$	$0.3090 \cdot 10^{-4}$

TABLE 20
Integral ec. Example 3 ; N= 128 tol= 10^{-5}, u_w

we have chosen have different orders. Also, the support of the wavelets doubles that of the interpolations and, especially for the parabolic and hyperbolic problems, the number of significant coefficients depends not only on the order but on the support. In order to compare particular performances it is necessary to consider some circumstances which are beyond the scope of this report. The general behavior, though, as noticed, is comparable in these three cases.

In each one of the problems we have studied there are two different processes. One is the discretisation of the problem, the other is its multiresolution analysis. In this report, we have emphasized the second one, but it is obvious that the obtained results can be improved if we strengthen the first one. We can state that the sparsity comes from the multiresolution process per se, and not by the particular local reconstruction we use in each scale, let it be either wavelet approximation, or point value or cell-average interpolation.

Thus, Harten's multiresolution analysis appears to be a good alternative to the traditional wavelet analysis, and a very natural one from the point of view of numerical analysis. Furthermore, the freedom to choose the reconstruction is quite remarkable.

Acknowledgments The authors thank S. Osher for his comments and his support.

REFERENCES

[1] F. Arándiga and V. Candela, *Multiresolution Standard Form of a Matrix*, UCLA Computational and Applied Mathematics Report 92-37 (1992)

[2] K. E. Atkinson, *A survey of Numerical Methods for the Solution of Fredholm Integral Equations of Second Kind*, SIAM, Philadelphia (1976)

[3] G. Beylkin, *Wavelets, Multiresolution Analysis and Fast Numerical Algorithms*, preprint

[4] G. Beylkin, R. Coifman and V. Rokhlin, *Fast Wavelet Transform and Numerical Algorithms I.*, Comm. Pure Appl. Math., **XLIV**, pp. 141-183, (1991)

[5] A. Brandt and A. A. Lubrecht , *Multilevel Matrix Multiplication and Fast Solution of Integral Equations*, J. Compu. Phys., Vol 90. no. 2, October 1990.

[6] B. Engquist, S. Osher, and S. Zhong, *Fast Wavelet Algorithms for Linear Evolution Equations*, ICASE Report 92-14 (1992)

[7] A. Harten, *Discrete Multiresolution Analysis and Generalized Wavelets*, UCLA Computational and Applied Mathematics Report 92-08 (1992)
[8] A. Harten and I. Yad-Shalom, *Fast Multiresolution Algorithms for Matrix-Vector Multiplication*, UCLA Computational and Applied Mathematics Report 92-31 (1992)
[9] S. Mallat, *Multiresolution approximations and wavelet orthonormal bases of* $L^2(R)$, Trans. Amer. Math. Soc., **Vol. 315**, 69-87, (1989)