Multiresolution Standard Form of a Matrix

Francesc Arandiga
Vicente F. Candela

August 1992

CAM Report 92-37

Multiresolution Standard Form of a Matrix

Francesc Aràndiga*
Vicente F. Candela †

Departament de Matemàtica Aplicada
Universitat de València

August 25, 1992

Abstract

We present a new method to compress matrices based on multiresolution interpolation, such as it appears in [4, 5]. The matrices so compressed allow all kind of algebraic operations, including matrix-matrix multiplication. We illustrate through examples the performance of the compressed matrices. We call the form devised here *"standard"* in contrast to the *"nonstandard"* form in [5], because they are the multiresolution equivalent to the standard and nonstandard wavelet forms in [2], respectively.

1 Introduction

One of the applications of wavelet bases is compression of matrices. If a matrix represents a smooth operator, its representation in a wavelet basis is a sparse matrix. This is called the standard form of the original matrix. In [2], the authors present a new form of the matrix which they call nonstandard.

*Research supported by a Grant from Conselleria de Cultura, Educaciò i Ciència de la Generalitat Valenciana and by ONR Grant N00014-91-J-1034

†Research supported by a Grant from DGICYT PS90-0265 and by ONR Grant N00014-91-J-1034

This nonstandard form is no longer a representation of the operator in any basis, but a two dimensional tensor form of the wavelet transform. Assuming the matrix represents the operator in the finest scale, they obtain versions of the matrix in every coarser scale. In this way, the nonstandard form decouples the different scales. The price to pay is a non direct form to multiply both matrix-vectors and matrix-matrix. While matrix-vector multiplication can be done ([2]) with a small increase of computational complexity (though the computational overall cost is smaller) over the usual multiplication, this gets worse in the case of matrix-matrix multiplication (when there is some previous knowledge of the operator, however, there is a way, in [1], to evaluate this multiplication). Therefore, the standard and the nonstandard forms have different properties (simplicity for the standard, speed for the nonstandard) which can make a cause for each of them for different problems.

Despite all their advantages, and due to the constraints imposed on their construction, wavelets have two main inconveniences. One is the large support needed if some conditions are asked. For example, if we wish orthogonality, the support of the wavelet doubles that of the spline basis, in order to get the same order of accuracy. On the other side, and once again in the orthogonal case, although they are seldom explicitly computed, wavelet bases are very oscillatory. This restricts their applications. For instance, it is not advisable to use wavelet interpolation.

These problems have been reduced by means of biorthogonal wavelets. But now, while we can control one of the wavelet functions, we need another one that, in most cases, is imposed by our first choice.

In [4, 5], a generalization of wavelet theory is presented. New algorithms based on multiresolution analysis are devised. These algorithms are broad enough to include wavelets as a special case. They are also less restrictive and it is possible to keep the advantages of wavelets, while eliminating their inconveniences.

In [5], a bidimensional form of the multiresolution is presented. Practically, when applied to matrices, this analysis produces a generalized nonstandard form.

However, as we said before, the nonstandard form is not always the best tool to work with matrices. Our main concern lies on matrix-matrix multiplications. In [3], fast methods to solve some kind of parabolic and hyperbolic equations are shown. These methods require successive squaring of matrices. The nonstandard form in [5] does not fit very well from this point of view.

Moreover, in [1], there are some algorithms to evaluate elementary matrix functions based on the sparsity of the matrix. They also need matrix-matrix multiplications. Although it is possible (and, indeed, sometimes it is preferable) to work with nonstandard matrices, it is clear that the standard form provides a simpler way to do it.

It is interesting, for all these reasons, to get a generalized standard form complementary to the generalized nonstandard form in [5], as a previous step to study the wavelet-based algorithms in the more general multiresolution analysis.

This report is organized as follows: in §2, we review some of the results in [4, 5]. In §3, we construct the generalized standard form and, in §4, we show some numerical examples. Finally, in §5, we draw conclusions.

[4, 5] consider multiresolution from three different perspectives: point value interpolation, cell averages and wavelets. The only one we are going to consider from now on is point value interpolation. Generalization to cell averages and comparison between all three will be done in further stages.

2 Multiresolution analysis

We review here some of the results in [4, 5] with respect to point value interpolation.

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

According to the usual notation in wavelet theory,

$$s^k_j = \int f(x)\varphi^k_j(x)dx. \tag{1}$$

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 $\{s^0_j\}$, it is easy to get the information from the coarser grids, $\{s^k_j\}$.

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

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

must be small.

From the knowledge of the coarsest scale and the differences $Q_k(x)$, we can construct 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

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

The more accurate $R_k(x)$ is, the smaller the absolute values of $\{d_j^k\}$ are. If we truncate to zero the $\{d_j^k\}$ 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, and in this context, (1) and (3) are equivalent to:

$$s_j^k = \int f(x)\frac{1}{2^k h_0}\delta\left(\frac{2^{-k}x - j}{h_0}\right)dx = f(x_j^k) \tag{7}$$

$$R_k(x_j^k) = f(x_j^k), \quad j = 1, \dots, N_k \tag{8}$$

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:

$$f_j^0=c_j, \quad 1\le j\le N_0 \tag{9}$$

$$\left\{\begin{array}{l}\text{Do for } k=1,\ldots,L\\[2ex] f_j^k=f_{2j}^{k-1}, \quad 1\le j\le N_k\\[2ex] d_j^k=f_{2j-1}^{k-1}-I_k(x_{2j-1}^{k-1};f_i^k), \quad 1\le j\le N_k\\[2ex] \text{End}\end{array}\right. \tag{10}$$

$$c^{MR}=\{(d^1,\ldots,d^L),f^L\} \tag{11}$$

is the multiresolution representation of c.

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

Algorithm 2

$$\left\{\begin{array}{l}\text{Do for } k=L,\ldots,1\\[2ex] f_{2j}^{k-1}=f_j^k, \quad 1\le j\le N_{k-1}\\[2ex] f_{2j-1}^{k-1}=d_j^k+I_k(x_{2j-1}^{k-1};f_i^k), \quad 1\le j\le N_k\\[2ex] \text{End}\end{array}\right. \tag{12}$$

$$c=f^0 \tag{13}$$

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 $k+1$,

$$\{d_j^{k+1}\} = G\{f_j^k\} \tag{14}$$

$$\{f_j^{k+1}\} = H\{f_j^k\} \tag{15}$$

A two dimensional version of this algorithm can be found in ([5]).

Given a matrix A, its multiresolution form A^{MR} can be evaluated according to the following

Algorithm 3:

$$A^0 = A \tag{16}$$

$$\left\{ \begin{array}{l} \text{Do for } k = 1, \ldots, L \\ \\ A^k = HA^{k-1}H^* \\ \\ E^{k-1} = I_k A^k I_k^* - A^{k-1} \\ \\ \left\{ \begin{array}{l} D_1^k = GE^{k-1}G^* \\ D_2^k = GE^{k-1}H^* \\ D_3^k = HE^{k-1}G^* \end{array} \right. \\ \\ \text{End} \end{array} \right. \tag{17}$$

$$A^{MR} = \{\{D_i^1\}_{i=1}^3, \cdots, \{D_i^L\}_{i=1}^3, A^L\} \tag{18}$$

As it happened with vectors, if the interpolation approximates the matrix, the coefficients in E^k are small in absolute value, and so are those in $\{D_i^k\}_{i=1}^3$.

In A^{MR}, the scales are decoupled. Each D_i^k represents interaction between samples and (or) differences at the scale k. However, A^{MR} is not the representation of the matrix in any basis. Matrix-vector multiplication is not straightforward. Moreover, matrix-matrix multiplication is not easy to implement in the nonstandard form.

In §3, our goal is to get another representation of the matrix using the multiresolution schemes we have reviewed in this paragraph. Among other requirements, in the form we are going to get both matrix-vector and matrix-matrix multiplication are implemented in the usual way. The coefficients will be the components of the matrix in a multiresolution basis.

3 Standard form of a matrix

Our main interest is to get the *standard form* of the matrix. Although this concept has a clear equivalence in wavelet theory, we shall explain here what we mean by standard form in the general multiresolution context.

Given a vector f and a matrix A, the standard form of A is the operator that relates the multiresolution form of f, f^{MR}, and the multiresolution form of $\tilde{f} = Af$. In other words, we are looking for a way to get $\tilde{f}^{MR}$ directly from f^{MR}, without evaluating Af.

The problem can be stated in the following way:

Problem:

Find the operator A^s such that $(Af)^{MR} = A^s f^{MR}$.

In the case of wavelets, the standard form of the matrix is obtained by a simple change of bases, using Mallat's pyramidal algorithm. In the general frame we considered in §2, A^s does not even have to be a matrix. If the interpolation I_k depends on data, as it happens in ENO methods, for example, the multiresolution transformation (the operator M such that $f^{MR} = Mf$) is not a linear operator, and, therefore, A^s is not a matrix.

We are going to assume that the interpolation does not depend on data. In this case, the multiresolution is a linear process, and we hope the transformation of a matrix to its standard form is also linear. Furthermore, as we shall see, this standard form is going to be the matrix expressed in another basis, in the same way as the wavelet standard form. However, we are going to get this form without computing explicitly the new multiresolution basis.

The multiresolution is linear. So, there exists a matrix M such that

$$f^{MR} = Mf \tag{19}$$

As we saw in §2, M is invertible. Its inverse is given by Algorithm 2. This means that we can recover the original vector from its multiresolution:

$$f = M^{-1} f^{MR} \tag{20}$$

Another way to look at it is to consider M as a change of bases.

Using this notation it is easy to get the standard form of the matrix A,

$$(Af)^{MR} = M(Af) = MAM^{-1}(Mf) = (MAM^{-1}) f^{MR} \tag{21}$$

Thus,

$$A^s = MAM^{-1} \tag{22}$$

From (21), it is immediate that the multiplication of a standard matrix times a vector is implemented in the usual way. The other algebraic operation we want to check is matrix-matrix multiplication.

If A and B are two matrices (we are going to suppose they are square matrices with the same dimension, though it is not necessary), then,

$$(AB)^s = M(AB)M^{-1} = (MAM^{-1})(MBM^{-1}) = A^s B^s \tag{23}$$

Therefore, the standard form of the multiplication of two standard matrices equals the multiplication of their standard forms. This result and (21) allows us to work with standard forms in a simple and natural way.

Practically, the standard form of the matrix is obtained applying Algorithm 1 to the columns of the matrix and, then, the adjoint of Algorithm 2 to the rows of the transformed matrix.

Because of Algorithm 2, the transformation of rows needs sums instead of differences. The standard matrix, however, still compresses data. The sums we have to do are from coefficients we have previously transformed. Then, most of the coefficients we have to sum are previous differences. So, the final coefficients are also small in absolute values.

When the interpolation is centered around the node we want to interpolate and we suppose periodicity, the standard form of the matrix can be obtained by means of the following

Algorithm 4

```
For $j = 1, \cdots, N$
            $f_j(i) = a_{i,j}, \quad 1 \le i \le N$                                  (24)
            $b_{*,j} = f_j^{MR}$                                                     (25)
End
For $i = 1, \cdots, N$
            $g_i^0(j) = b_{i,j}, \quad 1 \le j \le N$                                (26)
        Do for $k = 1, \ldots, L$
            $s_i^k(j) = g_i^{k-1}(2j-1), \quad 1 \le j \le N_k$                      (27)
            $g_i^k(j) = g_i^{k-1}(2j) + I_k(x_{2j}^{k-1}, s_i^k), \quad 1 \le j \le N_k$
End
            $c_{i,*} = \{(s_j^1)_{j=1}^{N_1}, \ldots, (s_j^L)_{j=1}^{N_L}, (g_j^L)_{j=1}^{N_L}\}$   (28)
            $A^s = (c_{i,j})_{i,j=1}^{N}$                                            (29)
```

As it can be seen in the figures, the typical *finger shape* of the matrix appear when we plot A^s, the standard form of a matrix A representing a pseudodifferential function. Unlike the wavelet case, this is not a symmetric transformation, and the horizontal shapes are more remarkable than the vertical ones, mainly in the coarsest scales.

The standard form is organized as a block matrix. Each block $A^{i,j}$ is a matrix $N_i \times N_j$, which represents the interactions between the scales i and j. The difference at k level of Af (denoted by $\tilde{d}^k$) is the sum of the contributions of the differences from all the scales, and the contribution from the samples in the coarsest scale, f^L:

$$\tilde{d}^k = \sum_{j=1}^{L} A^{k,j} d^j + B^{k,L} f^L \tag{30}$$

4 Numerical Examples

We present now the results after applying the *multiresolution standard form* of three different matrices.

Example 1 Multiresolution standard form of the Lax-Wendroff scheme, $cfl = 0.4$.

Example 2

$$a_{i,j} = \begin{cases} 0, & \text{if i=j} \\ \log((i-j)^2), & \text{elsewhere} \end{cases}$$

Example 3

$$a_{i,j} = \begin{cases} 0, & \text{if i=j} \\ \frac{1}{i-j}, & \text{elsewhere} \end{cases}$$

Example 1, in its original form, is a tridiagonal matrix. The standard form is more dense than it was originally. Nevertheless, we hope that after successive squaring, the matrix remains sparse, as it happens in the wavelet case, in order to apply the algorithms in [3] to general multiresolution.

Examples 2 and 3 appear in [2]. We can compare these results to the ones shown there.

We tested the examples with linear and cubic interpolation. In the tables we display the number of significant coefficients (those with absolute values greater than a given tolerance, *tol*), and the error, after thresholding by *tol*, carried out when multiplying the matrix times the test vector $\{u_i\}_{i=1}^N$, where $u_i = \sin(2\pi \frac{i}{N})$. We show the results for different dimensions of the matrix, N, thresholdings, *tol*, and number of scales, L. Not very surprisingly, when N_L is small (less than 8, for instance), the influence of L is not very significant.

The number of significant coefficients in example 2 is slightly greater than in example 3, due to the fact that the decay of the coefficients in example 3 is higher than in example 2.

The number of significant coefficients in all cases seems to be $O(N \log N)$.

Finally, the error produced by truncation is the same order as the thresholding. So, we are always under the desired accuracy.

The figures show the standard form of each of the matrices above mentioned. We display two cases for examples 2 and 3. One of them is the standard form assuming periodicity. The second figure of each examples has been obtained by a small variation of algorithm 4, modifying the interpolation near the extremes of the columns and rows of the matrix in order to suppress discontinuities in the boundaries. The tables, however, display the results assuming periodicity.

5 Conclusions

The multiresolution we have just presented is a simple two dimensional generalization of the multiresolution in [4]. The results obtained so far show that this standard form mimics the behaviour of the wavelet standard form. The sparsity of the matrix grows with the rate of decay of the function it represents. The significant coefficients appear around the diagonals, and its shifted translates, where it exists some kind of sharp variation.

Generalization to cell averages can be done in a rather direct form. Furthermore, as the transformations of rows and columns are independent, mixed transformations (from cell averages to point values and *viceversa*) are possible, allowing discretization of functions of two variables in a different way for each variable, as it is suggested in [5]. These options are currently under investigation.

Finally, the standard form is adequate for the fast algorithms presented in [3]. It is important the fact that multiplication of standard matrices is itself a standard matrix in order to apply those methods. The algorithms shown in [1] to get elementary functions of matrices can also be tried from this multiresolution point of view.

Acknowledgements

The authors thank A. Harten for suggesting the problem and for providing numerous insights throughout the research and writing of this paper; and S. Osher and R. Donat for their useful comments.

References

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

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

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

[4] A. Harten, *Discrete Multiresolution Analysis and Generalized Wavelets*, UCLA Computational and Applied Mathematics Report 92-08 (1992)

[5] A. Harten and I. Yad-Shalom *Fast Multiresolution Algorithms for Matrix-Vector Multiplication*, UCLA Computational and Applied Mathematics Report 92-31 (1992)

$tol = 10^n$	$scales$	$nonzeros$	$\|\cdot\|_1$ $-error$	$\|\cdot\|_\infty$ $-error$
$n=-3$	6	1166	$2.7225 \cdot 10^{-4}$	$8.9900 \cdot 10^{-4}$
	4	1164	$5.3897 \cdot 10^{-4}$	$1.4996 \cdot 10^{-3}$
$n=-4$	6	1361	$2.4046 \cdot 10^{-5}$	$1.2774 \cdot 10^{-4}$
	4	1396	$2.4046 \cdot 10^{-5}$	$1.2775 \cdot 10^{-4}$
$n=-7$	6	1437	$1.4279 \cdot 10^{-16}$	$4.4409 \cdot 10^{-16}$
	4	1464	$1.3986 \cdot 10^{-16}$	$4.4409 \cdot 10^{-16}$

Table 1: Absolute errors for *example 1*; points 64×64=4096; cubic interpolation; Lax-Wendroff scheme ($\lambda = 0.4$).

$tol = 10^n$	$scales$	$nonzeros$	$\|\cdot\|_1$ $-error$	$\|\cdot\|_\infty$ $-error$
$n=-3$	8	5076	$3.3595 \cdot 10^{-4}$	$1.1915 \cdot 10^{-3}$
	6	5036	$2.9666 \cdot 10^{-4}$	$1.0581 \cdot 10^{-3}$
$n=-4$	8	6126	$3.0173 \cdot 10^{-5}$	$1.0466 \cdot 10^{-4}$
	6	6136	$5.1946 \cdot 10^{-5}$	$1.4536 \cdot 10^{-4}$
$n=-7$	8	7675	$1.4653 \cdot 10^{-8}$	$8.2769 \cdot 10^{-8}$
	6	7884	$1.4653 \cdot 10^{-8}$	$8.2769 \cdot 10^{-8}$

Table 2: Absolute errors for *example 1*; points 256×256=65536; cubic interpolation; Lax-Wendroff scheme ($\lambda = 0.4$).

$tol = 10^n$	$scales$	$nonzeros$	$\|\cdot\|_1$ $-error$	$\|\cdot\|_\infty$ $-error$
$n=-3$	6	2427	$1.6718 \cdot 10^{-2}$	$2.6123 \cdot 10^{-2}$
	4	2402	$1.6718 \cdot 10^{-2}$	$2.6123 \cdot 10^{-2}$
$n=-4$	6	2977	$8.3642 \cdot 10^{-4}$	$1.4747 \cdot 10^{-3}$
	4	2968	$8.3642 \cdot 10^{-4}$	$1.4746 \cdot 10^{-3}$
$n=-7$	6	4093	$1.5029 \cdot 10^{-13}$	$3.3538 \cdot 10^{-12}$
	4	4087	$1.2942 \cdot 10^{-13}$	$3.2969 \cdot 10^{-12}$

Table 3: Absolute errors for *example 2*; points 64×64=4096; cubic interpolation ; $A(i,j) = log((i-j)^2)$;

$tol = 10^n$	$scales$	$nonzeros$	$\|\cdot\|_1 -error$	$\|\cdot\|_\infty -error$
$n=-3$	8	16099	$7.3426 \cdot 10^{-2}$	$1.1190 \cdot 10^{-1}$
	6	15929	$7.3425 \cdot 10^{-2}$	$1.1190 \cdot 10^{-1}$
$n=-4$	8	21441	$4.3509 \cdot 10^{-3}$	$7.3054 \cdot 10^{-3}$
	6	21310	$4.3509 \cdot 10^{-3}$	$7.3054 \cdot 10^{-3}$
$n=-7$	8	49913	$3.6926 \cdot 10^{-6}$	$7.6262 \cdot 10^{-6}$
	6	49906	$3.6926 \cdot 10^{-6}$	$7.6262 \cdot 10^{-6}$

Table 4: Absolute errors for *example 2*; points 256×.256=65536; cubic interpolation ; $A(i,j) = log((i-j)^2)$;

$tol = 10^n$	$scales$	$nonzeros$	$\|\cdot\|_1 -error$	$\|\cdot\|_\infty -error$
$n=-3$	6	2055	$3.0791 \cdot 10^{-3}$	$9.8342 \cdot 10^{-3}$
	4	2088	$3.0791 \cdot 10^{-3}$	$9.8342 \cdot 10^{-3}$
$n=-4$	6	2448	$3.6412 \cdot 10^{-6}$	$9.6723 \cdot 10^{-4}$
	4	2476	$3.6412 \cdot 10^{-6}$	$9.6723 \cdot 10^{-4}$
$n=-7$	6	3879	$1.0820 \cdot 10^{-7}$	$6.5119 \cdot 10^{-7}$
	4	3880	$1.0820 \cdot 10^{-7}$	$6.5119 \cdot 10^{-7}$

Table 5: Absolute errors for *example 3*; points 64×64=4096; cubic interpolation ; $A(i,j) = 1/(i-j)$;

$tol = 10^n$	$scales$	$nonzeros$	$\|\cdot\|_1 -error$	$\|\cdot\|_\infty -error$
$n=-3$	8	11961	$1.1988 \cdot 10^{-2}$	$3.3126 \cdot 10^{-2}$
	6	12060	$1.1950 \cdot 10^{-2}$	$3.5127 \cdot 10^{-2}$
$n=-4$	8	15618	$1.3721 \cdot 10^{-3}$	$3.9480 \cdot 10^{-3}$
	6	15810	$1.3721 \cdot 10^{-3}$	$3.9480 \cdot 10^{-3}$
$n=-7$	8	33405	$1.5002 \cdot 10^{-6}$	$3.3952 \cdot 10^{-6}$
	6	33525	$1.5002 \cdot 10^{-6}$	$3.3952 \cdot 10^{-6}$

Table 6: Absolute errors for *example 3*; points 256×.256=65536; cubic interpolation ; $A(i,j) = 1/(i-j)$;

$tol = 10^n$	$scales$	$nonzeros$	$\|\cdot\|_1$ $-error$	$\|\cdot\|_\infty$ $-error$
$n = -3$	6	737	$3.6578 \cdot 10^{-5}$	$3.9646 \cdot 10^{-4}$
	4	728	$3.6578 \cdot 10^{-5}$	$3.9646 \cdot 10^{-4}$
$n = -4$	6	737	$1.1666 \cdot 10^{-16}$	$3.3307 \cdot 10^{-16}$
	4	728	$1.1059 \cdot 10^{-16}$	$2.3592 \cdot 10^{-16}$
$n = -7$	6	737	$1.1666 \cdot 10^{-16}$	$3.3307 \cdot 10^{-16}$
	4	728	$1.1059 \cdot 10^{-16}$	$2.3592 \cdot 10^{-16}$

Table 7: Absolute errors for *example 1*; points 64×64=4096; linear interpolation; Lax-Wendroff scheme ($\lambda = 0.4$).

$tol = 10^n$	$scales$	$nonzeros$	$\|\cdot\|_1$ $-error$	$\|\cdot\|_\infty$ $-error$
$n = -3$	8	3171	$2.7344 \cdot 10^{-4}$	$9.2703 \cdot 10^{-4}$
	6	3148	$2.7344 \cdot 10^{-4}$	$9.2703 \cdot 10^{-4}$
$n = -4$	8	3205	$7.6981 \cdot 10^{-6}$	$7.9359 \cdot 10^{-5}$
	6	3192	$7.6981 \cdot 10^{-6}$	$7.9359 \cdot 10^{-5}$
$n = -7$	8	3205	$1.4347 \cdot 10^{-16}$	$5.5511 \cdot 10^{-16}$
	6	3192	$1.4347 \cdot 10^{-16}$	$5.5511 \cdot 10^{-16}$

Table 8: Absolute errors for *example 1*; points 256×256=65536; linear interpolation; Lax-Wendroff scheme ($\lambda = 0.4$).

$tol = 10^n$	$scales$	$nonzeros$	$\|\cdot\|_1$ $-error$	$\|\cdot\|_\infty$ $-error$
$n = -3$	6	3599	$4.7775 \cdot 10^{-3}$	$1.4181 \cdot 10^{-2}$
	4	3594	$4.7775 \cdot 10^{-3}$	$1.4181 \cdot 10^{-2}$
$n = -4$	6	4062	$4.0244 \cdot 10^{-14}$	$9.9476 \cdot 10^{-14}$
	4	4057	$2.3133 \cdot 10^{-14}$	$9.2371 \cdot 10^{-14}$
$n = -7$	6	4062	$4.0244 \cdot 10^{-14}$	$9.9476 \cdot 10^{-14}$
	4	4057	$2.3133 \cdot 10^{-14}$	$9.2371 \cdot 10^{-14}$

Table 9: Absolute errors for *example 2*; points 64×64=4096; linear interpolation ; $A(i,j) = log((i-j)^2)$;

$tol = 10^n$	$scales$	$nonzeros$	$\|\cdot\|_1$ $-error$	$\|\cdot\|_\infty$ $-error$
$n = -3$	8	30103	$1.1630 \cdot 10^{-1}$	$1.8500 \cdot 10^{-1}$
	6	30003	$1.1630 \cdot 10^{-1}$	$1.8500 \cdot 10^{-1}$
$n = -4$	8	53127	$3.4429 \cdot 10^{-3}$	$7.5692 \cdot 10^{-3}$
	6	53122	$3.4429 \cdot 10^{-3}$	$7.5692 \cdot 10^{-3}$
$n = -7$	8	65406	$2.7282 \cdot 10^{-13}$	$1.0516 \cdot 10^{-13}$
	6	65401	$2.5198 \cdot 10^{-13}$	$9.6634 \cdot 10^{-13}$

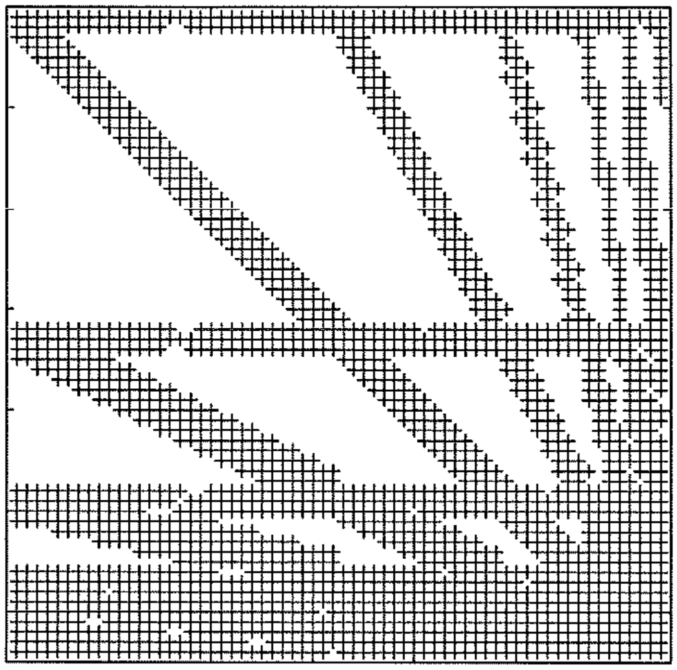
Table 10: Absolute errors for *example 2*; points 256×.256=65536; linear interpolation ; $A(i,j) = log((i-j)^2)$;

$tol = 10^n$	$scales$	$nonzeros$	$\|\cdot\|_1$ $-error$	$\|\cdot\|_\infty$ $-error$
$n = -3$	6	2055	$6.5417 \cdot 10^{-3}$	$1.4959 \cdot 10^{-2}$
	4	2063	$6.5417 \cdot 10^{-3}$	$1.4959 \cdot 10^{-2}$
$n = -4$	6	3034	$4.7950 \cdot 10^{-4}$	$1.0387 \cdot 10^{-3}$
	4	3043	$4.7950 \cdot 10^{-4}$	$1.0387 \cdot 10^{-3}$
$n = -7$	6	4039	$2.4876 \cdot 10^{-10}$	$1.3171 \cdot 10^{-9}$
	4	4037	$9.6796 \cdot 10^{-11}$	$7.7437 \cdot 10^{-10}$

Table 11: Absolute errors for *example 3*; points 64×64=4096; linear interpolation ; $A(i,j) = 1/(i-j)$;

$tol = 10^n$	$scales$	$nonzeros$	$\|\cdot\|_1$ $-error$	$\|\cdot\|_\infty$ $-error$
$n = -3$	8	11777	$2.7858 \cdot 10^{-2}$	$7.4191 \cdot 10^{-2}$
	6	11812	$2.6671 \cdot 10^{-2}$	$7.2322 \cdot 10^{-2}$
$n = -4$	8	19849	$3.0270 \cdot 10^{-3}$	$7.1844 \cdot 10^{-3}$
	6	19876	$3.0270 \cdot 10^{-3}$	$7.1844 \cdot 10^{-3}$
$n = -7$	8	64553	$8.2595 \cdot 10^{-8}$	$1.4858 \cdot 10^{-6}$
	6	64551	$8.2595 \cdot 10^{-8}$	$1.4858 \cdot 10^{-6}$

Table 12: Absolute errors for *example 3*; points 256×.256=65536; linear interpolation ; $A(i,j) = 1/(i-j)$;



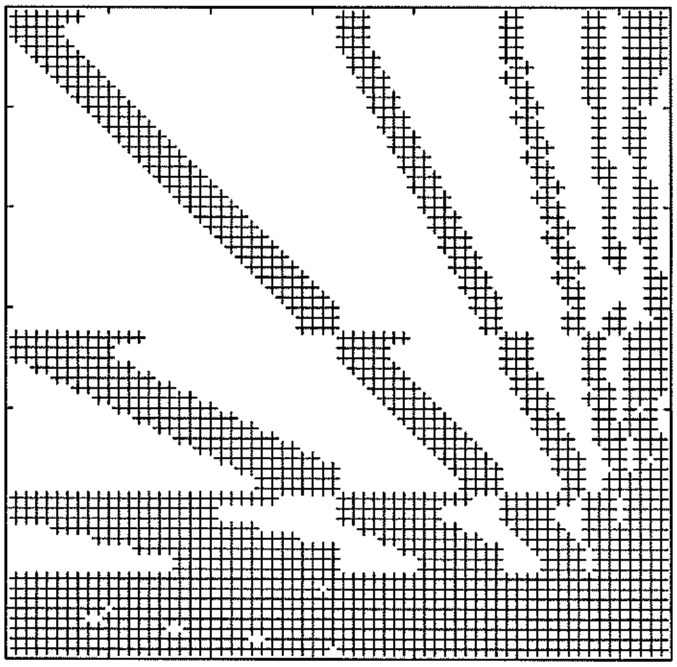
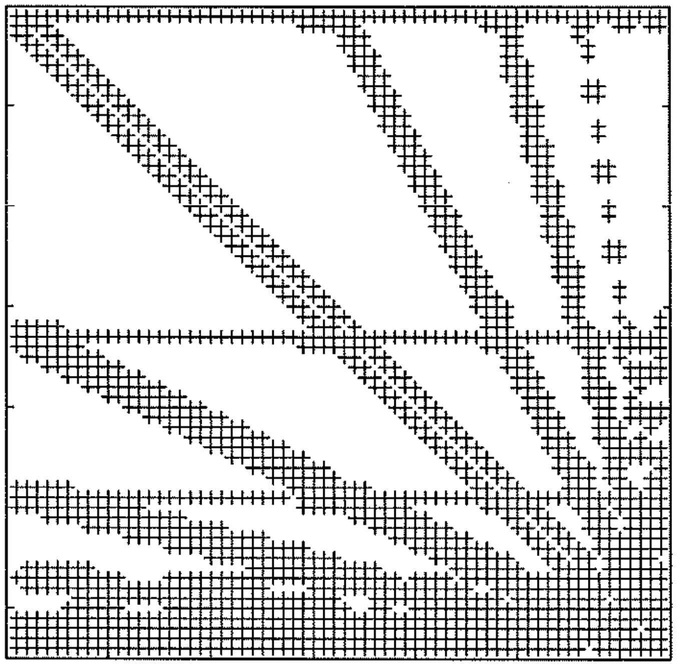


Figure 1: Example 2



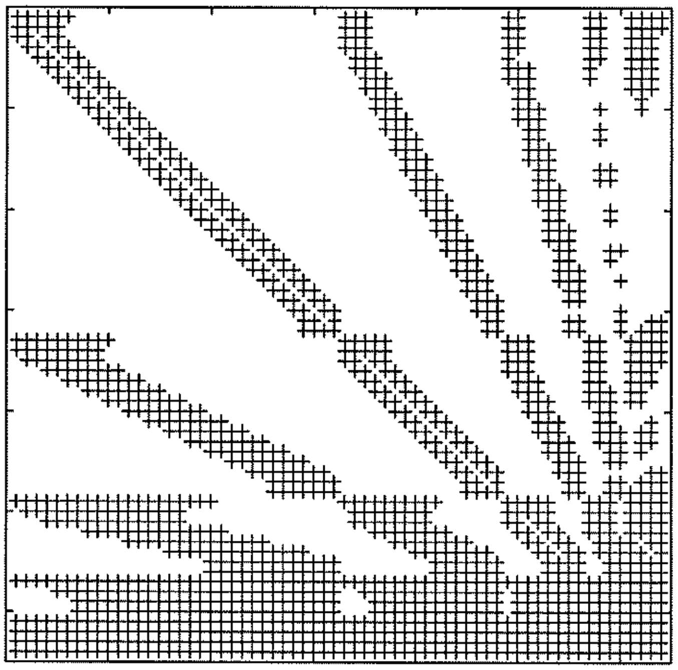


Figure 2: Example 3

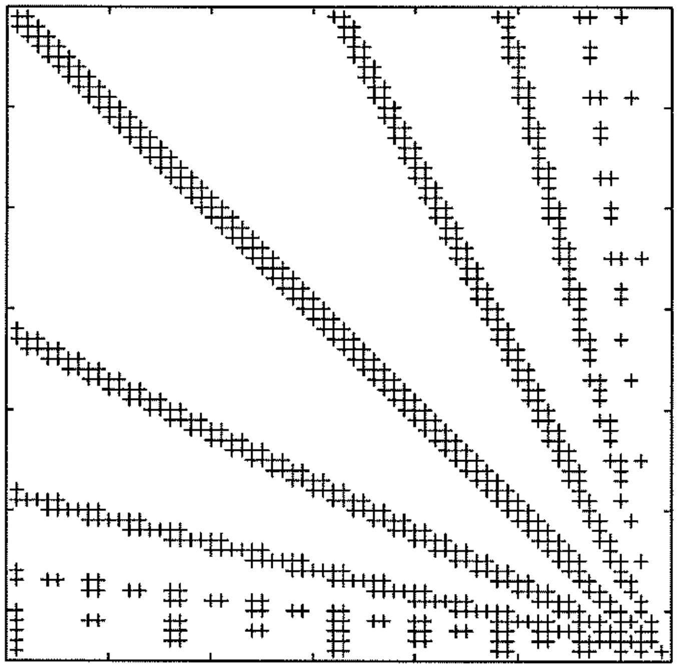


Figure 3: Example 1