Adaptive wavelet representation and differenciation on block-structured grids

M. O. Domingues [a,*,1]

[a] *Instituto Nacional de Pesquisas Espaciais, LMO/CPTEC, Caixa Postal 515, 12221-970, São José dos Campos, SP, Brazil*

S. M. Gomes [b,1]

[b] *Universidade Estadual de Campinas, IMECC, Caixa Postal: 6065, 13083-970 Campinas, SP, Brazil*

L. M. A. Díaz [c,1]

[c] *Instituto de Cibernética Matemática y Física - Calle E 309, esq. 15, Habana, Cuba*

Abstract

This paper considers an adaptive finite difference scheme for the numerical solution of evolution partial differencial equations. The computational domain is formed by non-overlapping blocks. Each block is a uniform grid, but step size may change from one block to another. The blocks are not predetermined, but they are dinamically constructed according to the refinement needs of the numerical solution. The decision over whether a block should be refined or unrefined is taken by looking at the magnitude of wavelet coefficients of the numerical solution on such block. The main objective of this paper is to establish a general framework for the construction and operation on such adaptive block-grids in 2D. The algorithms and data structure are formulated by using abstract concepts borrowed from quaternary trees. This procedure helps the understanding of the method and its computational implementation. The ability of the method is demonstrated by solving some typical test problems.

Key words:
Wavelet Analysis, Finite Differences, Adaptive Grids
PACS: 02.70.Bf, 02.30.Jr

* Corresponding author
Email addresses: margaret@cptec.inpe.br (M. O. Domingues), soniag@ime.unicamp.br (S. M. Gomes), lilliam@cidet.icmf.inf.cu (L. M. A. Díaz).
[1] Supported by Fundação de Amparo à Pesquisa do Estado de São Paulo, grants $94/2016-9$, $2000/04148-2$ and $1997/2248-5$
[2] Supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico, grant $302714/88-0$

1 Introduction

Solutions to many interesting flow problems may exhibit localized singular features, such as sharp transition layers, propagating steep fronts or pronounced spikes. Reliable approximations of these problems present a challenging computational task. Uniform gridding is not a practical option since high resolution is only needed in small regions, where irregularities occur. Therefore, significant improvements in accuracy and computational efficiency may be obtained by economically adapting the grid points to the numerical solution.

In computational fluid dynamics, there are several approaches for constructing such locally adapted meshes. Some of them resort on *ad hoc* criteria, others are based on more elaborated *a posteriori* error bounds and there are those ones using Richardson extrapolation techniques. Nowadays, another kind of adaptive criteria, which looks at the magnitude of wavelet coefficients to obtain sparse representations, is becoming useful in the construction of adaptive solvers for partial differential equations (see [1] and references therein enclosed). In the present paper we shall be concerned with such class of methods.

For instance, consider the SPR method (for sparse point representation), introduced in [2]. It is an adaptive finite difference strategy that combines the simplicity and accuracy of traditional finite difference schemes with the ability of wavelet coefficients in the characterization of local regularity of functions. The idea is to represent the functions by the point values corresponding to their significant wavelet coefficients. Typically, few points are found in each time step, the grid being coarse in smooth regions, and refined close to irregularities. At each point, spatial derivatives are discretized by uniform finite differences, using step size proportional to the point local scale. Eventually, stencils not present in the grid are approximated from coarser scales by using an interpolating subdivision scheme. Other wavelet adaptive methods have been proposed with many similarities to the SPR method. For instance, the filter bank method in [3] and the second generation wavelet collocation method in [4] may be considered as generalizations of the SPR method.

A rigorous study of the effectiveness of nonlinear wavelet representations has already being established [1]. However, sparse grids coming from wavelet data compression may present a complicated topology. This fact causes an overhead involved in operations like accessing or interpolating neighboring stencils to compute finite differences at scattered grid points. In order to reduce the overhead, one possible way is by imposing some sort of regularity in the adaptive grid at the cost of losing some sparcity. For instance, the suggestion in [2] is to use an adaptive block representation (ABR). The computational domain is formed by non-overlapping blocks. Each block is a uniform grid, but the step

size may change from one block to another. In the automatic construction of such block-grids, the wavelet coefficients are also used as the main tool to decide whether a block needs to be refined or may be coarsened.

As it is well discussed in [3], such class of adaptive wavelet solvers can be separated into two basic parts: the representation part and the operator part. The operator part is performed by finite differences on uniform grids which may be chosen by considering stability and consistency criteria. The representation part is formulated in the context of wavelet data compression by means of a simple thresholding operation. This kind of separation of the solver into independent parts has several advantages. It makes it general: it is simple to change the differential equation, the order of the finite difference method, the boundary conditions, the wavelet transform etc. Consequently, it fits well into the object oriented programming philosophy. Furthermore, this methodology may be beneficiary of the considerable achievements in the well stablished field of finite differences as well as of the more recent advances in wavelet analysis.

The main objective of this paper is to establish a general framework for the application of the ABR method in 2D. The algorithms and data structure are formulated by using abstract concepts borrowed from quaternary trees. With this procedure, we expect to improve the understanding of the method and help the process of its computational implementation. The ability of the method is demonstrated by solving some test problems showing typical features of spikes, propagating fronts and the formation of sharp transition layers.

2 Adaptive Block-Structured Grids

In this section, we shall describe the type of grids used in the ABR method and their quaternary tree structure.

Let $\mathcal{X}^0$ be the uniform grid on the rectangle $[0, L] \times [0, D]$ with spatial steps $h_x = \frac{L}{N_x}$ in the x-direction and $h_y = \frac{D}{N_y}$ in the y-direction. That is,

$$\mathcal{X}^0 = \{\gamma = (kh_x, \ell h_y), 0 \leq k < N_x, 0 \leq \ell < N_y\}\,.$$

Starting from 0, a ladder of uniform grids $\mathcal{X}^j$ are constructed by successive dyadic refinements

$$\mathcal{X}^j = \left\{\gamma = (kh_x^j, \ell h_y^j), 0 \leq k < N_x^j, 0 \leq \ell < N_y^j\right\},$$

where $N_x^j = 2^j N_x$, $N_y^j = 2^j N_y$, $h_x^j = 2^{-j} h_x$ and $h_y^j = 2^{-j} h_y$. Thus, each $\mathcal{X}^j$ is obtained from $\mathcal{X}^{j-1}$ by midpoint insertions. This grid sequence can be organized in a quaternary tree structure (quad-tree).

2.1 Quad-tree structure

Let $\mathcal{B}_\mu^j$ be a generic block of $N_x \times N_y$ points in $\mathcal{X}^j$ of type

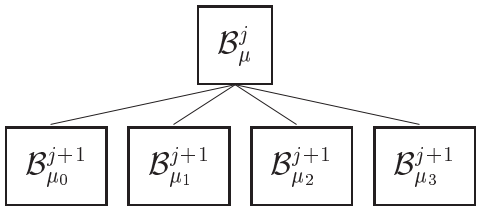
$$\mathcal{B}_\mu^j = \left\{ \gamma = \mu + (kh_x^j, \ell h_y^j), 0 \leq k < N_x, 0 \leq \ell < N_y \right\},$$

which is uniquely represented by its origin position μ and its scale level j. From $\mathcal{B}_\mu^j$, new blocks at level $j+1$ are obtained, firstly by dyadic refinement, and then by splitting into four parts

$$\mathcal{B}_\mu^j \rightarrow \mathcal{S}(\mathcal{B}_\mu^j) = \{\mathcal{B}_{\mu_0}^{j+1}, \mathcal{B}_{\mu_1}^{j+1}, \mathcal{B}_{\mu_2}^{j+1}, \mathcal{B}_{\mu_3}^{j+1}\}.$$

If μ is the origin of the starting block, then the origins of the new blocks are $\mu_0 = \mu$, $\mu_1 = \mu + (0, \frac{N_y}{2}\, h_y^j)$, $\mu_2 = \mu + (\frac{N_x}{2}\, h_x^j, \frac{N_y}{2}\, h_y^j)$ and $\mu_3 = \mu + (\frac{N_x}{2}\, h_x^j, 0)$. Using the quad-tree terminology, the elements of the set $\mathcal{S}(\mathcal{B}_\mu^j)$ are called the children of the node $\mathcal{B}_\mu^j$, $\mathcal{B}_\mu^j$ is the mother of $\mathcal{B}_{\mu_\kappa}^{j+1}$, $\mathcal{B}_{\mu_{\kappa'}}^{j+1}$ is son of $\mathcal{B}_\mu^j$, and $\mathcal{B}_{\mu_\kappa}^{j+1}$ is brother of $\mathcal{B}_{\mu_{\kappa'}}^{j+1}$. One generation of a quad-tree block structure is illustrated in Figure 1.

a) Tree structure



b) Grid structure

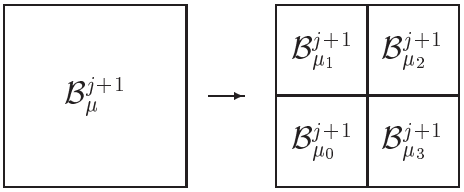


Fig. 1. The representation of a block generation

By defining $\mathcal{B}_{\mu_0}^0 = \mathcal{X}^0$ as the root of the tree, and performing $\mathcal{J}$ generations, a complete quad–tree having $\mathcal{J}+1$ levels is obtained. At each level $0 \leq j \leq \mathcal{J}$ there exist 2^{2j} blocks (nodes). Let us denoted by $\mathcal{I}^j$ the set of block origin points at level j.

Incomplete quad–trees occur when some nodes have no children at intermediary levels. This leads to the concept of tree leaves: *leaves are nodes that do not have children.*

In a complete block quad–tree, the leaves correspond to the blocks $\mathcal{B}_\mu^{\mathcal{J}}$ at the last level. The union of these blocks constitutes the uniform grid at the finest scale

$$\mathcal{X}^{\mathcal{J}} = \bigcup_{\mu \in \mathcal{I}^{\mathcal{J}}} \mathcal{B}_\mu^{\mathcal{J}}.$$

In an incomplete block quad–tree, the leaves at intermediary levels $0 < j < \mathcal{J}$ correspond to blocks where the refinement process has been interrupted. Let $\Lambda^j \subset \mathcal{I}^j$ be the set of origin points associated to the leaf-blocks at level j. The union of such blocks forms a block-structured grid

$$\mathcal{M} = \bigcup_{j=1}^{\mathcal{J}} \bigcup_{\mu \in \Lambda^j} \mathcal{B}_\mu^j.$$

The diagram in Figure 2 illustrates a 4-level incomplete tree and its corresponding grid.

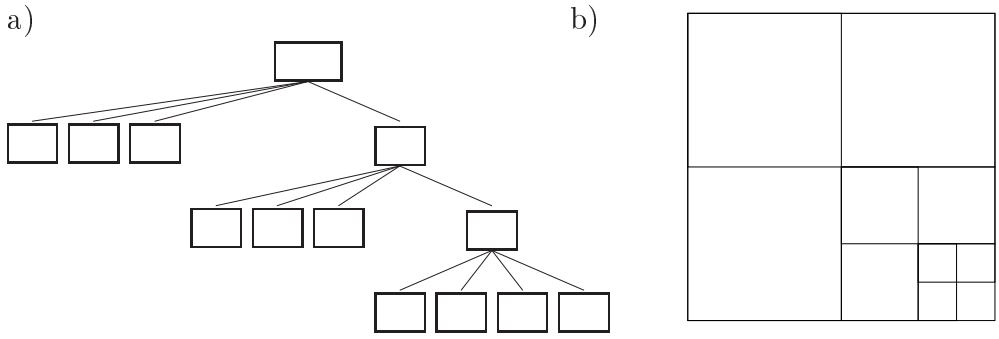


Fig. 2. Incomplete 4-level quad-tree and its corresponding block-structured grid.

3 Adaptive Construction of Block–Grids

For a given block-structured grid $\mathcal{M}$, we shall denote by $f^{\mathcal{M}}$ the collection of point values of a given function $f(x, y)$ represented at $\mathcal{M}$. According to the tree structure of $\mathcal{M}$, $f^{\mathcal{M}}$ can be organized as a vector whose components are $N_x \times N_y$ matrices containing the data corresponding to the point values at the leaf-blocks $\mathcal{B}_\mu^j$.

The purpose of the ABR technique is to obtain representations $\{\mathcal{M}, f^{\mathcal{M}}\}$ as sparse as possible. This means that a small total number of blocks is found within an incomplete quad-tree $\mathcal{M}$, the leaves in lower levels (big scales) corresponding to smooth regions and those at higher levels (small scales) corresponding to irregularity regions. In the construction of such an adaptive representation, the main tool is a local regularity indicator $i(\mathcal{B})$ to decide, at

each generation, whether a given block should be a leaf, $i(\mathcal{B}) = 0$, or not, $i(\mathcal{B}) = 1$. Next, we shall describe how wavelet coefficients in an interpolatory multi-resolution analysis can be used in the definition of $i(\mathcal{B})$. In the wavelet literature, there are several examples of multi-resolution analyses that may be used as well. In fact, some of them offer convenient aspects, like shorter filters for the same order of polynomial cancellation (zero moments for the dual wavelets), as described in [3], or additional zero moments for the primal wavelets, like in the modified lifting scheme adopted by [4].

3.1 Wavelet Indicators

Given a generic block $\mathcal{B}_\mu^j$ at level j, define the set $\tilde{\mathcal{B}}_\mu^j$ as the completion of $\mathcal{B}_\mu^j$ by the inclusion of extra points in the right and upper lateral lines. That is,

$$\tilde{\mathcal{B}}_\mu^j = \left\{ \gamma = \mu + (kh_x^j, \ell h_y^j), 0 \leq k \leq N_x, 0 \leq \ell \leq N_y \right\}.$$

Consider the rectangular grid $\mathcal{R}^j = \tilde{\mathcal{B}}_\mu^j$ and its dyadic refinement $\mathcal{R}^{j+1}$ given by

$$\mathcal{R}^{j+1} = \left\{ \gamma = \mu + (kh_x^{j+1}, \ell h_y^{j+1}), 0 \leq k \leq 2N_x, 0 \leq \ell \leq 2N_y \right\}.$$

Note that $\mathcal{R}^{j+1}$ can also be expressed as

$$\mathcal{R}^{j+1} = \bigcup_{\kappa=0}^{3} \tilde{\mathcal{B}}_{\mu_\kappa}^{j+1},$$

where $\tilde{\mathcal{B}}_{\mu_\kappa}^{j+1}$ are the completion of the children of $\mathcal{B}_\mu^j$.

Let f^j be the matrix containing the values of a given function $f(x, y)$ at the grid points in $\mathcal{R}^j$

$$f_{k,\ell}^j = f(\mu + (kh_x^j, \ell h_y^j)).$$

In wavelet analysis, transformation algorithms relating f^{j+1}, f^j and the wavelet coefficients d^j, containing the difference of information between two consecutive levels, play a crucial role. For discretizations by means of points values, the wavelet coefficients are usually defined in terms of interpolation error [1]. Values $\tilde{f}_{2k,2\ell+1}^{j+1}$, $\tilde{f}_{2k+1,2\ell}^{j+1}$, and $\tilde{f}_{2k,2\ell+1}^{j+1}$, at the new midpoints $\gamma_{k,\ell}^{(\alpha),j} \in \mathcal{R}^{j+1} \setminus \mathcal{R}^j$,

$$\begin{aligned}
\gamma_{k,\ell}^{(1),j} &= \mu + (2kh_x^{j+1}, (2\ell+1)h_y^{j+1}), \\
\gamma_{k,\ell}^{(2),j} &= \mu + ((2k+1)h_x^{jj+1}, 2\ell h_y^{j+1}), \\
\gamma_{k,\ell}^{(3),j} &= \mu + ((2k+1)h_x^{j+1}, (2\ell+1)h_y^{j+1}),
\end{aligned}$$

are computed by polynomial Lagrange interpolation from the values f^j at the coarser grid. Wavelet coefficients are then defined as the differences between the known function values $f^{j+1}_{2k,2\ell+1}$, $f^{j+1}_{2k+1,2\ell}$, and $f^{j+1}_{2k,2\ell+1}$ and the values predicted by the interpolation procedure. Precisely,

$$
\begin{aligned}
f^j_{k,\ell} &= f^{j+1}_{2k,2\ell} \\
d^{(1)\,j}_{k,\ell} &= f^{j+1}_{2k,2\ell+1} - \tilde{f}^{j+1}_{2k,2\ell+1}, \\
d^{(2)\,j}_{k,\ell} &= f^{j+1}_{2k+1,2\ell} - \tilde{f}^{j+1}_{2k+1,2\ell}, \\
d^{(3)\,j}_{k,\ell} &= f^{j+1}_{2k,2\ell+1} - \tilde{f}^{j+1}_{2k,2\ell+1}.
\end{aligned}
$$

In case of rectangular 2D grids, interpolation can be expressed in terms of 1D algorithms. For instance, $\tilde{f}^{j+1}_{2k,2\ell+1}$ is obtained from $f^j_{k,q}$ by one-dimensional interpolation along the l-direction, $\tilde{f}^{j+1}_{2k+1,2\ell}$ is obtained from $f^j_{s,l}$ by one-dimensional interpolation along the k-direction. Finally, $\tilde{f}^{j+1}_{2k+1,2\ell+1}$ is obtained from $\tilde{f}^{j+1}_{2k+1,2q}$ by applying the one-dimensional interpolatory scheme in the l-direction (or, equivalently, by one-dimensional interpolation of the $\tilde{f}^{j+1}_{2s,2\ell+1}$ values in the k-direction). The general 1D interpolation formula reads

$$
\tilde{f}^{j+1}_{2\ell+1} = \sum_q p_{\ell,q} f^j_{\ell+q},
$$

in which $p_{\ell,q}$ are the Lagrange interpolation weights. In the interior of the grid, central interpolation is used. For this case, the weights do not depend on the location, i.e., $p_{\ell,q} = p_q$. One-sided interpolation is required close to the boundaries, leading to weights that do depend on the location k [2].

As interpolation errors, wavelet coefficients are good indicators of local smoothness. This fact leads to the definition of a set index $i_\epsilon(\mathcal{B})$ which is based on the significance of the wavelet coefficients associated to the block $\mathcal{B}$, as compared to a certain given threshold ϵ. By definition, $i_\epsilon(\mathcal{B}^0_0) = 1$. Then, at level $j > 0$, for each block $\mathcal{B}^j_\mu$ having set index equal to one, the children set indexes $i_\epsilon(\mathcal{B}^{j+1}_{\mu_{p,q}})$ are computed according to the following strategy. First we perform the one-level wavelet transform, as described before, and consider $\mathcal{D}(\mathcal{B}^{j+1}_{\mu_\kappa})$ as the set of those wavelet coefficients $d^{(\alpha),j}_{k,\ell}$ which are associated to points $\gamma^{(\alpha),j}_{k,\ell}$ in $\tilde{\mathcal{B}}^{j+1}_{\mu_\kappa}$. If all wavelet coefficients in $\mathcal{D}(\mathcal{B}^{j+1}_{\mu_\kappa})$ are not significant, this means that the function is smooth in this region. Consequently, the block does not need to be refined, and it will be leaf-tree. On the other hand, if at least one significant wavelet coefficient exists in the block, this means that the function is not represented there with the prescribed accuracy, and that refinement is

needed. Therefore, according to these principles, the set index i_ϵ is defined by

$$i_\epsilon(\mathcal{B}_{\mu_\kappa}^{j+1}) = \begin{cases} 0 \text{ if } |d_{k,\ell}^{(\alpha),j}| < \epsilon \ \forall d_{k,\ell}^{(\alpha),j} \in \mathcal{D}(\mathcal{B}_{\mu_{p,q}}^{j+1}) \\ 1 \text{ otherwise.} \end{cases}$$

Following this strategy, the adaptive grid construction ends when, at a certain level, all the analyzed blocks are leaves.

3.2 Numerical Examples

The first example is for the spike function

$$f(x,y) = 3\ \exp^{-2500.0\,((x-0.3)^2+(y-0.3)^2)} + sen(2\pi x) + sen(2\pi y),$$

with a singularity point at $(x,y) = (0.3, 0.3)$, as illustrated in Figure 3. For

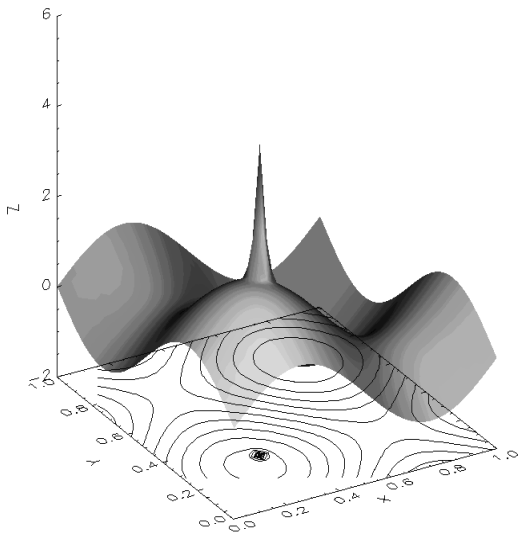


Fig. 3. The spike function.

this function, the ABR grids $\mathcal{M}_\epsilon$ are represented in Figure 4. The wavelet coefficients are for cubic polynomial interpolation ($M = 4$), the blocks have $N_x = N_y = 32$ and $\epsilon = 10^{-3}, 10^{-4}$ and 10^{-5}. As ϵ becomes smaller, more refined blocks are tended to concentrate near the singularity region. As indicated in Table 1, the number of blocks grows as ϵ becomes smaller. However, if the number of points in the ABR grid $\mathcal{M}_\epsilon$ is compared to number of points in the uniformed mesh $\mathcal{X}^\ell$, in which ℓ is the finest scale level present in $\mathcal{M}_\ell$, then the ratio $\sharp(\mathcal{M}_\epsilon)/\sharp(\mathcal{X}^\ell)$ reduced very quickly as ϵ decreases.

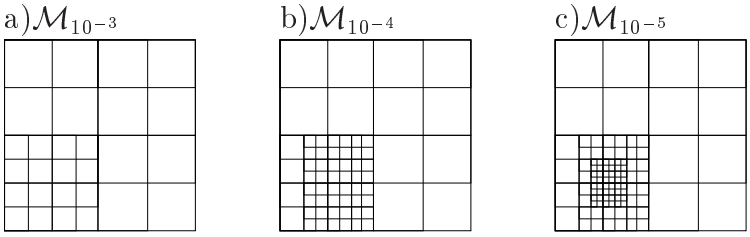


Fig. 4. ABR for the spike function

Table 1
Efficiency in ABR for the spike.

ϵ	$\sharp blocks$	$\sharp(\mathcal{M}_\epsilon)$	$\sharp(\mathcal{X}^\ell)$	$\sharp(\mathcal{M}_\epsilon)/\sharp(\mathcal{X}^\ell)$
10^{-5}	100	102400	2097152	0.05
10^{-4}	64	65536	262144	0.25
10^{-3}	28	28672	65536	0.43

As a second example, we shall consider the oblique-front function

$$f(x,y) = 1 - tanh(25x + 5(y-1)), \tag{1}$$

that exhibit abrupt changes close to the line $25x+5(y-1)=0$, as displayed in Figure 5. Therefore, the most refined blocks in the ABR grids $\mathcal{M}_\epsilon$ are expected to be placed around the oblique-front region, as ϵ decreases. This behavior is illustrated in Figure 6 according to the same parameters of the previous example. An indication of the efficiency of the ABR method for the oblique-front function can be obtained from the data in Table 2. Their comparison with the corresponding results in Table 1 implies that, given an accuracy ϵ, the spike ABR meshes $\mathcal{M}_\epsilon$ need higher resolution levels and more blocks than in the oblique-front case. In spite of that, the efficiency ratio is better for the spike function.

Table 2
Efficiency in ABR for the oblique-front

ϵ	$\sharp$ blocks	$\sharp(\mathcal{M}_\epsilon)$	$\sharp(\mathcal{X}^j)$	$\sharp(\mathcal{M}_\epsilon)/\sharp(\mathcal{X}^\ell)$
10^{-5}	44	45056	262144	0.17
10^{-4}	19	19456	65536	0.30
10^{-3}	10	10240	16384	0.60
10^{-2}	4	4096	4096	1.00

Fixing $\epsilon = 10^{-2}$ and the block size 16×16, ABR grids $\mathcal{M}_{10^{-2}}$ for the oblique front are presented in Figure 7 (a) and (b), for different interpolation order.

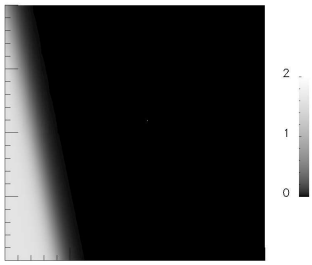


Fig. 5. Oblique-front function.

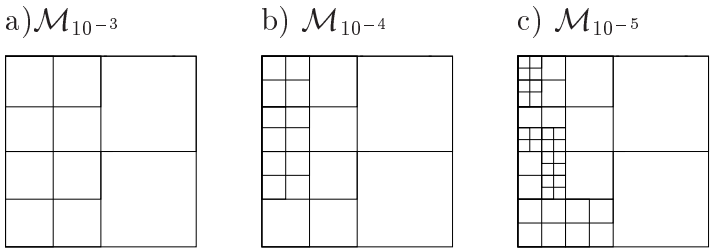


Fig. 6. ABR for the oblique-front function

The number of blocks formed with linear interpolation ($M = 2$) is 34 and with the cubic interpolation ($M = 4$) is 10 . Fixing $M = 4$ and growing the block size to $N_x = N_y = 32$ the number of blocks in $\mathcal{M}_{10^{-2}}$ decreases, as presented in Figure 7 (c). However, despite of the smaller number of blocks, the total number of points increases in comparison with the $N_x = N_y = 16$ mesh in Figure 7 (a). Similar behavior also occurs in relation to different truncation parameter ϵ.

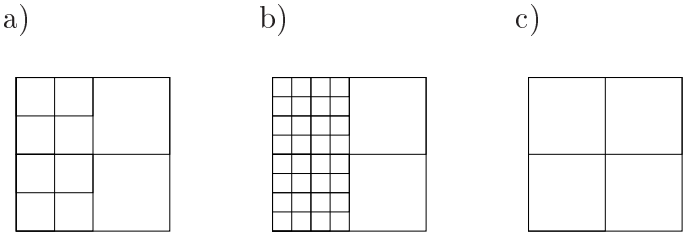


Fig. 7. Oblique front representation in block adaptive mesh $\mathcal{M}_{10^{-2}}$: a) $M = 4$, with 10 blocks 16×16; b) $M = 2$, with 34 blocks 16×16 and c) $M = 4$, with 4 blocks 32×32 .

4 Operations on Block-Structured Grids

For the application of ABR method in the numerical solution of partial differential equations, there are some typical operations involved: grid refinement and coarsening, functional operations and differentiation.

Refining and Coarsening

The automatic adaptation of the grid structure during time evolution requires a simple procedure for refining or coarsening the grids. For block-structured grids, these operations are easily defined in the context of quad-trees by means of tree extension or reduction. In a general sense, if $\mathcal{A}$ and $\tilde{\mathcal{A}}$ are two quad-trees such as $\mathcal{A} \subset \tilde{\mathcal{A}}$, then $\tilde{\mathcal{A}}$ is an extension of $\mathcal{A}$, or $\mathcal{A}$ is a reduction of $\tilde{\mathcal{A}}$. Specifically, the extension and reduction operations of interest are described next.

- Tree Extension (Grid Refinement) : $\mathcal{A} \xrightarrow{\mathcal{E}} \tilde{\mathcal{A}}$

  An extended tree $\tilde{\mathcal{A}}$ may be obtained by adding a new block generation to all leaves of the tree $\mathcal{A}$. Figure 8 shows an example of a block structured grid $\mathcal{M}$ and its corresponding extension $\tilde{\mathcal{M}}$. Given the representation $\left(\mathcal{M}, f^{\mathcal{M}}\right)$, the functional values $f^{\tilde{\mathcal{M}}}$ at the extended grid $\tilde{\mathcal{M}}$ may be not available. In such case, each missing value in $f^{\tilde{\mathcal{M}}}$ is obtained by the interpolatory refinement scheme.

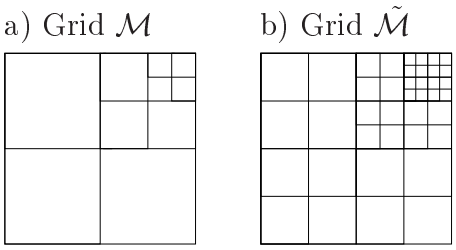


Fig. 8. Extension operation (a) Original grid; (b) Extended grid.

- Tree Reduction (Grid Coarsening): $\tilde{\mathcal{A}} \xrightarrow{\mathcal{T}_\epsilon} \mathcal{A}$

  As done for the grid construction, the reduction operator $\mathcal{T}_\epsilon$ is characterized by a regularity indicator $\imath_\epsilon(\mathcal{B})$. All leaves in $\tilde{\mathcal{A}}$, whose three brothers are also leaves, are tested. If one of these leaves and their brothers have $i_\epsilon = 0$, these nodes are removed from the tree structure. Consequently, their mother becomes a new leaf-node. When this particular case happens, the leaf–structure changes. Therefore, the reduction process must be executed again.

Functional Operations

Operations between two functions represented in block–structured grids are

straightforward point-wise evaluations if their grids coincide. Otherwise, it is necessary to extend both grids in order to get representations in a common grid.

Differentiation

The idea is to use finite difference operators with uniform spacing in each block. For blocks at level $\jmath$, partial derivatives in the x-direction are discretized with spacing $h_x^\jmath$, and in the y-direction with $h_y^\jmath$. For points close to block boundaries, stencil information from neighboring blocks may be required. To avoid demanding search procedures, the process of block construction should consider the addition of needed extra rows and columns around the block boundaries. For instance, in the particular case of forth order discretization of derivatives by a central scheme, two extra rows or columns are necessary at each side of the blocks. The importance in maintaining the block independence, by adding auxiliary extra rows or columns, is crucial for implementations of the method on parallel computers.

5 Application to Evolution Problems

The concepts in the ABR method are particularly suitable for adaptive solvers to evolution partial differential equations. For the applications of this paper, we shall consider equations of the form

$$\frac{\partial \mathcal{U}}{\partial t} = L\,\mathcal{U}$$

where $L(\mathcal{U})$ is a differential operator acting on spatial variables. The solution $\mathcal{U} = \mathcal{U}(x, y, t)$ is searched for $(x, y) \in [0, 1] \times [0, 1]$ and $t > 0$, augmented with initial and boundary conditions.

Suppose that at time $t_n = n\Delta\, t$ an sparse block–grid representation $(\mathcal{M}^n, \mathcal{U}^n)$ for the approximate solution is given, in which $\mathcal{U}^n$ is formed by the numerical solution values at an adaptive mesh $\mathcal{M}^n$.

In the next time step, the representation $(\mathcal{M}^{n+1}, \mathcal{U}^{n+1})$ is obtained after the following procedure.

(1) **Extension:** $(\mathcal{M}^n, \mathcal{U}^n) \xrightarrow{\mathcal{E}} (\mathcal{M}^{n+}, \mathcal{U}^{n+})$

The grid $\mathcal{M}^n$ may not be suitable for the solution at the next time step, since regions of smoothness or irregularities of the solution may change from one step to the next. Therefore, before doing time evolution, the representation of the solution should be extended to a grid $\mathcal{M}^{n+}$, which

is expected to contain $\mathcal{M}^{n+1}$. This refinement stage is very important. In the applications of this paper, the simple one level refinement described previously is adopted.

(2) **Time evolution:** $(\mathcal{M}^{n+}, \mathcal{U}^{n+}) \xrightarrow{\mathcal{L}_a} (\mathcal{M}^{n+}, \breve{\mathcal{U}}^{n+1})$.
A discrete evolution operator $\mathcal{L}_a$ is applied. The action of $\mathcal{L}_a$ includes the the discretization of the partial derivatives in L by uniform finite difference schemes, adapted to each block in $\mathcal{M}^{n+}$, the discretization in time by some explicit ODE solver, an the enforcement of boundary conditions.

(3) **Truncation:** $(\mathcal{M}^{n+}, \breve{\mathcal{U}}^{n+1}) \xrightarrow{\mathcal{T}_\epsilon} (\mathcal{M}^{n+1}, \mathcal{U}^{n+1})$.
Finally, a thresholding operation is applied, in order to unrefine those blocks in $\mathcal{M}^{n+}$ that are unnecessary for an accurate representation of $\mathcal{U}^{n+1}$

5.1 Numerical Examples

A solver that performs these tasks is implemented using a C^{++} object oriented paradigm programming. All the simulations are for 4-th order interpolation, finite differences and Runge-Kutta ODE solver. The time step is dynamically chosen according to the smallest scale $h^n_{\min}$ present in the extended adaptive mesh $\mathcal{M}^{n+}$, such that $h^n_{\min}/\Delta t^n = \lambda$.

Moving Spike

For the linear Advection Equation

$$\frac{\partial \mathcal{U}}{\partial t} + \left(\frac{\partial \mathcal{U}}{\partial x} + \frac{\partial \mathcal{U}}{\partial y} \right) = 0, \tag{2}$$

we shall consider the spiky initial condition

$$\mathcal{U}(x, y, 0) = exp^{-300((x-0.5)^2+(y-0.5)^2)} + 0.2\,\sin(2\pi x) + \sin(2\pi y)$$

and the periodic boundary condition

$$\begin{aligned} \mathcal{U}(0, y, t) &= \mathcal{U}(1, y, t), \qquad 0 \le y \le 1 \\ \mathcal{U}(x, 0, t) &= \mathcal{U}(x, 1, t), \qquad 0 \le x \le 1. \end{aligned}$$

The exact solution is the spike moving along the diagonal $x = y$, without changing its format. Some typical features of the numerical solution and the corresponding adaptive block-grids are presented in Figure 9. For this simulation, the parameters are $\epsilon = 10^{-3}$, $N_x = N_y = 32$ and $\lambda = 5 \times 10^{-2}$. As depicted in Figure 9, at $t = 0.2$ the spike is approaching the down-left corner of the square. As expected, the grid is refined in this quadrant and coarse in the other ones. Similarly, at $t = 0.9$ the spike is already coming from the up-right

corner, and the refinement automatically switches to this region. Figure 10 describes how the number of the blocks in the adaptive mesh evolves during the simulation.

a) $t = 0.2$

b) $t = 0.9$

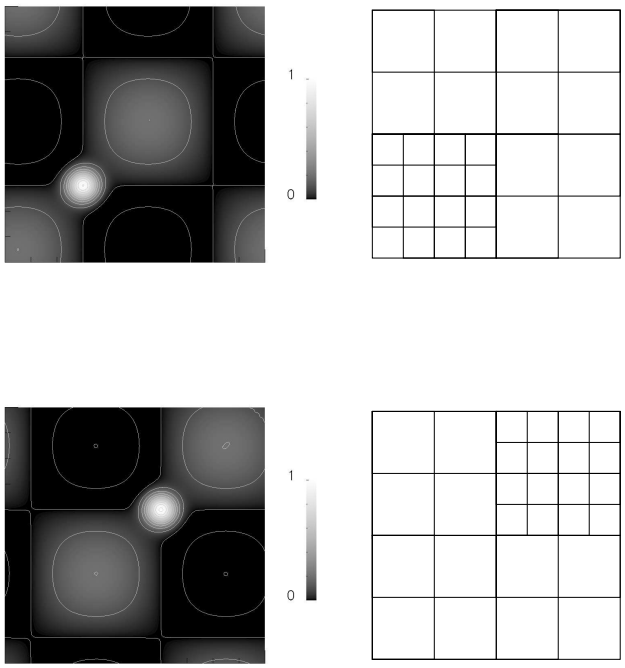


Fig. 9. Advecting spike.

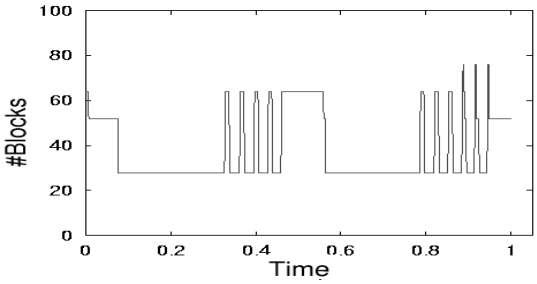


Fig. 10. Number of blocks in the adaptive meshes for the advected spike.

Oblique Front

Let us consider the advection-diffusion equation

$$\frac{\partial \mathcal{U}}{\partial t} + \frac{\partial \mathcal{U}}{\partial x} + \frac{\partial \mathcal{U}}{\partial y} - \frac{\partial^2 \mathcal{U}}{\partial x^2} - \frac{\partial^2 \mathcal{U}}{\partial y^2} = \mathbf{F} \tag{3}$$

for $t \geq 0$, $(x, y) \in [0, 1] \times [0, 1]$. Given the forcing term

$$\mathbf{F}(x, y, t) - \beta\, sech^2\Big(z\Big) - 2(\alpha^2 + \beta^2) sech^2\Big(z\Big)\, tanh\Big(z\Big),$$

where $z = 25(x - t) + 5(y - 1)$, and appropriate boundary conditions, the exact solution is

$$\mathcal{U} = 1 - tanh\Big(z\Big).$$

It describes a propagating steep front moving to the right, as presented in [5]. Figure 11 exhibits some typical features for the numerical solution and the corresponding extended adaptive block-grids. The parameters for this simulation are $\epsilon = 10^{-2}$, $N_x = N_y = 16$ and $\lambda = 10^{-3}$. The Figure 12 shows how the number of blocks in the adaptive mesh evolves during the simulation. The largest number of blocks occurs when the front crosses the center of the region.

Sharp Transition Layers

The following results are for the Burgers' equation

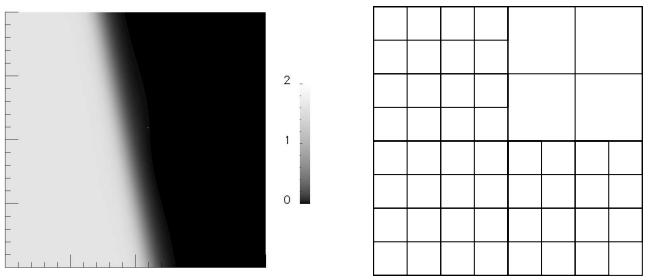
$$\frac{\partial \mathcal{U}}{\partial t} + \mathcal{U}\frac{\partial \mathcal{U}}{\partial x} + \mathcal{U}\frac{\partial \mathcal{U}}{\partial y} - \mu\frac{\partial^2 \mathcal{U}}{\partial x^2} - \mu\frac{\partial^2 \mathcal{U}}{\partial y^2} = 0, \tag{4}$$

$t \geq 0$, $(x, y) \in [0, 1] \times [0, 1]$ and $\mu = 10^{-2}$, with periodic boundary conditions and initial data

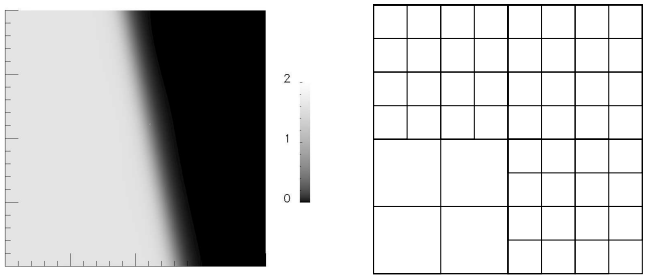
$$\mathcal{U}(x, y, 0) = sen(2\pi x)\; sen(2\pi y).$$

Since negative and positive features move on opposite directions, this example shows sharp transition layers, as time evolves. In the present simulation, the parameters are $\epsilon = 10^{-5}$, $N_x = N_y = 32$ and $\lambda = 5 \times 10^{-2}$. Figure 13 exhibits some typical features of the numerical solution and the corresponding extended adaptive grids. Initially, the function is smooth and the mesh is totally uniform with 16 blocks. At $t = 0.05$, the steepness of the solution requires more refinement, but the grid is still globally uniform. At $t \approx 0.9$, the application of the ABR method produces an irregular mesh with refined blocks close to the sharp transition regions. Figure 14 shows how the number of blocks in the adaptive meshes increases according to the steepness of the solution during the simulation.

a) $t = 0.35$



b) $t = 0.45$



c) $t = 0.70$

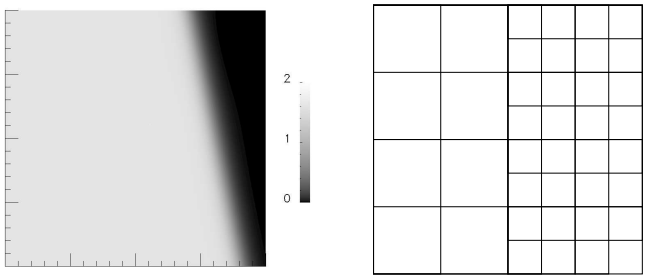


Fig. 11. Oblique front

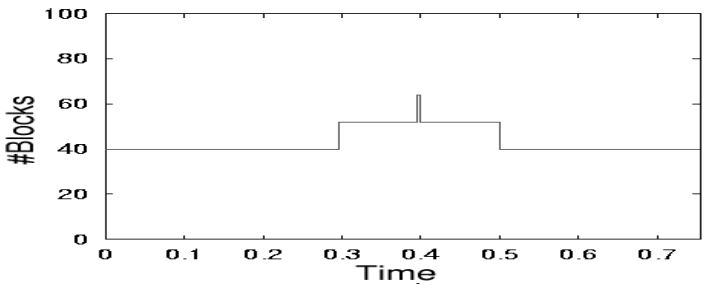
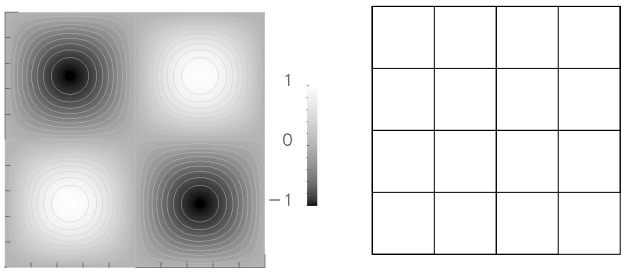
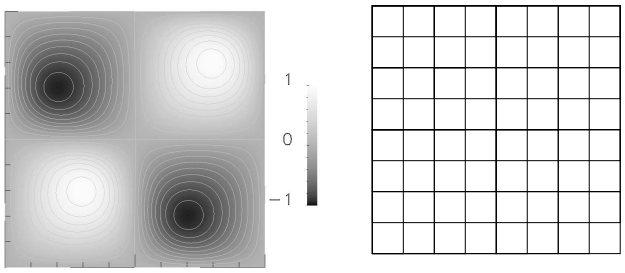


Fig. 12. Number of blocks in the adaptive meshes for the evolution of the oblique front.

a) $t = 0$



b) $t = 0.05$



c) $t = 0.09$

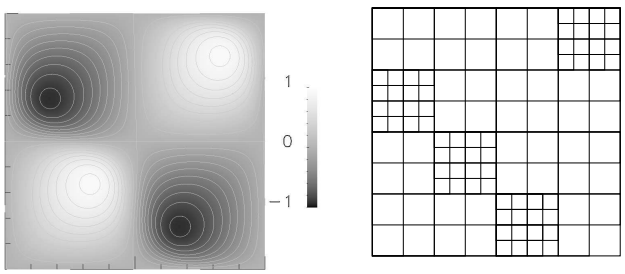


Fig. 13. Sharp transition layer formation.

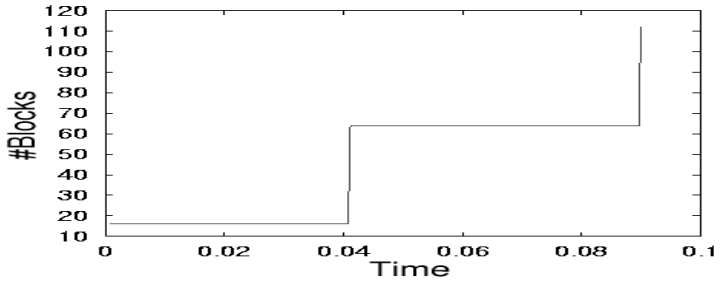


Fig. 14. Number of blocks in the adaptive meshes for the evolution of the sharp transition layer.

6 Conclusions

This work describes an adaptive finite–difference scheme for PDEs based on block–structured grids, which are dinamically generated by wavelet representation techniques. The algorithms and data structure are formulated by using abstract concepts borrowed from quaternary trees. This procedure helps the understanding of the method and its computational implementation. Given any desired accuracy, the method in intended to produce simulations with automatic grid refinement, as required by the numerical solution. For the class of singular structures considered in this paper, gain in local spacial resolution may be obtained without penalizing the computational complexity. Nevertheless, it is worth saying that the method, as stated, faces the typical dilema of adaptive solvers with explicit time discretization. Since, for stability razon, Δt is adjusted to the current finest scale level, the effect of any grid refinement is the increment of the total number of time steps. This aspect affects the computational performance of the method and requires further studies. A typical alternative would be to consider implicit time stepping or to introduce some kind of local time-space adaptivity, for instance, following one of the guidelines indicated in [6,5,7].

References

[1] A. Cohen, Wavelet Methods in Numerical Analysis, in: P. G. Ciarlet, J. L. Lions (Eds.), Handbook of Numerical Analysis, Vol. VII, Elsevier, Amsterdam, 2000.

[2] M. Holmström, Wavelet Based Methods for Time Dependent PDEs, Ph.D. thesis, Uppsala University (1997).

[3] J. Walden, A general adaptive solver for hyperbolic PDEs based on filter bank subdivisions, Appl. Numer. Math. 33 (1-4) (2000) 317–325.

[4] O. V. Vasilyev, C. Browman, Second generation wavelet collocation method for the solution of partial differential equations, J. Comput. Phys. 165 (2000) 660–693.

[5] R. A. Tromper, J. G. Verwer, Runge-Kutta methods and local uniform grid refinement, Math. Comput. 60 (202) (1993) 591–616.

[6] E. Bacry, S. Mallat, G. Papanicolau, A wavelet based space-time adaptive numerical method for partial equations, Math. Model. Numer. Anal. 26 (7) (1992) 793–834.

[7] P. Lötstedt, S. Söderberg, A. Ramage, L. Hemmingsson-Frändén, Implicit solution of hyperbolic equations with space-time adaptivity, BIT 42 (2002) 134–158.