

# Multiresolution analysis on multidimensional dyadic grids

Douglas A. Castro<sup>(1)</sup>, Sônia M. Gomes<sup>(1)</sup>, Anamaria Gomide<sup>(2)</sup>, Andrielber S. Oliveira<sup>(1)</sup>, Jorge Stolfi<sup>(2)</sup>

(1) IMECC-Unicamp, Caixa Postal 6065, CEP 13083-859 Campinas-SP, Brazil.

(2) IC-Unicamp, Caixa Postal 6176, CEP 13081-970 Campinas-SP, Brazil.

{douglas, andriel, soniag}@ime.unicamp.br {anamaria, stolfi}@ic.unicamp.br

## Abstract:

We propose a modified adaptive multiresolution scheme for representing  $d$ -dimensional signals which is based on cell-average discretization in dyadic grids. A dyadic grid is an hierarchy of meshes where a cell at a certain level is partitioned into two equal children at the next refined level by hyperplanes perpendicular to one of the coordinate axes which varies cyclically from level to level. Adaptivity is obtained by interrupting the refinement at the locations where appropriate scale (wavelet) coefficients are sufficiently small. One important aspect of such multiresolution representation is that we can use a binary tree data structure in all dimensions, that helps to compress data while still being able to navigate through it. Dyadic grids provide a more gradual refinement as compared with traditional multiresolution analyses that use, for instance, different quad-trees or oct-trees in 2D or 3D multiresolution applications. The cells may have different scales in different directions, this property can be explored to improve data compression of signals having anisotropic aspects.

## 1. Introduction

In recent years, many multiscale techniques have been used to provide more efficient algorithms than those that use just one level of resolution. In such frameworks, the differences between the information at consecutive levels of refinement are computed, and only the significant coefficients are stored. These are the principles of wavelet compression which have been successfully applied in many different contexts [3]. For example, multiresolution finite volume schemes of Müller [6] and Domingues et al. [4] use adaptive grids that are dynamically obtained by taking local regularity information indicated by wavelet coefficients in the context of multiresolution analysis for cell averages of signals. Such adaptive discretizations allow the efficient solution of problems with vastly different scales of detail in different parts of the domain.

For computational efficiency, one important aspect of such multiresolution methods is the topology of the mesh and data structure used to represent it. Often quad-grids and oct-grids are used for 2D and 3D domains, respectively, represented by quad-tree and oct-tree data structures [1]. We describe here a type of mesh, the *dyadic grid*, that can be efficiently represented by a binary tree, in domains of arbitrary dimension. For illustration, we apply adaptive

dyadic grids to multiresolution analysis, using cell averaging as the discretization method.

The paper is organized as follows. In Section 2 we define dyadic grids and related concepts. In Section 3 we present a general overview of multiresolution analysis. Section 4 contains numerical results on sample problems to show the efficiency of the proposed scheme.

## 2. Dyadic grids

Let the coordinates of  $\mathbb{R}^d$  be indexed from 0 to  $d - 1$ . An infinite *dyadic grid* is a hierarchy of meshes that begins with a  $d$ -cube at level  $k = 0$ , and, for each higher level  $k > 0$ , is the result of dividing each cell of level  $k$  into two equal children by a hyperplane perpendicular to the coordinate axis  $(k \bmod d)$  [2]. Figure 1 illustrates five steps of the refinement process for  $d = 3$ .

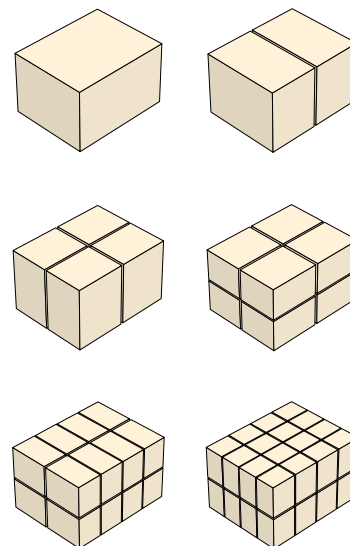


Figure 1: 3D dyadic grids.

In practice, one uses only finite segments of this grid, where the subdivision stops at a maximum level. In a *regular* dyadic grid, the refinement stops at the same level everywhere. In an *irregular* grid, the maximum level varies from place to place.

The topology of a dyadic grid can be represented by a  $0$ - $2$  *binary tree*. This is a data structure consisting of a set

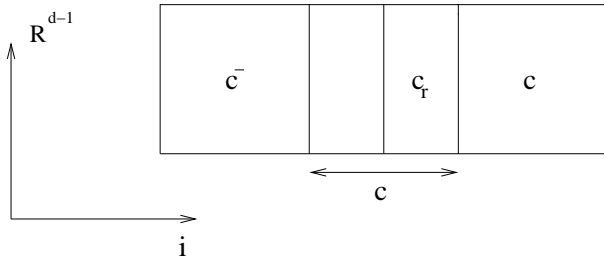


Figure 2: Definition of  $c^-$ ,  $c^+$ ,  $c$ ,  $c_r$ .

of elements named *nodes*, among which there is a special node  $r$ , the *root*; every node has either zero or two children nodes; and every node, except the root, has exactly one parent node. A node that has no children is called a *leaf node*. The children of a non-leaf node  $t$  are called the *left child*  $t_\ell$  and the *right child*  $t_r$ .

Each node of this tree represents a cell that appeared at some level of the subdivision; the leaf nodes represent the cells that weren't divided. By convention, the left child  $t_\ell$  of a non-leaf node  $t$  in level  $k$  represents the "lower" half  $c_\ell$  of the cell  $c$  represented by  $t$ ; that is, the half whose projection on the axis  $i = k \bmod d$  has smallest  $i$ -coordinates.

### 3. Multiresolution analysis

In multiresolution analysis, signals can be represented in two ways, as ordinary samples at each scale, or as differences between two consecutive scales. Connecting these two views are the *prediction* and the *restriction* operators. The prediction operator  $P_k^{k+1}$  takes information from a coarse level  $k$  and gives an estimate for the information at the next finer level  $k+1$ . Conversely, the restriction operator  $P_{k+1}^k$  takes information from a fine level  $k+1$  and gives an estimate of the information at a coarser level  $k$ .

In this paper, the samples of a  $d$ -dimensional signal  $f$  are averages computed over the cells of a  $d$ -dimensional dyadic grid. That is, the sample associated with a cell  $c$  in level  $k$  of the grid is

$$\bar{f}_c^k = \frac{1}{|c|} \int_c f(x) dx, \quad (1)$$

where  $|c|$  is the volume of  $c$ . The restriction operation is therefore (trivially and exactly) the sum of the averages in the children cells,

$$\bar{f}_c^k = \frac{1}{2} [\bar{f}_{c_\ell}^{k+1} + \bar{f}_{c_r}^{k+1}]. \quad (2)$$

In the other direction, we predict the cell average of a child cell  $c_r$  or  $c_\ell$  by the formulas

$$\bar{f}_{c_r}^{k+1} \approx \hat{f}_{c_r}^{k+1} = \bar{f}_c^k + \frac{1}{8} [\bar{f}_{c^+}^k - \bar{f}_{c^-}^k] \quad (3)$$

$$\bar{f}_{c_\ell}^{k+1} \approx \hat{f}_{c_\ell}^{k+1} = \bar{f}_c^k - \frac{1}{8} [\bar{f}_{c^+}^k - \bar{f}_{c^-}^k] \quad (4)$$

where  $c^-$  and  $c^+$  are the two closest neighbor cells of  $c$  at level  $k$  in the direction of refinement. See Figure 2. These estimators are exact for quadratic polynomials.

**Detail coefficients.** In the structure we do not store the averages ( $\bar{f}_c^k$  or  $\hat{f}_c^k$ ), but only the *details* or *wavelet coefficients*. Each detail  $d_c^k$  is the difference between the exact average in the cell  $c$  and the value predicted for it by formulas (3) and (4) from the cell's parent and its neighbors:

$$d_c^{k+1} = \bar{f}_c^{k+1} - \hat{f}_c^{k+1}. \quad (5)$$

Note that the detail of the root cell is not defined.

**Analysis and synthesis.** The *analysis algorithm* computes the details of every cell, given the average values  $\bar{f}_c^k$  for every cell  $c$ . It scans the tree bottom-up, level by level. For each non-leaf cell  $c$  in level  $k$ , it executes

$$\begin{aligned} \bar{\delta} &\leftarrow \frac{1}{2} [\bar{f}_{c_r}^{k+1} - \bar{f}_{c_\ell}^{k+1}]; \\ \hat{\delta} &\leftarrow \frac{1}{8} [\bar{f}_{c^+}^k - \bar{f}_{c^-}^k]; \\ \delta &\leftarrow \bar{\delta} - \hat{\delta} \\ d_{c_r}^k &\leftarrow +\delta; \\ d_{c_\ell}^k &\leftarrow -\delta. \end{aligned} \quad (6)$$

Once the detail  $d_c^k$  of a cell has been computed, its average  $\bar{f}_c^k$  is no longer needed, so we may store the detail in its place. In the root node  $r$ , however, we must still keep the average  $\bar{f}_r^0$  of the function over the whole domain.

The inverse of the analysis algorithm is the *synthesis algorithm*, which recomputes the averages  $\bar{f}_c^k$  from the details. It scans the tree top down, level by level. At each cell  $c$  in level  $k$ , it executes

$$\begin{aligned} \hat{\delta} &\leftarrow \frac{1}{8} [\bar{f}_{c^+}^k - \bar{f}_{c^-}^k]; \\ \bar{\delta} &\leftarrow \hat{\delta} - d_{c_r}^{k+1}; \\ \bar{f}_{c_r}^{k+1} &= \bar{f}_c^k + \bar{\delta} \\ \bar{f}_{c_\ell}^{k+1} &= \bar{f}_c^k - \bar{\delta}. \end{aligned} \quad (7)$$

After this step, the details  $d_{c_r}^{k+1}$  and  $d_{c_\ell}^{k+1}$  of the children are no longer needed, and can be overwritten with the reconstructed averages  $\bar{f}_{c_r}^{k+1}$  and  $\bar{f}_{c_\ell}^{k+1}$ .

**Compact representation.** These algorithms show that knowledge of the cell averages for all leaves is equivalent to knowledge of the average value  $\bar{f}_r^0$  for the root cell together with the detail of every right child cell. To save space, we could store the detail of the right child in its parent's node (and keep the domain average  $\bar{f}_r^0$  in variable external to the tree). Then the leaf nodes would carry no information, and could be omitted from the structure. We will refer to this variant (which is an ordinary binary tree) as the *compact tree representation*.

**Adaptive resolution grid.** As in any wavelet representation, we can save space and processing time by pruning all sub-trees which do not contribute significantly to the reconstructed signal. If we start with a tree of sufficient depth, we can eliminate all sibling leaf nodes  $c_\ell$  and  $c_r$  such that  $|d_c^k|$  falls below a prescribed tolerance  $\epsilon_k$ . This condition implies that the predictions  $\hat{f}_{c_\ell}^{k+1}$  and  $\hat{f}_{c_r}^{k+1}$  will be very close to the actual averages  $\bar{f}_{c_\ell}^{k+1}$  and  $\bar{f}_{c_r}^{k+1}$ . Here we use Harten's thresholds [5],

$$\epsilon_k = (1 - q)q^{(L-k)} \epsilon, \quad (8)$$

where  $q$  and  $\epsilon$  are specified by the user, with  $\epsilon > 0$  and  $0 < q < 1$ , and  $L$  is the maximum level of the initial tree.

#### 4. Numerical results

In order to compare the efficiencies of dyadic grids and quad-grids, we performed the multiresolution analyses of two different examples in 2D, using cell-average discretization. In all tests, the root cell was the rectangle  $[0, 1] \times [0, \frac{\sqrt{2}}{2}]$ , and the starting tree was a uniform grid with  $2^{10} \times 2^{10} = 2^{20} = 1,048,576$  leaf cells. This corresponds to tree structures with  $L = 20$  and  $L = 10$  levels for dyadic grid and quad-grid frameworks, respectively. The cell averages  $f_c^L$  were computed for every leaf cell  $c$  by Gaussian quadrature with  $5 \times 5$  sampling points. The trees were pruned as described in the previous section, with threshold parameters  $\epsilon = 0.1$  and the  $q = 0.5$ . The number of non-leaf nodes in the initial tree was  $\sum_{i=0}^{19} 2^i = 1,048,575$  for the dyadic grid, and  $\sum_{i=0}^9 4^i = 349,525$  for the quad-grid. In the first test, we used the signals

$$f(x, y) = 1 - \tanh(100(x - 0.2 - t) + 0.001(y - 1)), \quad (9)$$

for  $t$  varying from 0 to 0.6 in steps of 0.1. Equation (9) describes a 2D smooth step function with an almost vertical straight front, moving from left to right. Figure 3 shows the dyadic grid and the corresponding tree at  $t = 0.1$ , after pruning cells with small details. Figure 4 shows the corresponding quad-grid and quad-tree. Figure 5 shows the number of leaf cells in both grids for each time step, as a percentage of the number of leaves in the uniform grid.

For the second test, we used the signals

$$f(x, y) = \begin{cases} 1 & \text{if } \sqrt{(x - 0.5)^2 + (y - 0.35)^2} < t, \\ 0 & \text{otherwise.} \end{cases} \quad (10)$$

for  $t$  varying between 0.05 and 0.35 in steps of 0.05. Equation (10) describes a step function with a sharp circular front, expanding from the center of the domain. Figure 6 shows the dyadic grid and its tree at  $t = 0.2$ , and Figure 7 shows the corresponding quad-grid and its quad-tree. The number of leaves is plotted in Figure 8.

**Space efficiency.** If leaves are explicitly represented in the tree structure, and all nodes have the same fields, then the space  $E$  used by the structure is  $E = (pA + B)n$ , where  $n$  is the number of nodes,  $p$  is the number of pointers in each node,  $A$  is the size of a pointer in bytes, and  $B$  is the size of any additional information stored in each node (such as the detail coefficients  $d_c^k$ ). In all these trees we have  $n = (pm - 1)/(p - 1) \approx mp/(p - 1)$ , where  $m$  is the number of leaf nodes.

From the plots in Figure 5, we see that, in the first test, the quad-grid ( $p = 4$ ) had about 8 times as many leaf cells as the dyadic grid ( $p = 2$ ), and therefore about 5 times as many tree nodes, for the same accuracy. Assuming  $A = 4$  and  $B = 8$  bytes, we conclude that the quad-tree used  $5(24/16) \approx 7.5$  times as much storage as the dyadic grid.

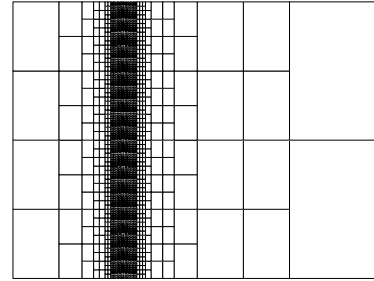
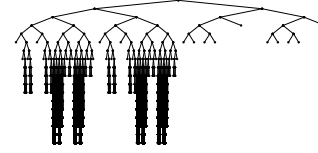


Figure 3: Pruned dyadic tree (top) and dyadic grid (bottom) for the first signal at  $t = 0.1$ .

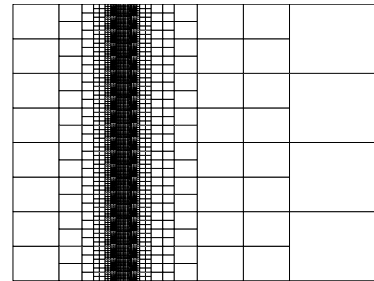
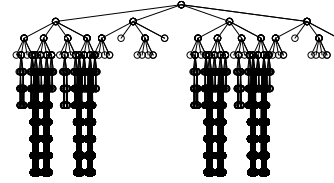


Figure 4: Pruned quad-tree (top) and quad-grid (bottom) for the first signal at  $t = 0.1$ .

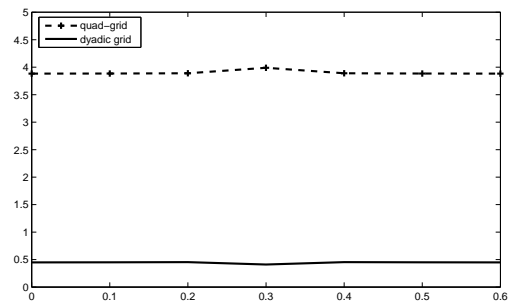


Figure 5: Leaf count in the pruned trees for the first test.

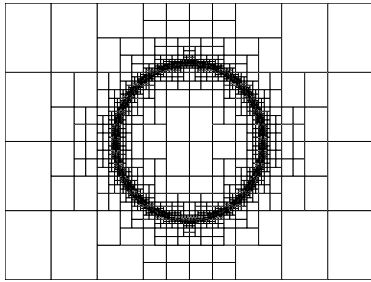
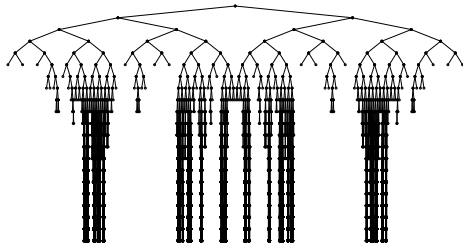


Figure 6: Pruned dyadic tree (top) and dyadic grid (bottom) for the second signal at  $t = 0.2$ .

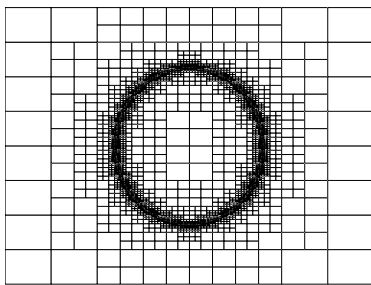
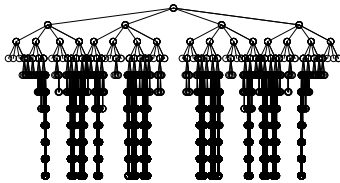


Figure 7: Pruned quad-tree (top) and quad-grid (bottom) for the second signal at  $t = 0.2$ .

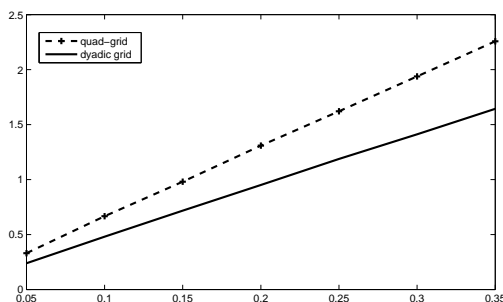


Figure 8: Leaf count in the pruned trees for the second test.

In the second test, the quad-grid had about 1.32 times as many leaf cells as the dyadic grid, and therefore about 0.88 as many tree nodes. With the same  $A$  and  $B$ , the quad grid still used  $0.88(24/16) \approx 1.32$  times as much space as the dyadic grid.

Had we used the compact representation of the tree, with omitted leaves, the storage cost would be  $E = (pA + (p - 1)B)(n - m)$ . The quad-tree would use 7.5 times as much storage as the dyadic tree in the first example, and 1.1 times as much in the second example.

## 5. Conclusions

Our tests show that adaptive dyadic grids are substantially more efficient than quad-grids for the same level of accuracy, both in terms of space needed to store the topology (tree structure) of the grid, and in the number of leaf cells retained — which determines the time cost of most adaptive numeric algorithms.

## 6. Acknowledgments

The authors thank CNPq (grants 06631/07-5, 472402/07-2, and 142191/06-0) and FAPESP (07/52015-0) for financial support.

## References:

- [1] B. L. Bihari and A. Harten. Multiresolution schemes for the numerical solution of 2-d conservation laws. *SIAM J. Sci. Comput.*, 18(2):315–354, 1997.
- [2] C. G. S. Cardoso, M. C. Cunha, A. Gomide, D. J. Schiozer, and J. Stolfi. Finite elements on dyadic grids with applications. *Mathematics and Computers in Simulation*, 73:87–104, 2006.
- [3] A. Cohen. *Wavelet Methods in Numerical Analysis. Handbook of Numerical Analysis.* in: Ph. Ciarlet and J. L. Lions (Eds.), *Handbook of Numerical Analysis*, Vol VII, Elsevier, Amsterdam, 2000.
- [4] M. O. Domingues, S. M. Gomes, O. Roussel, and K. Schneider. An adaptive multiresolution scheme with local time-stepping for evolutionary pdes. *Journal of Computational Physics*, 227:3758–3780, 2008.
- [5] A. Harten. Multiresolution representation of cell-averaged data. Technical Report CAM/Report/94-21, UCLA, Los Angeles, US, July 1994.
- [6] S. Muller. *Adaptive Multiscale Schemes for Conservation Laws.* Vol. 27 of *Lecture Notes in Computational Science and Engineering*, Springer, Heidelberg, 2003.